Speed versus Accuracy in Neural Sequence Tagging for Natural Language Processing

by

Xinxin Kou

B.Sc. (Hons.), Dalhousie University, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2017

Approval

Name:	**Xinxin Kou**
Degree:	**Master of Science (Computing Science)**
Title:	**Speed versus Accuracy in Neural Sequence Tagging for Natural Language Processing**
Examining Committee:	**Chair:** Dr. Arrvindh Shriraman Professor
	Dr. Anoop Sarkar Senior Supervisor Professor
	Dr. Fred Popowich Supervisor Professor
	Dr. Jiannan Wang Examiner Assistant Professor
Date Defended:	**12 September 2017**

Abstract

Sequence Tagging, including part of speech tagging, chunking and named entity recognition, is an important task in NLP. Recurrent neural network models such as Bidirectional LSTMs have produced impressive results on sequence tagging. In this work, we first present a Bidirectional LSTM neural network model for sequence tagging tasks. Then we show a simple and fast greedy sequence tagging system using a feedforward neural network. We compare the speed and accuracy between the Bidirectional LSTM model and the greedy feedforward model. In addition, we propose two new models based on Mention2Vec by Stratos (2016): Feedforward-Mention2Vec for named entity recognition and chunking, and BPE-Mention2Vec for part-of-speech tagging. Feedforward-Mention2Vec predicts tag boundaries and corresponding types separately. BPE-Mention2Vec uses the Byte Pair Encoding algorithm to segment words first and then predicts the part-of-speech tags for the subword spans. We carefully design the experiments to demonstrate the speed and accuracy trade-off in different models. The empirical results reveal that the greedy feedforward model can achieve comparable accuracy and faster speed than recurrent models for sequence tagging, and Feedforward-Mention2Vec is competitive with the fully structured BiLSTM model for named entity recognition while being more scalable in the number of named entity types.

Keywords: Natural Language Processing; Sequence Tagging; Neural Networks

Dedication

To my beloved families who always support me and encourage me.

Acknowledgements

I would like to express my profound sense of gratitude to my supervisor Dr. Anoop Sarkar for introducing me to this research topic and providing his continuous support and valuable guidance throughout my graduate study. I can not imagine having a better advisor and mentor. In addition, I would like to express my sincere appreciation to Dr. Fred Popowich for his useful advice and feedback on this work, and Dr. Jiannan Wang for being my defence examiner and reading my thesis.

Thanks to all of my Natural Language Processing lab mates who helped me. I enjoy spending time with them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we first describe sequence tagging tasks and introduce the motivation of this thesis. Then, we summarize our major contributions and describe the structure of the thesis.

1.1 Sequence Tagging Task

1.1.1 Part-of-Speech Tagging (POS)

Part-of-Speech Tagging (POS) is a basic sequence tagging task, which assigns each word with a unique tag that indicates its syntactic role, such as noun, adverb, verb, etc. Figure 1.1 illustrates a typical POS task.

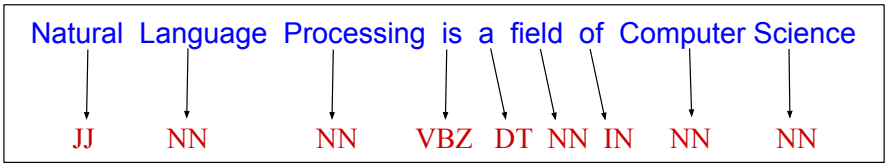


Figure 1.1: An example of POS

Most POS systems are evaluated on the English Penn TreeBank data set (Marcus et al., 1993), which contains 45 part-of-speech tags. The standard split uses section 1-18 of the Treebank for training, section 19-21 for tuning, and section 22-24 for testing (Toutanova et al., 2003). The experimental data are summarized in Table 1.1. Many existing models are linear models, such as Maximum entropy Markov models (MEMMs) which obtain 96.46% per word accuracy (McCallum et al., 2000); and the averaged perception discriminative model which obtains 97.11% per word accuracy (Collins, 2002). More recently, neural network models have been proposed to improve the state-of-the-art score. The Bidirectional LSTM network model proposed by Huang et al. (2015) reaches 97.55% per word accuracy.

