Character-level Tagging

by

Onur Kuru

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

KOÇ
UNIVERSITY

August, 2016

Character-level Tagging

Koç University

Graduate School of Sciences and Engineering

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

Onur Kuru

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. Arzucan Özgür

_____

Assoc. Prof. Engin Erzin

Date: _____

To my beloved family

ABSTRACT

I describe and evaluate a language-independent character-level tagger for sequence labeling problems: Named Entity Recognition (NER), Part-of-Speech (POS) tagging and Chunking. Instead of words, a sentence is represented as a sequence of characters. The model consists of stacked bidirectional LSTMs which input characters and output tag probabilities for each character. These probabilities are then converted to consistent word level phrase tags using a Viterbi decoder. The model uses only labeled data and does not rely on hand-engineered features or other external resources like syntactic taggers or Gazetteers. The model is able to achieve close to state-of-the-art NER performance in seven languages, performs as well as or better than previous work in four languages for POS tagging and yields competitive results for English Chunking dataset.

ÖZETÇE

Bu tezde dilden bağımsız, karakter seviyesinde etiketleme yapan bir model tarif ettim ve bu modeli Varlık İsmi Tanıma, Sözcük Türü Etiketleme ve Yüzeysel Çözümleme üzerinde değerlendirdim. Bu modelde bir cümle, kelimeler yerine karakter dizisi olarak temsil edildi. Model, karakterleri girdi olarak alan ve her karakter için etiket olasılık dağılımı üreten, üst üste yığılmış çift yönlü Geri Dönüşümlü Yapay Sinir Ağları'ndan oluşmaktadır. Daha sonra bu olasılık dağılımları, Viterbi algoritması kullanılarak kelime seviyesindeki etiketlere çevirildi. Model, sadece işaretli veri kümesi kullanmakta olup, elle belirlenmiş özniteliklere ve harici kaynaklardan gelen bilgilere (diğer sözdizimsel işaretleyicilerin çıktısı, isim listeleri) ihtiyaç duymamaktadır. Bu tezde tarif edilen model, Varlık İsmi Tanima'da 7 dilde elde edilen en iyi sonuçlara yakın, Sözcük Türü Etiketleme'de 4 dilde geçmiş sistemlerden daha iyi ve İngilizce Yüzeysel Çözümleme'de rekabetçi sonuçlar vermiştir.

ACKNOWLEDGMENTS

I am grateful to my advisor Deniz Yuret for giving me the opportunity to work with him and granting me the freedom to explore whatever I wanted.

I am fortunate to be a member of Koc AI-Lab where I spent invaluable time with brilliant people. Among them, Ozan Arkan Can deserves a special mention for sharing my interest in pursuing crazy ideas and his help.

I also would like to thank Arzucan Özgür and Engin Erzin for participating in my defense committee.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Tagging problems are commonly formulated as word-level classification problems where each word in the sentence is mapped to a tag depending on the task at hand. A typical approach is to slide a window over each word position to extract features for a classifier that produces tags. Moreover, one can allow the classifications at adjacent positions to interact by chaining local classifiers together and perform joint inference. To achieve good performance, one has to overcome the data sparsity problem in the labeled training data. This is achieved by handcrafting good word-level features by exploiting affix, capitalization, or punctuation (Ratnaparkhi et al., 1996; Zhang and Johnson, 2003), using the output of syntactic analyzers (part-of-speech taggers, chunkers) and external resources such as gazetteers, word embeddings, word cluster ids to improve performance further (Turian et al., 2010; Ratinov and Roth, 2009). These solutions require significant engineering effort or language specific resources that are not readily available in all languages of interest.

It is even harder to design a multilingual model with handcrafted features considering each language offers different challenges. A distinguishing feature for one language may not be informative for another. For example, capitalization (a typical feature for English NER) is not a distinguishing orthographic feature for the Arabic script. Models for languages with agglutinative or inflectional morphologies may need to utilize the output of a language specific morphological analyzer. In some morphologically rich languages, production of hundreds of words from a given root is common, which makes models even more susceptible to the data sparsity problem.

This work is motivated by the desire to eliminate the tedious work of feature

engineering, language specific syntactic and morphological analyzers, and language specific lexical or named-entity resources which are considered necessary to accomplish good performance. I accomplish this by combining the following ideas: First, instead of considering entire words as the basic input features, I take the characters as the primary representation as in (Klein et al., 2003; Gillick et al., 2015). Second, I use a stacked bidirectional Long Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997) which is able to operate on sequential data of arbitrary length and encode observed patterns in its memory at different scales. Finally, I use a Viterbi decoder to convert the character level tag probabilities produced by the LSTM into consistent word level tags.

Considering characters as the primary representation proves fruitful in several ways. Characters provide sub-word-level syntactic, morphological, and orthographic information that can be directly exploited by my model, whereas word-based models have to incorporate this information using feature engineering. Furthermore, useful sub-word features may vary from language to language, which my model can automatically learn but word-based models would have to incorporate using language-specific resources and features. Finally, character-based models reduce the size of the input vocabulary compared to word-level models, using fewer parameters, increasing computational and statistical efficiency.

I report results similar to or better than the state-of-the-art in resource-free Named Entity Recognition in seven languages. Moreover, I evaluate my model on four languages in Part-of-Speech tagging and outperform previous work. Furthermore, I report competitive results on English Chunking task.

The structure of this thesis is organized as follows:

Chapter 2 gives a brief description of Named Entity Recognition, Part-of-Speech tagging and Chunking tasks.

Chapter 3 outlines related work on tagging tasks with word- and character-level in-

puts, then summarizes the applications of character-based models in NLP in general.

Chapter 4 details the model and describes the input/output representation, stacked bidirectional LSTMs and inference details.

Chapter 5 describes evaluation datasets in detail, presents results and demonstrates ablation experiments.

Chapter 6 summarizes the contributions and discusses future work.

Appendix contains information about datasets used in this thesis.

Chapter 2

TASKS

In this chapter, I give a brief description of the tagging (sequence labeling) problems that I attempted to solve, namely, Part-of-Speech tagging, Chunking and Named Entity Recognition. In a tagging problem, the goal is to build a model which inputs a sentence and outputs a tag sequence. The tag sequence is the same length as the input sentence, and therefore specifies a single tag for each word in the sentence.

2.1 Part-of-speech Tagging

In Part-of-speech (POS) tagging, the goal is to tag each word in a sentence with its corresponding part of speech (noun, verb, adjective etc.). Part of speeches are divided into categories such that words with the same part of speech display similar behaviour in terms of syntax. One of the main challenges in POS tagging is ambiguity. A word can take several possible parts of speech depending on the context it appears. I provide an example sentence tagged with POS tags:

> Pierre/NNP Vinken/NNP 61/CD years/NNS old/JJ will/MD join/VB the/DT board/NN as/IN a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD ./.

In this sentece, POS tag of every word is represented next to it. For more details about POS tags, see Appendix A.3 and Appendix A.4.

2.2 Chunking

In chunking, the goal is to divide an input sentence into syntactically related non-overlapping groups of words called chunks or phrases. These phrases are non-overlapping

such that one word can only be a member of one chunk. I give an example sentence below:

> [NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only £ 1.8 billion] [PP in] [NP September] .

Chunks have been represented as groups of words between square brackets. Type of the chunk is indicated next to the open bracket. NP, VP, PP stands for noun phrase, verb phrase and preposition phrase respectively. More information for phrase types is provided in Appendix A.5.

2.3 Named Entity Recognition

Named Entity Recognition task involves finding names in text. Categories of the names can be persons, organizations, locations, etc. Example:

> [PER Wolff] , currently a journalist in [LOC Argentina] , played with [PER Del Bosque] in the final years of the seventies in [ORG Real Madrid] .

This sentence contains four named entities: *Wolff* and *Del Bosque* are person names, *Argentina* is a location and *Real Madrid* is an organization.

2.4 Tagging Representation

In this section, I represent all three task in a common format. POS tags are assigned to each word in a sentence, however, NER and Chunking phrases may span over multiple words. I use the BIO scheme for encoding phrase types to word-level tags (see Appendix A.1 for details). I provide an example sentence in Table 2.1.

In this representation, beginning of a phrase is indicated by *B-* prefix and *I-* prefix signals continuation of the same phrase. *O* tag is used to denote that the token belongs to no phrase. Since POS tags are always affiliated with one word, tag prefixes *(B-, I-)* are not used.

Words	POS	Chunking	NER
Onur	NNP	B-NP	B-PER
Kuru	NNP	I-NP	I-PER
is	VBZ	B-VP	O
a	DT	B-NP	O
research	NN	B-NP	O
assistant	NN	I-NP	O
at	IN	B-PP	O
Koc	NNP	B-NP	B-ORG
University	NNP	I-NP	I-ORG
.	.	O	O

Table 2.1: An example sentence tagged for POS tagging, Chunking and NER.

2.5 Common features of tagging problems

Tagging tasks are commonly formulated as word based classification problems where one defines features for words in each position. These features can be grouped in four main categories as: context, morphological, orthographic and external. I summarize the common features to all three task in Table 2.2.

Moreover, output of one task is utilized in another such as POS tags are good indicators for Chunking and both POS tags and phrases are important for NER. Furthermore, word cluster ids, word embeddings, trigger words (gazetteers) are utilized as external resources depending on the task.

Context	Current word
	Surrounding tags
	Surrounding words
Morphological	Suffixes
	Prefixes
Orthographic	Contains uppercase character
	Contains number
	Contains hyphen
	Capitalized and mid. sentence
	All letters uppercase
External	Word cluster IDs
	Word embeddings
	Output of morphological analyzers
	Trigger words (Gazetteers)

Table 2.2: Common features of tagging problems.

Chapter 3

RELATED WORK

In this chapter, I first outline the previous work on tagging tasks with word-level inputs then move onto character-based tagging models. Next, I summarize the applications of character-based models in NLP in general.

3.1 Word based tagging

In this section, I outline the tagging systems with word level inputs in the order of NER, POS tagging and Chunking.

Early successful studies on NER use hand-crafted features and language specific name lists with word-level classifiers. Both of the first place submissions in CoNLL-2002 (Spanish & Dutch), CoNLL-2003 (English & German) NER shared tasks (Carreras et al., 2002; Florian et al., 2003) use a rich set of handcrafted features along with gazetteers to achieve top performance. Subsequently, semi-supervised approaches (Ando and Zhang, 2005; Suzuki and Isozaki, 2008; Turian et al., 2010) have reported better results by utilizing large unlabeled corpora. In addition to CoNLL shared task datasets, other line of research studied morphologically rich and inflectional languages. Demir and Ozgur (2014) employs a semi-supervised learning approach to achieve best result for Czech. Darwish (2013) exploits cross-lingual features and knowledge bases from English data sources to achieve the top performance on Arabic. The current state-of-the-art system for Turkish (Seker and Eryigit, 2012) is based on Conditional Random Field (CRF) and utilizes language dependent features along with gazetteers.

Initial explorations on Part-of-speech tagging are linear statistical models which include Hidden Markov Models (HMM), Maximum Entropy Markov Models (MEMM) (Ratnaparkhi et al., 1996; McCallum et al., 2000), Conditional Random Field (CRF)

(Lafferty et al., 2001). Collins (2002) proposed Structured Perceptron algorithm as an alternative to MEMMs or CRFs and demonstrated improved results. Collobert et al. (2011) use a multilayer neural network (NN) architecture which consists of convolutional neural network (CNN) taking word contexts as the input and a CRF layer on the output. (Huang et al., 2015) employ bidirectional LSTM over the sentence to make use of both past and future input features and uses a CRF layer on the output similar to (Collobert et al., 2011).

Kudoh and Matsumoto (2000) won the CoNLL-2000 challenge on chunking with an SVM based approach by utilizing the window around the word of interest, POS tags and surrounding tags. Subsequently, CRF based models (Sha and Pereira, 2003; McDonald et al., 2005; Sun et al., 2008) which offer advantages over both generative models like HMMs and classiers applied at each sequence position reported better results. Shen and Sarkar (2005) obtained the best result with a voting classifier scheme, where each classifier is trained on different tagging schemes (IOB, IOE, ...). Also, neural network architectures with word level inputs (Collobert et al., 2011; Huang et al., 2015) reported results comparable to CRF based models.

3.2 Character based tagging

Here I summarize the previous work which make use of character representations and reported results on NER, POS tagging and Chunking.

Klein et al. (2003) introduce a Hidden Markov Model (HMM) with character-level inputs to alleviate the data sparsity problem inherent in word-level inputs. Their character-level HMM achieves a 30% error reduction over an HMM with word-level inputs. However, the character-level HMM suffers from unrealistic independence assumptions and it is not able to compete with their best system where they utilize a maximum-entropy conditional Markov model using richer set of features (word, all substrings of the word, part-of-speech and chunk tags). Ma and Hovy (2016) utilize pretrained word embeddings and character-level representation of a word by using a combination of Convolutional Neural Network (CNN), bidirectional LSTM and CRF.

They report both of the best published result for NER and POS tagging on English dataset. Lample et al. (2016) also utilize characters to construct word representations with LSTM-CRF model and relies on unsupervised word representations extracted from unannotated corpora. They announce the best results for German and Spanish NER datasets. (dos Santos and Zadrozny, 2014) replace hand-crafted word-level features with character embeddings in a semi-supervised model and evaluate it on WSJ portion of Penn Treebank for POS tagging, later (dos Santos et al., 2015) for Spanish NER. They learn character embeddings with convolutional neural networks from a labeled training set and use these alongside word embeddings pretrained on huge unlabeled corpus. Ling et al. (2015a) first construct vector representations of words by composing characters using bidirectional LSTMs and then uses another bidirectional LSTM to determine the Part-of-Speech tag of each word. Gillick et al. (2015) adapt the sequence-to-sequence model used for machine translation (Sutskever et al., 2014) to Part-of-Speech tagging and NER. The model inputs the text as sequence of bytes and outputs span annotations of the form (phrase start byte, length of the phrase in bytes, type of the phrase). My model has a similar input/output representation with the character-based HMM of (Klein et al., 2003) and employs a much more compact network than (Gillick et al., 2015). To the best of my knowledge, no previous character based model reported results for Chunking.

3.3 Other character-based models

Character-based models have been used successfully for other NLP tasks as well. Depending on the nature of the task, characters are utilized in two different ways. One line of work uses characters to form a word representation for each token in a sentence (Kim et al., 2015; Ling et al., 2015a; Ballesteros et al., 2015). Alternatively character representations are used without mapping to words first (Klein et al., 2003; Zhang and LeCun, 2015; Ling et al., 2015b). and Kim et al. (2015) employs a convolutional neural network (CNN) over characters to form word representations. Ling et al. (2015a) achieve state-of-the-art results in language modeling by utilizing word

representations modeled by bidirectional LSTMs. Kim et al. (2015) use word representations constructed by CNN with recurrent neural network for language modeling. They show that taking characters as the primary representation is sufficient to encode both semantic and orthographic information and their model is on par with the existing state-of-the-art despite having significantly fewer parameters. Ballesteros et al. (2015) employs the same strategy with (Ling et al., 2015a) to represent each token for a continuous-state dependency parsing model. They show that the parsing model benefits from incorporating the character-based encodings of words for morphologically rich languages.

Zhang and LeCun (2015) demonstrate that convolutional neural networks are successful at mapping characters directly to ontology/sentiment classes and text categories. Ling et al. (2015b) introduce a neural machine translation model that views the input and output sentences as sequences of characters rather than words. They show that the model is capable of interpreting and generating unseen word forms. They achieve translation results that are on par with conventional word-based models.