Ling et al. (2015) presents the compositional character-to-word LSTM model which reaches the state-of-the-art performance: 97.78% per word accuracy.

Table 1.1: Number of Words and Labels in Each Training, Validation and Test Section of Different Data Sets

	Training	Validation	Test	labels
Penn Treebank	950011	40068	56671	45
CoNLL 2000	211727	N/A	47377	22
CoNLL 2003	204567	51578	46666	9
OntoNotes	1088503	147724	152728	18

1.1.2 Named Entity Recognition (NER)

Named Entity Recognition (NER) identifies expressions that refer to named entities, such as people, places, organizations and others. The main difference between NER and POS is that each named entity label can span multiple words while each part-of-speech tag is only for one word. A popular convention in NER is to use the "IOB" label scheme (Inside, Outside, Beginning): if the word is the beginning of a named entity label, it is marked with B-label; if the word is inside a named entity label but not the first one, it is marked with I-label; if the token is outside the named entity, it is marked with "O". An example of NER is shown in Figure 1.2. There are different numbers of named entity types in different NER tasks. For example, the shared task of CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) contains 4 types of named entities: locations (LOC), persons (PER), organizations (ORG), and miscellaneous (MISC); and the OntoNotes English data set (Hovy et al., 2006) contains 18 types of named entities shown in Table 1.2. The predefined training, development, and testing split of the data sets are shown in Table 1.1.

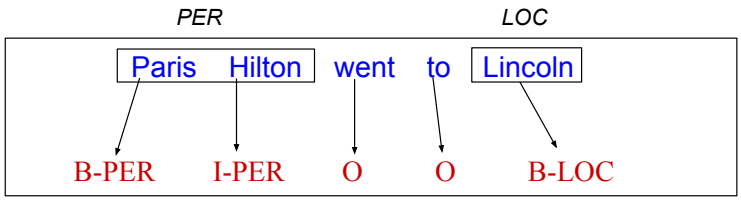


Figure 1.2: An example of NER

Most NER models are evaluated on CoNLL 2003 data set and are measured by the F1 score, which is the harmonic mean of precision and recall.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{1.1}$$

Table 1.2: Name Entity Types in OntoNotes

Types	Named Entities
PERSON	People, including fictional
NORP	Nationalities or religious or political groups
FACILITY	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions
GPE	Countries, cities, states
LOC	Non-GPE locations, mountain ranges, bodies of water
PRODUCT	Vehicles, weapons, foods, etc. (Not services)
EVENT	Named hurricanes, battles, wars, sports events
WORK OF ART	Titles of books, songs
LAW	Named documents made into laws
LANGUAGE	Any named language
DATE	Absolute or relative dates or period
TIME	Times smaller than a day
PERCENT	Percentage
MONEY	Monetary values, including unit
QUANTITY	Measurements, as of weight or distance
ORDINAL	First, Second, etc.
CARDINAL	Numerals that do not fall under another type

Since CoNLL 2003 has a relatively smaller amount of training data, most of the existing models make use of pretrained word embeddings along with the training data. A commonly used pretrained word embedding is GloVe (Pennington et al., 2014) which contains 40K words. The best system presented at the NER CoNLL 2003 challenge by Florian et al. (2003) obtains 88.76 F1 score. The model using Bidirectional LSTM by Huang et al. (2015) reaches 88.83 F1 score. Both of these two models use many external features along with a gazetteer[1]. Lample et al. (2016) proposed two NER models with no external features or a gazetteer: the first one makes structured prediction using Bidirectional LSTM, Character Embeddings (Ling et al., 2015) and Conditional Random Fields (CRF) (Lafferty et al., 2001); and the second one uses a Shift-Reduce framework with Stack-LSTM (Dyer et al., 2015). The first model achieves 90.94 F1 score, while the second one performs slightly worse and achieves 90.33 F1 score.

1.1.3 Chunking

Chunking is also known as shallow parsing which labels segments of a sentence with syntax tags, such as NP, VP, PP, etc. An example of chunking is shown in Figure 1.3. The "IOB" label scheme is used in chunking. Most Chunking systems are trained and evaluated on the CoNLL 2000 data set (Tjong Kim Sang and Buchholz, 2000). Same as CoNLL 2003,

[1]Gazetteers are language-specific knowledge resources

the number of training data in CoNLL 2000 is limited, and a pretrained word embedding is commonly used in training Chunking models. The best system using SVMs presented at the CoNLL 2000 challenge by Kudoh and Matsumoto (2000) obtains 93.48 F1 score. The current state-of-the-art score is 95.23 which is achieved by Shen and Sarkar (2005). They use part-of-speech tags as features and a voting classifier for chunking. Indig and István (2016) replicated their model and got 95.08 F1 score. Recently, Huang et al. (2015) employs Bidirectional LSTM to solve chunking and achieved 94.46 F1 score with a number of hand-crafted features.

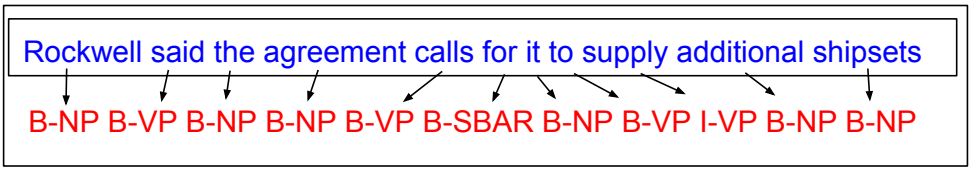


Figure 1.3: An example of Chunking

1.2 Motivation

Recurrent Neural Networks (RNNs) have obtained impressive results in many NLP tasks, such as speech recognition (Graves et al., 2013) and machine translation (Cho et al., 2014). Bidirectional Long Short Term Memory (BiLSTM) (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) is one of the RNN architectures that can maintain long-distance information from the past and future elements in an input sequence. Based on the existing work (POS system by Ling et al. (2015) and NER system by Lample et al. (2016)), state-of-the-art results on POS and NER can be obtained by using a BiLSTM network with Character Embeddings and Conditional Random Fields (CRF). Some work has also shown that feedforward neural networks can achieve comparable accuracy to recurrent models in tasks such as POS and Dependency Parsing (Andor et al., 2016). One approach presented in this thesis is to employ a greedy transition system with a feedforward network to make independent classification decisions on each word. While the state-of-the-art model focuses on sentence level and computes the score of every possible output sequence during decoding, the greedy system makes decisions on word level and can produce output sequence faster. However, the greedy system is limited when there are strong correlations between output labels. NER is one such task which has grammar constraints on output label sequences. For

example, “I-PER” cannot follow “B-LOC” in NER. We want to build a fast neural network model that can capture the dependencies among output tags.

Since the labels in Chunking and NER often span multiple tokens, most neural architectures for Chunking and NER predict the boundaries and types of entities together using the “IOB” label scheme. Mention2Vec (Stratos, 2016) is proposed to address the natural segment-level representation by separating the NER task into boundary detection (I, O, B) and type prediction (PER, LOC, etc.). While Mention2Vec employs two BiLSTMs for each sub-task, we replace the BiLSTM layer for boundary detection with a feedforward network in order to obtain a simpler and faster model. This new model is denoted as Feedforward-Mention2Vec in this thesis.