Chapter 4

MODEL

In this chapter, I outline the architecture of my model. First, I define the input/output representation. Next, I provide preliminary background on Recurrent Neural Networks (RNNs), Long Short-Term Memory RNNs (LSTMs) and bidirectional LSTMs. Then I explain the deep stacked bidirectional LSTM network that I used in the model. Finally, I detail the decoding process.

4.1 Input/Output Representation

This section details the input/output representation of a (sentence, tag sequence) pair both word- and character-level.

	John	works	for	Globex	Corp.	.
NER:	B-PER	O	O	B-ORG	I-ORG	O
POS:	NNP	VBZ	IN	NNP	NNP	.
CHU:	B-NP	B-VP	B-PP	B-NP	I-NP	O

	J	o	h	n		w	o	r	k	s		f	o	r		G	l	o	b	e	x		C	o	r	p	.		.
NER:	P	P	P	P	O	O	O	O	O	O	O	O	O	O	O	O	G	G	G	G	G	G	G	G	G	G	G	O	O
POS:	N	N	N	N	O	V	V	V	V	V	O	I	I	I	O	N	N	N	N	N	N	O	N	N	N	N	N	O	.
CHU:	N	N	N	N	O	V	V	V	V	V	O	I	I	I	O	N	N	N	N	N	N	N	N	N	N	N	N	O	O

Figure 4.1: An example sentence with word level and character level tags for NER, POS tagging and Chunking (CHU).

Since POS tagging, Chunking and NER are proposed as a word-level tagging

problems, all of the proposed data sets use word-level tags. In Chunking and NER, a phrase may span multiple words, hence a word tag is composed of concatenation of a position indicator (B- Beginning, I- Inside) and a phrase type (NP, VP, PER, ORG, LOC, ...). A phrase starts with the B- tag and if it consists of multiple words, the following word tags are prefixed with I-. In addition, an O tag indicates that the word is not inside a phrase. In Part-of-Speech tagging, position indicator prefixes are not employed and syntactic categories (NN, VB, JJ, ...) are used directly.

My model, however, examines a sentence as a sequence of characters and outputs a tag distribution for each character. Therefore, I convert word-level tags to character-level tags. Figure 4.1 shows word-level and character-level tags for an example sentence. I abandon the position indicator prefixes (B-, I-) and use phrase types (NP, VP, PER, ORG, ...) directly as character tags. Notice that I keep delimiters (internal spaces) between words at character-level representation. If a character subsequence in a sentence constitutes a phrase, all of the characters in that subsequence (including internal spaces) receive the same phrase tag. Otherwise, characters get the outside tag (O).

4.2 Neural Architecture

Here I provide preliminary background on Recurrent Neural Networks (RNNs), Long Short-Term Memory RNNs (LSTMs), bidirectional LSTMs and explain the deep BLSTM network that I used in my model.

4.2.1 Recurrent Neural Networks

This section explains the general structure of Recurrent Neural Networks.

Recurrent Neural Networks (RNN) have become first class citizens in natural language processing including language modeling (Mikolov et al., 2010), parsing (Dyer et al., 2015), and machine translation (Cho et al., 2014; Sutskever et al., 2014). A simple RNN –called Elman type network (Elman, 1990)– calculates outputs for a given sequence of inputs using its own memory. Figure 4.2 demonstrates an RNN unfolded

in time. I give the formal equations of the network below and denote matrices with bold upper case letters and vectors with lower case letters. Subscripts of matrices and vectors indicate the connection of the matrix: as letter on the right is *from* and letter on the left is *to* (e.g. $\boldsymbol{W_{hx}}$: input to hidden weight matrix).

$$h_t = f(\boldsymbol{W_{hx}} x_t + \boldsymbol{W_{hh}} h_{t-1} + b)$$
$$y_t = g(\boldsymbol{W_{yh}} h_t + c)$$

where f is the activation function (e.g. tanh function) and g is the nonlinearity function for the output (e.g. $softmax$ function). W_{hx}, W_{hh}, W_{yh} are weight matrices between inputs and hiddens, among hiddens, hiddens to outputs, b and c stand for biases.

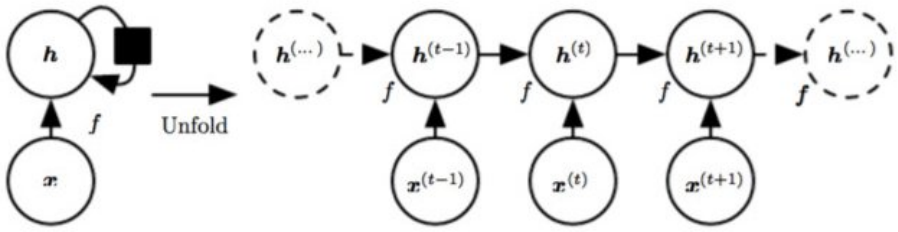

Figure 4.2: Recurrent neural netwok unfolded in time. (Bengio and Courville, 2016)

4.2.2 Long Short-Term Memory

In this section, I detail the Long Short-Term Memory architecture which overcomes the long term dependency problem in Elman type RNN.

One major problem with simple RNNs is that they are difficult to train for long term dependencies due to the vanishing and the exploding gradient problems (Bengio et al., 1994). Hochreiter and Schmidhuber (1997) developed Long Short-Term Memory (LSTM) to overcome the long term dependency problem. They introduced a special memory cell and structures (called gates) to optionally let information through.

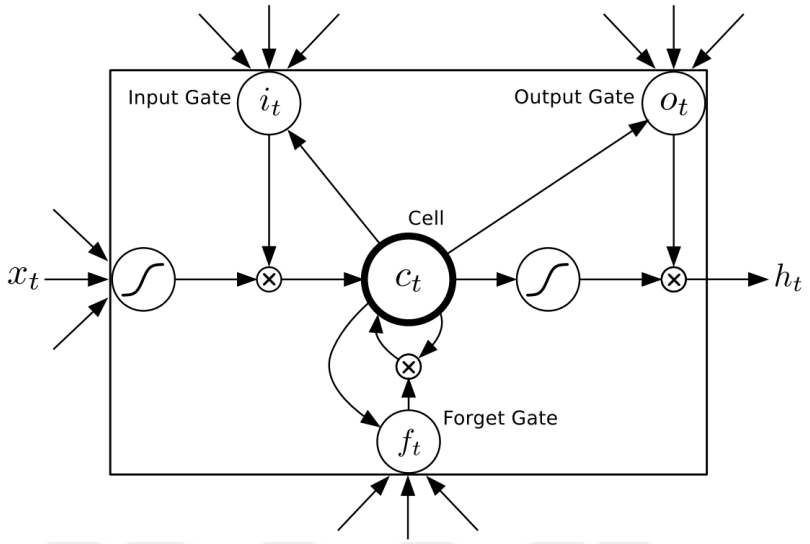

Figure 4.3: Long Short-Term Memory unit. (Graves et al., 2013)

The memory cell is controlled by input, output, and forget gates to protect and control its state. The input gate controls how much new information should be added to current cell, the forget gate controls what old information should be deleted. The output gate controls the information flow from the cell to the output. Figure 4.3 demonstrates an LSTM unit. Many variants of the LSTM have been developed, in this study I use the LSTM architecture with peephole connections which was proposed by Gers et al. (2000). The LSTM memory cell is defined by the following equations:

$$
\begin{aligned}
f_t &= \sigma(\boldsymbol{W_{fx}} x_t + \boldsymbol{W_{fh}} h_{t-1} + w_{fc} * c_{t-1} + b_f) \\
i_t &= \sigma(\boldsymbol{W_{ix}} x_t + \boldsymbol{W_{ih}} h_{t-1} + w_{ic} * c_{t-1} + b_i) \\
c_t &= f_t * c_{t-1} + i_t * \tanh(\boldsymbol{W_{cx}} x_t + \boldsymbol{W_{ch}} h_{t-1} + b_c) \\
o_t &= \sigma(\boldsymbol{W_{ox}} x_t + \boldsymbol{W_{oh}} h_{t-1} + w_{oc} * c_t + b_o) \\
h_t &= o_t * \tanh(c_t)
\end{aligned}
$$

where σ is the logistic sigmoid function, tanh is the hyperbolic tangent function and $*$ is element wise multiplication. f, i and o are the forget, input and output gates respectively, c denotes the cell vector, and h is the hidden state vector. All gate

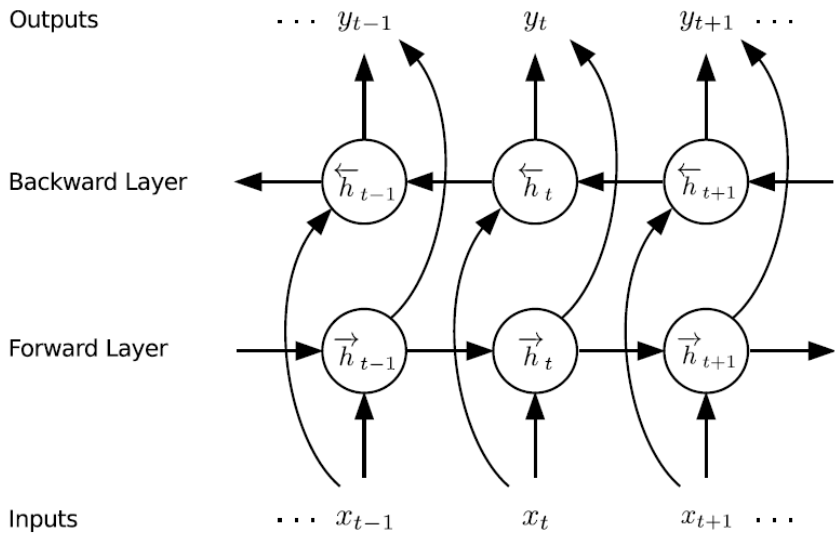

Figure 4.4: Bidirectional LSTM in time. (Graves et al., 2013)

vectors and the cell vector have the same dimensionality as the hidden state vector.

4.2.3 Bidirectional LSTMs

This section describes the bidirectional LSTMs which utilize both past and future inputs when making calculations on the current input.

It is a common approach to use both preceding and following tokens to derive features for the current token in natural language processing tasks. If one were to look at the LSTM equations, the current output depends only on previous inputs, the initial cell value and hidden state. Graves and Schmidhuber (2005) proposed bidirectional LSTM (BLSTM) to gain information from following inputs. In a BLSTM, two LSTM components are present, namely the forward LSTM and the backward LSTM (see Figure 4.4). The forward LSTM traverses the sequence in the forward direction and the backward LSTM traverses same sequence in the reverse order using h_{t+1} and c_{t+1} instead of h_{t-1} and c_{t-1} for the gate calculations. The backward LSTM equations for

time t are the following:

$$f_t = \sigma(\boldsymbol{W_{fx}} x_t + \boldsymbol{W_{fh}} h_{t+1} + w_{fc} * c_{t+1} + b_f)$$
$$i_t = \sigma(\boldsymbol{W_{ix}} x_t + \boldsymbol{W_{ih}} h_{t+1} + w_{ic} * c_{t+1} + b_i)$$
$$c_t = f_t * c_{t+1} + i_t * \tanh(\boldsymbol{W_{cx}} x_t + \boldsymbol{W_{ch}} h_{t+1} + b_c)$$
$$o_t = \sigma(\boldsymbol{W_{ox}} x_t + \boldsymbol{W_{oh}} h_{t+1} + w_{oc} * c_t + b_o)$$
$$h_t = o_t * \tanh(c_t)$$

In a bidirectional model the output at time t depends on both the forward hidden state $\overrightarrow{h_t}$ and the backward hidden state $\overleftarrow{h_t}$.

4.2.4 Deep BLSTMs

In this section I outline deep BLSTMs where a layer of bidirectional LSTM hidden states are fed as input to the next layer of BLSTM.

My model uses bidirectional stacked LSTMs (Graves et al., 2013) to map character sequences to tag sequences. Figure 4.5 demonstrates the model overview. The network takes characters as the input sequence and each character is fed into the first forward and backward LSTM layers as one-hot vectors. The output of the first forward and backward layers are concatenated and fed into the next layer. The same process is carried on for the additional BLSTM layers. To obtain the distribution over the tag at position t, an affine transformation followed by a softmax is applied to the hidden representation of the final BLSTM.

4.3 Decoder

The deep BLSTM gives a tag distribution for each character position. In this section I discuss the final step of turning these character level tag distributions into word level tags.

In early experiments, I observed that the most probable character tags within a word were not always consistent. For example, the model may assign higher probability to person (P) tags in the beginning of a word and organization (G) tags at the

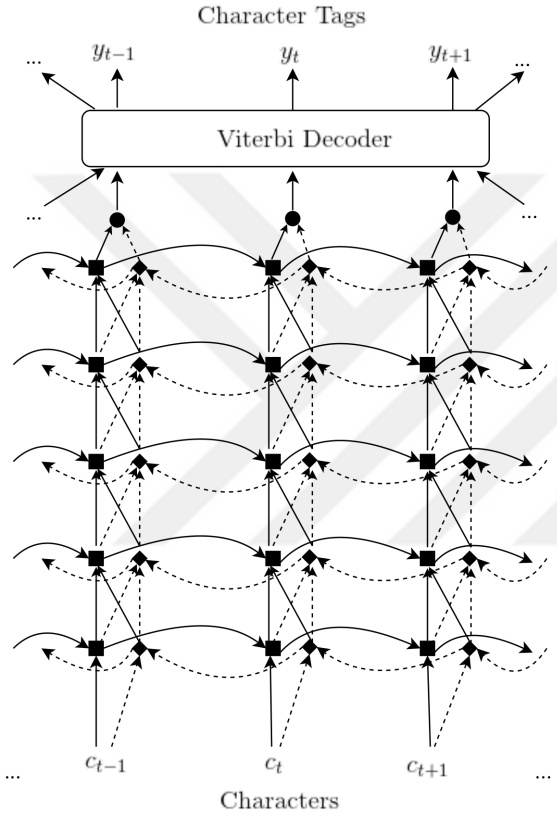

Figure 4.5: 5-layer Bidirectional LSTM Network with a Viterbi decoder. Each square and diamond node represents a forward and backward hidden LSTM layer, respectively. Circles denote the output layer (i.e. softmax layer). Solid lines show forward connections and dashed lines show backward connections. The model takes characters as an input sequence and each character is represented with a one-hot vector (c_t). A Viterbi decoder takes the sequence of character tag probabilities that is produced by the softmax layer and produces most likely character tag sequence (y) that is consistent at the word level.

end of the same word. Even though the deep BLSTM has access to both left and right input contexts, it is unable to learn word level consistency for output tags. To remedy this, I use a decoder similar to (Wang et al., 2015).

Given a character sequence $c_1, c_2, ..., c_n$ (henceforth denoted with $[c]_1^n$), and a tag set $y \in \mathcal{Y}$, the decoder takes output tag probabilities from the LSTM, $o(c_i)_{y_i} = p(y_i \,|\, [c]_1^i, [c]_i^n)$, as emission probabilities and exploits transition matrices, A_{ij}, that only allow tags consistent within a word. Three types of transitions can occur between consecutive character tags (y_{i-1}, y_i) according to the position of the character at hand.