We use Byte Pair Encoding (BPE) (Gage, 1994) to deal with rare words in sentences for machine translation (Sennrich et al., 2016), and we propose a new model combining BPE and Mention2Vec for POS, which is denoted as BPE-Mention2Vec in this thesis. We use BPE to segment the input words in the hope of capturing the orthographic evidence of the words without using spelling features (like prefixes and suffixes) or Character Embeddings. After we segment input words, POS becomes an NER-like task. Then, we can use Mention2Vec to solve the rest of the problem. Since boundaries of output tags are known in POS, we only need to use a BiLSTM network for type prediction in BPE-Mention2Vec.

There is increasing demand for faster sequence tagging systems to decode real time natural language data, such as Twitter, Facebook, Wikipedia, and even the web. Improving decoding speed will benefit many real world sequence tagging applications, but there is very little research into examining the decoding speed and accuracy relationship of different models. In this thesis, we will explore the speed and accuracy trade-off in different neural network models for sequence tagging. We re-implement many different types of feedforward models and BiLSTM models using Tensorflow (Abadi et al., 2016) and systematically compare the performance and decoding speed between them on sequence tagging tasks.

1.3 Contribution

The three main contributions of this thesis are:

1. We implement the state-of-the-art sequence tagging model:BiLSTM with Character Embeddings and CRF, and we apply the fully structured BiLSTM model on all three sequence tagging tasks.

   There is little work examining how the configurations of the fully structured BiLSTM model affect the decoding speed, such as whether the model is using Character Embeddings or the model is using CRF. We conduct experiments to compare the performance and decoding time of different configurations. Our implementation can be found in https://github.com/sfu-natlang/neural-network-tagger

2. We build a greedy tagging system with a feedforward neural network, denoted as Feedforward-History.

   Feedforward-History takes word features in context, spelling features, and previous tag features as input, and uses a feedforward neural network to predict tags word by word. There is little existing work measuring the decoding time using feedforward networks on sequence tagging, so we conduct experiments on sequence tagging and record the performance and decoding speed. To test the robustness of the feedforward networks, we also conduct experiments using a feedforward network with only word features, which also serves as the baseline. We compare the feedforward model with the fully structured BiLSTM model and provide analysis on the results.

3. Last but not least, we introduce two new neural architectures based on Mention2Vec: Feedforward-Mention2Vec for Chunking and NER, and BPE-Mention2Vec for POS.

   Originally, Mention2Vec was designed for NER and used BiLSTMs to detect named entity boundaries and predict corresponding types separately. We propose Feedforward-Mention2Vec for Chunking and NER, in which we use a feedforward network with CRF to predict named entity boundaries instead of using a BiLSTM. We denote this new model as Feedforward-Mention2Vec. We also adapt Mention2Vec for POS by combining Byte Pair Encoding (BPE) with BiLSTM in the model. BPE is used to segment the input words in our model, and it converts POS into a NER-like task. We denote this new model for POS as BPE-Mention2Vec. In BPE-Mention2Vec, we use a feedforward network to compute the hidden embeddings of the input segmented words. Since the boundaries of subword units are known, BPE-Mention2Vec does not need to predict the boundaries. It takes the hidden embeddings, tag boundaries, and a BiLSTM network to predict the actual POS tags. We benchmark these two multitask models against the state-of-the-art BiLSTM model on POS and NER. Since different NER tasks have different numbers of named entity types, the decoding time of a fully structured BiLSTM model grows quadratically in the number of types. In Mention2Vec and Feedforward-Mention2Vec, we only apply CRF on boundary labels (I, O, B), so the decoding time grows linearly in the number of types. To show the time difference on different NER tasks, we conduct the NER experiments on CoNLL 2003 which contains 4 different named entity types and OntoNotes which contains 18 different named entity types.

1.4 Overview

The thesis is organized as follows:

In **Chapter 2** we present the fully structured BiLSTM model, which combines BiLSTM, Character Embeddings and CRF. We denote the fully structured model as the

BiLSTM-CRF model. We explain the training and decoding process of BiLSTM-CRF, apply BiLSTM-CRF on sequence tagging tasks, and conduct experiments using three different configurations: BiLSTM with word features only, BiLSTM with Character Embeddings, and BiLSTM model with Character Embeddings and CRF.

In **Chapter 3** we present a greedy tagging system using a feedforward neural network. Since the greed system takes into account the previous tag information and use a feedforward network to predict tags, we denote this model as Feedforward-History. In order to show the robustness of this model, we also build a greedy feedforward model with only word features (denoted as Feedforward). We explain the training and decoding process of the feedforward models, describe the experiment design, and compare the performance and decoding time between feedforward models and BiLSTM models.

In **Chapter 4** we present two new multitask models based on Mention2Vec: Feedforward-Mention2Vec for Chunking and NER and BPE-Mention2Vec for POS. The intuition behind the multitask models is to capture the boundary information of tags and to reduce the decoding time. We explain the architectures of these two models and compare the performance and speed between multitask models and other models in this thesis.

In **Chapter 5** we discuss the empirical results from the previous chapters. We also analyze the trade-off between performance and speed in different neural network models.

In **Chapter 6** we summarize the contributions of this thesis and discuss ongoing and related future work.

Chapter 2

Bidirectional Long Short Term Memory Network Models

In this chapter, we first describe the fully structured BiLSTM model: the state-of-the-art model combining BiLSTM with Character Embeddings and Conditional Random Fields (CRF). Then, we show the performance and decoding speed of BiLSTM models with different configurations on POS and NER.

2.1 Model Description

2.1.1 Bidirectional Bidirectional Long Short Term Memory

Recurrent neural networks (RNNs) (Mikolov et al., 2010) have shown impressive results on sequence tagging tasks. RNNs can keep the future and the past data persistent by using memory cells with loops in them. However, RNNs are biased towards the most recent data in practice. Long Short Term Memory Networks (LSTMs), special RNNs with Long Short Term memory cells (Graves and Schmidhuber, 2005), are designed to combat the bias problem. LSTMs learn long dependencies in a sequence with the help of Gates (Graves and Schmidhuber, 2005). Gates control how much of the input information to pass to the next LSTM cell, and how much of the previous information to forget.

Given a sentence with n words each of which is represented as a dense vector x_i, a LSTM makes use of $x_{1:n}$ to compute a forward representation $\overrightarrow{h_i}$ for the ith word. In general, computing a backward representation $\overleftarrow{h_i}$ would be useful for sequence tagging. Bidirectional LSTM (BiLSTM) is an extension to LSTM which takes into account both the past elements and the future elements in a sequence. A BiLSTM generates both $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ for the ith word in a sequence. The hidden embedding h_i is the concatenation of $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$. BiLSTM mapping is defined as $BiLSTM_\theta$: $h_i = BiLSTM_\theta(x_{1:n}, i)$ where θ represents trainable parameters in BiLSTM. Figure 2.1 illustrates the application of the BiLSTM model on an NER example.