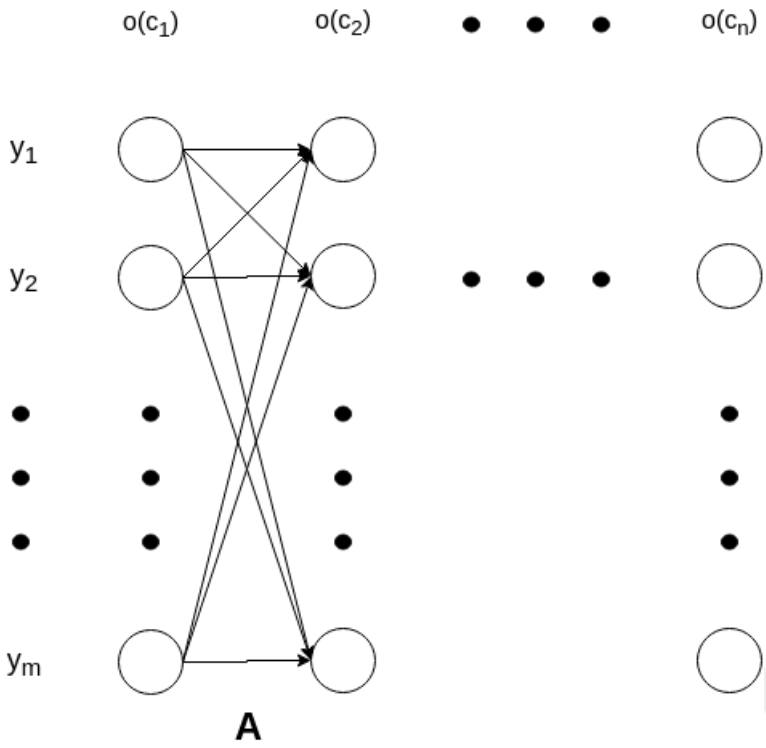

Figure 4.6: Decoder for n characters and m tags. Circles in the same column represent a tag distribution for a character position. A stands for the utilized transition matrices.

A character is either followed by a fellow character in the same word ($c \to c$) or it can be the last character of a word followed by space ($c \to s$) where c denotes a character inside word boundaries and s denotes the delimiter space. Finally, it can be a space character followed by the first character of the next word ($s \to c$). The entries in Table 4.1 show transition matrices A_{ij} for these three states with named entity tags as example.

The score of a sentence $[c]_1^n$ along a path of tags $[y]_1^n$ is computed by the product of transition scores and model output probabilities:

$$s([c]_1^n, [y]_1^n) = \prod_{i=1}^{n} A_{y_{i-1}y_i} o(c_i)_{y_i} \tag{4.1}$$

	PER	ORG	O	PER	ORG	O	PER	ORG	O
PER	1	0	0	1	0	1	1	0	0
ORG	0	1	0	0	1	1	0	1	0
O	0	0	1	0	0	1	1	1	1
	$c \rightarrow c$			$c \rightarrow s$			$s \rightarrow c$		

Table 4.1: Example transition matrices for phrase types (PER, ORG) and Outside (O). c denotes a character inside word boundaries and s denotes the delimiter space.

The path of tags with the highest sentence score is computed by:

$$[y^*]_1^n = \arg\max_{[y]_1^n} s([c]_1^n, [y]_1^n) \tag{4.2}$$

To find the path of tags with the highest sentence score I use the Viterbi algorithm (Viterbi, 1967). Once a character tag sequence is decoded without violating word boundaries, it is trivial to convert these tags to word-level tags.

Chapter 5

EXPERIMENTS

This chapter presents my experiments. I start by describing the properties of evaluation datasets for Named Entity Recognition, Part-of-speech tagging and Chunking. Then, I detail network training and impact of dropout. For each of the tasks, I demonstrate how changing the shape of the network effects performance. Moreover, I give a comparison of word-level and character-level views. Finally, I discuss the Viterbi decoding effect.

5.1 Datasets

In this section, I provide information about datasets used for my model evaluation. I present the number of sentences and unknown word percentages for each dataset.

5.1.1 Named Entity Recognition

In order to evaluate my system, I applied my model to NER datasets in various languages. The shared tasks of CoNLL-2002 and CoNLL-2003 provide NER data for four languages, Spanish, Dutch, English and German. In addition to these datasets, I evaluated my model on languages with rich morphologies: Arabic, Czech and Turkish. The Turkish NER dataset was prepared by (Tür et al., 2003). For Czech, Konkol and Konopík (2013) make the dataset prepared by (Ševčíková et al., 2007) CoNLL compatible. Benajiba et al. (2007) prepared ANERCorp for Arabic Named Entitiy Recognition. All of the datasets except Turkish and Arabic have conventional training, development and test splits. The Turkish dataset is provided with training and test sets. I separated a small fraction of the training set for development. Since the Arabic dataset (ANERCorp) does not have training and test splits, I applied Monte

Carlo cross validation (Arlot et al., 2010) with 3 random runs. Table 5.1 gives the number of sentences in each dataset. Additional infomation about CoNLL format is provided in Appendix A.1 and phrase types used in each dataset are provided in Appendix A.2.

	Arabic	Czech	Dutch	English	German	Spanish	Turkish
Train	3988	4644	15806	14041	12152	8323	30000
Dev.	-	572	2895	3250	2867	1915	2237
Test	797	577	5195	3453	3005	1517	3336
$\|T\|$	4	7	4	4	4	4	3

Table 5.1: Number of Sentences for Training, Development and Test sets and Number of Phrase Types ($|T|$).

	Arabic	Czech	Dutch	English	German	Spanish	Turkish
Unique	34.88	43.18	43.42	38.92	47.46	26.85	30.08
Phrase	18.82	35.54	39.28	31.27	42.20	18.39	13.82
Corpus	15.52	21.33	10.08	12.18	14.28	6.25	11.22

Table 5.2: Unknown word percentages for NER datasets.

Also, I show unknown word percentages for each dataset in Table 5.2. I conform to following when calculating the data sparsity of a dataset. I compute unique, phrase-wide and corpus-wide unknown word percentages where a word is considered unknown if test set contains it but the word is absent in training. The Unique row counts an unknown word only once while the Phrase and Corpus rows consider all the occurrences of an unknown word in phrases and the whole test set, respectively.

5.1.2 Part-of-speech tagging

For part-of-speech experiments, I used the Universal Dependency datasets which is a collection of treebanks across many languages annotated with a universal tagset (see Appendix A.3). For my experiments, I chose 4 languages (English, Finnish, German, Spanish).

	English	Finnish	German	Spanish	English (PTB)
Train	12543	12217	14118	14187	38219
Dev.	2002	716	799	1552	5527
Test	2077	648	977	274	5462
$\|T\|$	17	15	15	16	45

Table 5.3: Number of sentences and number of tags ($|T|$) for POS tagging datasets.

In addition, I use the Wall Street Journal (WSJ) portion of Penn Treebank (PTB) (Taylor et al., 2003) which is a canonical dataset for POS tagging (see Appendix A.4 for tag set). In order to compare with previous work, I adopt the splits — section 0-18 as training data, section 19-21 as development data and section 22–24 as test data. I summarize the number of sentences for training, development and test splits for all datasets in Table 5.3 and unknown word percentages in Table 5.4. Note that Finnish is an agglutinative language and has the highest unknown word percentages compared other datasets.

	English	Finnish	German	Spanish	English (PTB)
Unique	32.57	43.32	33.45	19.28	20.02
Corpus	9.11	24.43	11.85	7.28	3.07

Table 5.4: Unknown word percentages for POS tagging datasets.

5.1.3 Chunking

For Chunking experiments, I used the dataset that was introduced by (Tjong Kim Sang and Buchholz, 2000) in CoNLL-2000 shared task for English. The dataset contains approximately 9K and 2K sentences for training and test splits. I held out 1K sentences for development.

I summarize the number of sentences for training, development and test splits in Table 5.5 and unknown word percentages in Table 5.6. Appendix A.5 gives the details of chunk phrase types used in this dataset.

Training	Development	Test	$\|T\|$
7936	1000	2012	11

Table 5.5: Number of sentences and number of phrase types ($|T|$) for Chunking dataset.