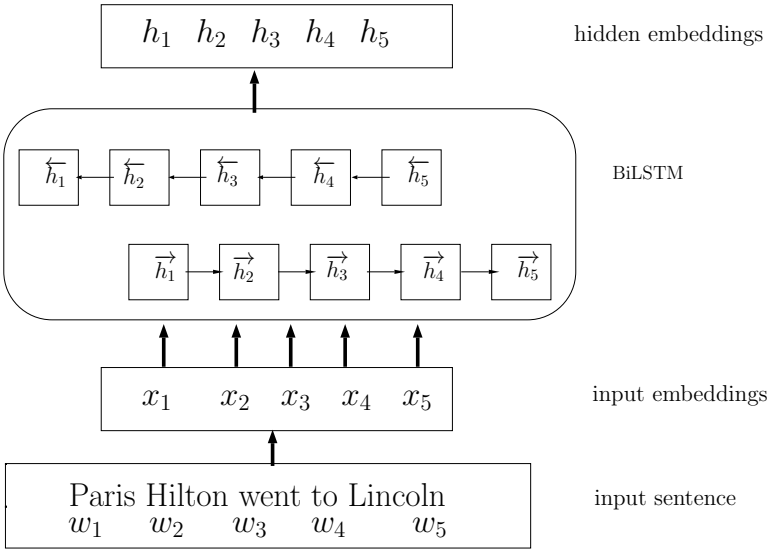


Figure 2.1: The architecture of BiLSTM on an NER example

2.1.2 Character Embeddings

Instead of using hand-engineered features listed in Chapter 2 (like prefixes and the suffixes of a word), we can use a BiLSTM network to construct word representations from characters in it (Lample et al., 2016). This architecture is denoted as Character Embeddings. It has been shown that learning character embeddings has been found useful for capturing morphological evidence (Ling et al., 2015). Figure 2.2 describes the architecture using character embeddings and BiLSTM to generate word embedding for word "Paris". The input to the BiLSTM is the letter sequence of a word. We define a character dictionary which maps each character to a d-dimensional vector representation. The English character dictionary contains uppercase and lowercase letters, numbers, and punctuation. We look up each c_i of the input letter sequence from the dictionary and get the character embedding vectors $X : \{x_1, x_2, \ldots, x_n\}$. Then, character embedding vectors X are fed into BiLSTM to generate forward and backward hidden embeddings of the character sequence. We concatenate the last forward hidden embedding $\overrightarrow{h_n}$ and the last backward hidden embedding $\overleftarrow{h_1}$ with the embeddings from word embedding dictionary lookup to obtain the final word embedding.

We add Character Embeddings in the sequence tagging system by concatenating the output of Character Embeddings and the word embeddings from lookup to form input embeddings. Then, we feed the input embeddings to BiLSTM and CRF, and we get the fully structured BiLSTM model (BiLSTM-CRF) for sequence tagging. The character embeddings and word embeddings are learned together during training. There are existing implementations of BiLSTM-CRF, such as NeuroNet by Dernoncourt et al. (2017) which

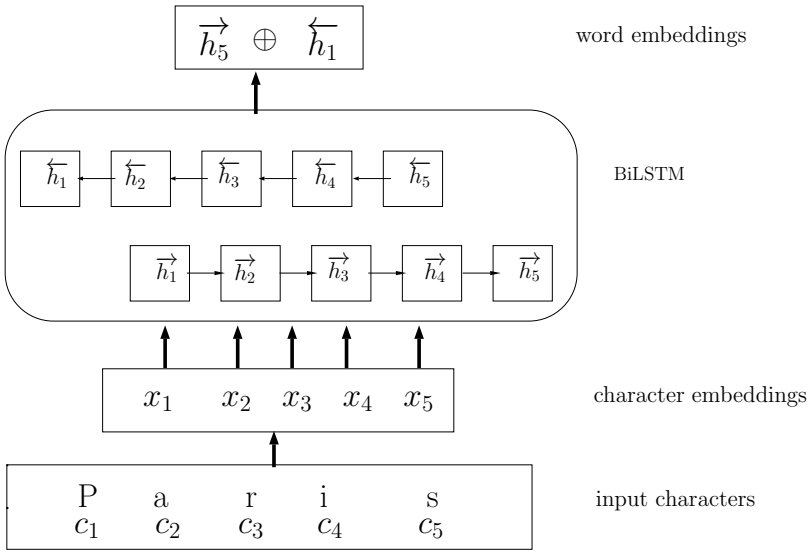


Figure 2.2: The word embedding derived from the character embeddings

achieves state-of-the-art performance. In order to compare BiLSTM-CRF with other models in this thesis, we re-implement the model.

2.1.3 Conditional Random Fields

Conditional Random Fields (CRF) models make decisions on sentence levels instead of making independent decisions on each word. We combine the BiLSTM model with a CRF layer to form the BiLSTM-CRF model. The architecture of BiLSTM-CRF is shown in Figure 2.3.