Unique	Phrase	Corpus
31.78	8.41	7.32

Table 5.6: Unknown word percentages for Chunking dataset.

5.2 Network Training

In this section, I detail the common network training regime for all experiments. I use Adam (Kingma and Ba, 2014) for the gradient based training of the network and I find that the default parameters given in the original study work well. I update my network parameters after each mini-batch of 32 sentences. Before mini-batching, I sorted sentences according to character length to group sentences with similar lengths to speed up training. Also, I shuffle the order of mini-batches prior to each epoch. To stabilize the network training, I use gradient norm clipping to prevent gradients

from diverging (Pascanu et al., 2012). I set the maximum total norm of the gradients as 1. I tested several configurations of the network with different depths and widths and picked the configuration that work well for each task. Although one may obtain further improvements by tuning hyperparameters (optimization algorithm, norm of the gradients, etc.) of the network for each dataset in each task individually, I kept the hyperparameters same for each dataset to demonstrate that one can achieve adequate results with the same set of hyperparameters across tasks and languages.

5.3 Results

In this section, I evaluate my model and compare results with related work for NER, POS tagging and Chunking.

5.3.1 Named Entity Recognition

Here I present the performance of my model on languages with diverse characteristics on NER. Since my model only uses the labeled training data, and no external resources such as gazetteers, I selected previous works which report the top scores without use of any additional data to make a fair comparison. I also included the best results which do make use of arbitrary external resources for each language. I summarized all the comparisons in Table 5.7. I use phrase-level F1 score as evaluation metric.

The results highlight that my model attains good performance and it is robust across variety of languages on NER. I achieve similar to or better than the state-of-the-art in Named Entity Recognition that use no external resources. Abdul-Hamid and Darwish (2010) employ a CRF sequence labeling model which is trained on features that primarily use character n-gram of leading and trailing letters in words and word n-grams. They assert that the proposed features help overcome some of the morphological and orthographic complexities of Arabic. Although they do not utilize any external resource, they apply Arabic specific input preprocessing before training which may be the reason for better performance. Demir and Ozgur (2014) employs a window-based classifier approach with language independent features for

Czech and Turkish. My model outperform their results by a considerable margin for both languages. Gillick et al. (2015) reports the top scores with no external resources for Dutch, English, German and Spanish. They achieve higher F1 score for German and Spanish datasets, however, my model performs better for Dutch and is on par for English despite having less parameters.

	Arabic	Czech	Dutch	English	German	Spanish	Turkish
External	84.30 [1]	75.61 [2]	82.84 [3]	91.21 [4]	78.76 [5]	85.75 [5]	91.94 [6]
Labeled	81.00 [6]	68.38 [2]	78.08 [3]	84.57 [3]	72.08 [3]	81.83 [3]	89.73 [2]
CharNER	79.06	72.56	80.26	84.67	69.90	80.61	90.55

Table 5.7: Phrase-level F1 scores. The bottom row presents my model's results across seven languages. The middle row consists of models that report the top scores while using only labeled NER training data, comparable to my model. The top row presents state-of-the-art models that achieve the best results in each language utilizing external resources like Gazetteers. Works are numbered in the order of appearance: [1] (Darwish, 2013), [2] (Demir and Ozgur, 2014), [3] (Gillick et al., 2015), [4] (Ma and Hovy, 2016), [5] (Lample et al., 2016), [6] (Seker and Eryigit, 2012), [7] (Abdul-Hamid and Darwish, 2010)

Most state-of-the-art NER models are obtained by handcrafting good word-level features and generally utilizing external information sources. Models usually resort to additional training resources: (1) when training and test set are not from the same data generating distribution (English) or (2) training set is small (Arabic and Czech). Ma and Hovy (2016) report the best published F1 score (91.21%) for CoNLL-2003 English dataset. Without pretrained word embeddings however, their model loses approximately 10% F1 score (absolute). Lample et al. (2016) also relies on unsupervised word representations extracted from unannotated corpora. They announce the best results for German and Spanish datasets. On the other hand, Demir and Ozgur (2014) utilize pretrained word embeddings along with their corresponding word cluster ids

to achieve top performance on Czech dataset. Darwish (2013) outperforms (Abdul-Hamid and Darwish, 2010) by 3.4% F1 score by exploiting cross-lingual features and knowledge bases from English data sources. Moreover, language specific expertise can be incorporated to improve the performance. The current state-of-the-art system for Turkish (Seker and Eryigit, 2012) achieves 91.94% F1 score by using language dependent features along with gazetteers. Finally, Gillick et al. (2015) achieves the best result for Dutch by using concatenated Dutch, English, German, Spanish training sets as one. Even though I do not use any additional training source, the performance of my model is competitive for Czech, Dutch and Turkish.

5.3.2 Part-of-Speech Tagging

In this section, I present my model's performance on Part-of-Speech Tagging datasets. First, I make a comparison with related work on Universal Dependency datasets. Then, I give results and compare with previous work on WSJ portion of Penn Treebank dataset. I report per word accuracy as evaluation score.

I compared my model to Byte-to-Span (BTS) (Gillick et al., 2015) and to a CRF trained with an extensive collection of features on Universal Dependency Datasets. Note that all of the models are trained with no external information resources. The results are presented in Table 5.8 and scores of BTS and CRF are taken from (Gillick et al., 2015). Both models outperform CRF by a large margin on Finnish dataset which indicates that character based models are powerful at generalizing to agglutinative languages. Lastly, CharPOS outperforms BTS and CRF on all datasets.

	English	Finnish	German	Spanish
CharPOS	94.89	96.61	93.79	96.20
CRF	94.51	92.82	91.99	95.03
BTS	94.00	96.05	92.34	95.26

Table 5.8: POS tagging results on Universal Dependency Datasets.

External	Ling et al. (2015a)	97.78
	Ma and Hovy (2016)	97.55
	dos Santos and Zadrozny (2014)	97.32
	Collobert et al. (2011)	97.29
Labeled	Ling et al. (2015a)	97.36
	Ma and Hovy (2016)	97.13
	Collobert et al. (2011)	96.37
	Collins (2002)	97.11
	Ratnaparkhi et al. (1996)	96.60
	CharPOS	97.34

Table 5.9: POS tagging results on English Penn Treebank

Moreover, I compare my model's performance on WSJ portion of Penn Treebank with previous work in Table 5.9. One of the earliest works (Ratnaparkhi et al., 1996) uses MEMM with handcrafted features to build a Part-of-Speech tagger and achieves 96.60% accuracy. Then, Collins (2002) utilizes the same set of features with (Ratnaparkhi et al., 1996) and improves on to 97.11% with Structured Perceptron. Recent works (Collobert et al., 2011; dos Santos and Zadrozny, 2014; Ling et al., 2015a; Ma and Hovy, 2016) utilizes neural networks to build taggers. (Collobert et al., 2011) employs Convolution Neural Networks (CNNs) to model contexts within a window of fixed size, and uses CRF to output tags. By utilizing handcrafted features and external sources, they achieve 97.29% accuracy. dos Santos and Zadrozny (2014) utilizes the same architecture with (Collobert et al., 2011) and in addition, they learn character embeddings with CNNs instead of crafting features. They report results comparable to (Collobert et al., 2011). Ling et al. (2015a) achieves the top accuracy 97.78% utilizing word representations obtained by bidirectional LSTM over characters along with pretrained word embeddings. Ma and Hovy (2016) uses a combination of bidirectional LSTM, CNN and CRF to benefit both from word- and

character-level representations and achieves 97.55% accuracy. However, all of the neural network based models (Ma and Hovy, 2016; Collobert et al., 2011; Ling et al., 2015a) experiences a performance drop without external data. (I can not include (dos Santos and Zadrozny, 2014) since they do not report results without using external data.) Lastly, CharPOS is on par with state-of-the-art models with 97.14% accuracy when only labeled data is utilized in training.

5.3.3 Chunking

In this section, I present my model results on CoNLL-2000 Chunking dataset and compare it with related work. I use phrase-level F1 score as evaluation metric.

Huang et al. (2015)	94.46
Collobert et al. (2011)	94.32
Sun et al. (2008)	94.34
McDonald et al. (2005)	94.29
Shen and Sarkar (2005)	95.23
Sha and Pereira (2003)	94.30
Kudoh and Matsumoto (2000)	93.48
CharChunk	93.59

Table 5.10: Chunking results on English CoNLL-2000 dataset

All chunking system performances are summarized in Table 5.10. Kudoh and Matsumoto (2000) won the CoNLL-2000 challenge on chunking with a F1-score of 93.48% using a SVM based approach. CRF based models (Sha and Pereira, 2003; McDonald et al., 2005; Sun et al., 2008) reports F1 score of around 94.30%. Shen and Sarkar (2005) obtained best performance with 95.23% F1 score by using a voting classifier scheme, where each classifier is trained on different tagging schemes (IOB,IOE, etc.) These systems use features composed of POS tags, words and tags. Neural architectures (Collobert et al., 2011; Huang et al., 2015) uses pretrained word embeddings.

Collobert et al. (2011) which uses CNNs with CRF achieves a score comparable to CRF based solutions. Huang et al. (2015) uses bidirectional LSTM with CRF and reports better score (94.46%) than (Collobert et al., 2011). The performance of my model fall behind Neural Network/CRF based solutions. One reason of this shortcoming might be that all of the systems report results by utilizing some type of external resources.

5.4 Ablation Studies

This section demonstrates the ablation studies for dropout usage, network shapes and utilizing a decoder.

5.4.1 Dropout

In this section, I explain the dropout usage and conduct ablation studies for dropout rates selected for each layer.

I use dropout (Srivastava et al., 2014) to prevent the network overfitting training data quickly. Dropout randomly (according to hyperparameter *dropout rate*) drops units from a neural network layer during training which prevents units from co-adapting too much. I applied dropout to outputs of each layer including the input layer to improve generalization power.

$\boldsymbol{d_i}$	d_m	d_o		d_i	$\boldsymbol{d_m}$	d_o		d_i	d_m	$\boldsymbol{d_o}$	
.5	.5	.8	72.52	.2	.8	.8	74.86	.2	.5	.9	76.53
.2	.5	.8	76.71	.2	.5	.8	76.71	.2	.5	.8	76.71
0	.5	.8	73.41	.2	.2	.8	76.62	.2	.5	.5	74.67

Table 5.11: Ablation study of dropout rates

In Table 5.11, I show the dropout rates and their effect on the network performance on NER Czech dataset. In my experiments, I used a 5 layer network where each layer

has 128 hidden units. The triplet (d_i d_m d_o) stands for the dropout rates applied to the outputs of input, middle layers and the final output layer respectively. If there are more than one middle layer, same dropout rate is applied to all of them. I altered one of the dropout rates while keeping others constant.

When d_i is greater than 0, dropout at the input layer turns off the 1 value in input character vector (one-hot representation) at random positions in the sequence. Applying dropout to inputs improves performance while a high dropout rate (.5) causes the network to underfit. Moreover, increasing dropout rate for middle layers hurts the performance. In addition, the network favors higher dropout rates at the final layer. Henceforth, I keep the (.2 .5 .8) setting for all my experiments.

5.4.2 Network Shape

In this section, I scrutinize the different network configurations by varying the number of layers the network has (depth) and number of hidden units in each layer (width).

I conducted experiments for each task independently until I find a configuration where locally deviating does not yield an improvement in a single parameter. I start my search from 4x128 network which denotes 4 layers deep and each layer has 128 hidden units. I increased the network depth up to 6 layers and doubled hidden layer size at each step. I used Czech dataset for NER, Universal Dependencies English dataset for Part-of-Speech and English dataset for Chunking. I summarized the results

	Hidden Size		
Depth	128	256	512
3	x	76.26	x
4	76.51	**78.03**	74.76
5	76.71	76.61	x
6	76.09	75.53	x

Table 5.12: Network shape ablation for NER.

in Table 5.12, Table 5.13, Table 5.14.

	Hidden Size		
Depth	128	256	512
3	x	x	x
4	95.09	95.13	x
5	95.06	**95.35**	95.24
6	95.15	95.14	x

Table 5.13: Network shape ablation for POS tagging.

For Named Entity Recognition, 5 layer deep network achieved the top performance when hidden unit size was 128, however 4x256 network yielded the best result overall. For both Part-of-Speech and Chunking, 5 layer deep network with 256 hidden units obtained the top result.

	Hidden Size		
Depth	128	256	512
3	x	x	x
4	94.47	94.41	x
5	94.49	**94.55**	94.43
6	94.35	94.54	x

Table 5.14: Network shape ablation for Chunking.

5.4.3 Need for a Decoder

As discussed in Section 4.3, the deep BLSTM does not output probabilities that always favor a single tag within a word. Character level tagging is a structured learning problem, i.e. there are dependencies between the outputs that are difficult to capture by the deep BLSTM which only has access to the input sequence. To quantify the

effect of capturing these dependencies, I ran a simpler baseline model for comparison. I did not apply Viterbi decoding to the output probability distributions of the network and picked the most probable tag for each character. Then, I used majority voting between characters of the same word to assign single tag to a word. With this scheme, I observed 1 F1 (absolute) performance drop on NER Czech dataset.

5.5 Word-level vs Character-level

In this section, I give a comparison of word-level and character-level models where the input representation is altered accordingly.

		Arabic	Czech	Dutch	English	German	Spanish	Turkish
	X	136	136	101	85	96	92	100
Char	μ_l	149	142	70	76	105	173	90
	X	27050	34869	27803	23623	32932	26099	63703
Word	μ_l	27	25	12	14	17	31	13

Table 5.15: Comparison of word-level and character-level views. X denotes the number of unique input tokens and μ_l denotes the average sequence length.

I use Named Entity Recognition datasets for demonstration but the same trend complies in other datasets as well. In Table 5.15, the number of unique tokens and average sequence length are indicated for each language. Switching from word-level to character-level view reduces the number of unique tokens drastically while making the input sequences longer.

Moreover, I compare both word-level and character-level model performance on Czech dataset. For comparison, I built a word-level model where I used words as the input instead of characters. Note that, I did not replace infrequent words with a special token ($< unk >$). Since network volume need of word-level model could differ from character-level one, I experimented with different network architectures until I find the network shape with a local maximum score. I present the results in Table

	Hidden Size		
Depth	256	512	1024
1	55.98	56.56	57.36
2	57.66	57.15	57.56
3	57.87	**58.38**	58.07
4	x	55.44	x

Word-level (3x512)	58.38
Char-level (4x256)	78.03

Table 5.16: Word-level network shape experiments.

5.16 along with the F1 scores of best models trained on Czech data with different input representations.

Character-level model outperforms word-level counterpart by a large margin. I observed that character-level model favors deeper and narrower network compared to word-level model. Also, character-level model require less parameters compared to word-level model hence is able to learn with less data — Czech dataset is modest in size and has high sparsity — and more robust to data sparsity problem.

Chapter 6

CONCLUSION

In this thesis, I described a character-level tagger employing a deep bidirectional LSTM architecture and evaluated it on three tagging problems, namely, Named Entity Recognition, Part-of-Speech and Chunking. I showed that taking characters as the primary representation is superior to considering words as the basic input unit. My main contribution is to show that the same deep character level model trained for a specific task is able to achieve good performance on multiple languages without hand engineered features or language specific external resources.

There are several potential directions for future work. First, incorporating semi-supervised learning can further improve the results. Moreover, one can take advantage of multi-task learning approaches such that the model is jointly trained with tags from multiple tasks (NER, Chunking, ...). Furthermore, utilizing transfer learning between tasks or languages might bring performance boost.

Appendices

Appendix 1

DATASETS

A.1 CoNLL Format: IOB, BIO Encoding Schemes

Here, I provide the details of phrase encoding schemes for datasets in CoNLL format. CoNLL uses two types of encoding schemes to encode phrase types to word-level tags, namely IOB and BIO.

IOB: Words tagged with O belong to no phrase and the I-XXX tag is used for words inside a phrase of type XXX. Whenever two phrases of type XXX are immediately next to each other, the first word of the second phrase will be tagged B-XXX in order to show that it starts another phrase.

BIO: The O tag is used for tokens which are not part of any phrase. Contrary to IOB, B-XXX tag is used for the first word of every phrase of type XXX and I-XXX is used for each other word in the phrase as necessary.

Chunking	NER						
English	Arabic	Czech	Dutch	English	German	Spanish	Turkish
BIO	BIO	BIO	BIO	IOB	IOB	BIO	-

Table A.1: Encoding schemes for Chunking and NER datasets

Table A.1 shows the phrase encoding schemes for each dataset. Note that Turkish dataset is not provided in CoNLL format, thus I converted it with BIO. Moreover, I converted English and German NER datasets to BIO encoding as well.

A.2 Named Entity Recognition Phrase Types

In this section, I give the details of phrase types used in Named Entity Recognition datasets. I summarized all the phrase types in Table A.2. PER, LOC, ORG, MISC are shorthands for Person, Location, Organization and Miscellaneous.

Arabic	Czech	Dutch	English	German	Spanish	Turkish
PER	PER	PER	PER	PER	PER	PER
LOC	LOC	LOC	LOC	LOC	LOC	LOC
ORG	ORG	ORG	ORG	ORG	ORG	ORG
MISC	Media	MISC	MISC	MISC	MISC	
	Address					
	Time					
	Other					

Table A.2: NER phrase types for each dataset.

A.3 Universal Dependency POS tags

Here, I describe the Part-of-Speech tags used in Universal Dependency (UD) datasets. UD datasets provide a universal inventory of categories to make consistent annotation across languages easier, while allowing language-specific extensions when necessary. Table A.3 demonstrates the set of all POS tags used in UD datasets.

ADJ	Adjective
ADP	Adposition
ADV	Adverb
AUX	Auxiliary verb
CONJ	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 A.3: Universal Dependency Part-of-Speech Tags

A.4 Penn Treebank POS tags

Here, I describe the Part-of-Speech tags used in Wall Street Journal (WSJ) portion of Penn Treebank. Table A.4 shows 36 POS tags for words. In addition, there are 9 more tags for punctuation marks (dot, comma, etc.) with the same label.

CC: Coordinating conjunction	CD: Cardinal number
DT: Determiner	EX: Existential there
FW: Foreign word	IN: Preposition
JJ: Adjective	JJR: Adjective, comparative
JJS: Adjective, superlative	LS: List item marker
MD: Modal	NN: Noun, singular or mass
NNS: Noun, plural	NNP: Proper noun, singular
NNPS: Proper noun, plural	PDT: Predeterminer
POS: Possessive ending	PRP: Personal pronoun
PRP$: Possessive pronoun	RB: Adverb
RBR: Adverb, comparative	RBS: Adverb, superlative
RP: Particle	SYM: Symbol
TO: to	UH: Interjection
VB: Verb, base form	VBD: Verb, past tense
VBG: Verb, gerund or present participle	VBN: Verb, past participle
VBP: Verb, non-3rd person singular present	VBZ: Verb, 3rd person singular present
WDT: Wh-determiner	WP: Wh-pronoun
WP$: Possessive wh-pronoun	WRB: Wh-adverb

Table A.4: Penn Treebank Part-of-Speech Tags

A.5 Chunking Phrase Types

This section summarizes the phrase types for CoNLL English Chunking dataset in Table A.5.

NP	Noun phrase
VP	Verb phrase
ADVP	Adverb phrase
ADJP	Adjective phrase
PP	Prepositonal phrase
SBAR	Complementizer phrase
CONJP	Conjunction phrase
PRT	Particle phrase
INTJ	Interjection phrase
LST	List marker phrase
UCP	Unlike coordinated phrase

Table A.5: Chunking Phrase Types

BIBLIOGRAPHY

Abdul-Hamid, A. and Darwish, K. (2010). Simplified feature set for arabic named entity recognition. In *Proceedings of the 2010 Named Entities Workshop*, pages 110–115. Association for Computational Linguistics.

Ando, R. K. and Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

Arlot, S., Celisse, A., et al. (2010). A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79.

Ballesteros, M., Dyer, C., and Smith, N. A. (2015). Improved transition-based parsing by modeling characters instead of words with lstms. *arXiv preprint arXiv:1508.00657*.

Benajiba, Y., Rosso, P., and Benedíruiz, J. M. (2007). Anersys: An arabic named entity recognition system based on maximum entropy. In *Computational Linguistics and Intelligent Text Processing*, pages 143–153. Springer.

Bengio, I. G. Y. and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Carreras, X., Marquez, L., and Padró, L. (2002). Named entity extraction using adaboost. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4. Association for Computational Linguistics.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Darwish, K. (2013). Named entity recognition using cross-lingual resources: Arabic as an example. In *ACL (1)*, pages 1558–1567.

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

dos Santos, C., Guimaraes, V., Niterói, R., and de Janeiro, R. (2015). Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

dos Santos, C. N. and Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

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

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

Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural*

language learning at HLT-NAACL 2003-Volume 4, pages 168–171. Association for Computational Linguistics.

Gers, F., Schmidhuber, J., et al. (2000). Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, pages 189–194. IEEE.

Gillick, D., Brunk, C., Vinyals, O., and Subramanya, A. (2015). Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

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

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2015). Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klein, D., Smarr, J., Nguyen, H., and Manning, C. D. (2003). Named entity recognition with character-level models. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 180–183. Association for Computational Linguistics.

Konkol, M. and Konopík, M. (2013). Crf-based czech named entity recognizer and consolidation of czech ner research. In *Text, Speech, and Dialogue*, pages 153–160. Springer.

Kudoh, T. and Matsumoto, Y. (2000). Use of support vector learning for chunk identification. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 142–144. Association for Computational Linguistics.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. (2015a). Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015b). Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

McCallum, A., Freitag, D., and Pereira, F. C. (2000). Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598.

McDonald, R., Crammer, K., and Pereira, F. (2005). Flexible text segmentation with structured multilabel classification. In *Proceedings of the conference on Human*

Language Technology and Empirical Methods in Natural Language Processing, pages 987–994. Association for Computational Linguistics.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Ratnaparkhi, A. et al. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, USA.

Seker, G. A. and Eryigit, G. (2012). Initial explorations on using crfs for turkish named entity recognition. In *COLING*, pages 2459–2474.

Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

Shen, H. and Sarkar, A. (2005). Voting between multiple data representations for text chunking. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 389–400. Springer.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.

(2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sun, X., Morency, L.-P., Okanohara, D., and Tsujii, J. (2008). Modeling latent-dynamic in shallow parsing: a latent conditional model with improved inference. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 841–848. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Suzuki, J. and Isozaki, H. (2008). Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*, pages 665–673.

Taylor, A., Marcus, M., and Santorini, B. (2003). The penn treebank: an overview. In *Treebanks*, pages 5–22. Springer.

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Tür, G., Hakkani-Tür, D., and Oflazer, K. (2003). A statistical information extraction system for turkish. *Natural Language Engineering*, 9(02):181–210.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Wang, P., Qian, Y., Soong, F. K., He, L., and Zhao, H. (2015). A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215*.

Zhang, T. and Johnson, D. (2003). A robust risk minimization based named entity recognition system. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 204–207. Association for Computational Linguistics.

Zhang, X. and LeCun, Y. (2015). Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Ševčíková, M., Žabokrtský, Z., and Krůza, O. (2007). Named entities in czech: annotating data and developing ne tagger. In *Text, Speech and Dialogue*, pages 188–195. Springer.