In BiLSTM-CRF, given a sequence of output predictions $y = (y_1, y_2, \ldots y_n)$, the score of the output sequence is:

$$S(y) = \sum_{i}^{n} T_{i,i+1} + \sum_{i}^{n} p_i(y_i) \tag{2.1}$$

where T is a matrix of transition scores such that $T_{i,j}$ represents the score of a transition from the label i to label j, and $p_i(y_i)$ is the probability of y_i being the label of word i.

During training, we score every possible output sequence, and use a softmax layer (Dugas et al., 2001) to generate the probability distribution of output sequences: $P(y) = \mathit{softmax}(S(y))$. Then, we minimize the negative log likelihood over the training sentences. During decoding, we can recover the tag sequences of test data using dynamic program-

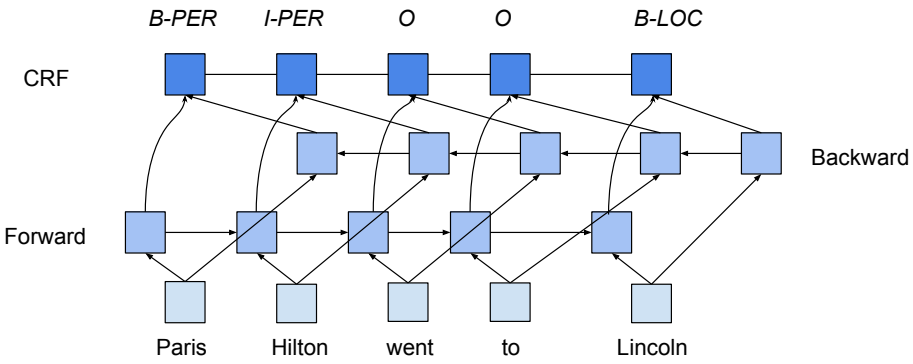


Figure 2.3: The architecture of BiLSTM-CRF on an NER example

ming. Thereby, decoding time grows quadratically in the number of tag types, and grows linearly in the number of sentences.

2.2 Experiments and Results

To evaluate BiLSTM models, we run BiLSTM with three configurations on POS, NER, and Chunking: BiLSTM with word features only (BiLSTM); BiLSTM with Character Embeddings (BiLSTM-Char); BiLSTM with Character Embeddings and a CRF layer (BiLSTM-CRF). We report the performance and decoding speed of the models on Penn Treebank data set for POS, CoNLL 2003 data set for NER, CoNLL 2000 data set for Chunking. The details of the data sets are shown in Table 1.1.

Table 2.1: Experiments Specification

CPU	Intel i7-5820k
GPU	NIVID GeForce GTK 1080
OS	Ubuntu 16.04.03
CUDA	8.0
Tensorflow	1.0
Python	2.7.13

The experiments were conducted on a Linux server with a single 10G Nvidia GeForce GPU. The experiment specifications are shown in Table 2.1. We use the hyperparameters defined in NeuroNER (Dernoncourt et al., 2017), which uses BiLSTM-CRF for NER and is shown to achieve near the state-of-the-art performance. Since dropout training (Hinton et al., 2012) can improve the performance by encouraging the model to depend on both character embeddings and word embeddings, we apply a dropout mask on the input em-

Table 2.2: Hyperparameters used in BiLSTM Models

Hyperparameters	Values
character embedding size	50
word embedding size	100
feedforward layer size	200
feedforward layer	1
BiLSTM layer size	100
BiLSTM layer	2
optimizer	Adam
learning rate	0.01
batch size	32
dropout	0.5

beddings before the BiLSTM layer. The dropout rate is set to 0.5 in the experiments. Table 2.2 shows the hyperparameters used in BiLSTM models. We use the pretrained word embeddings from GloVe which contains 40K words, and initialize all the weight parameters using Xavier initialization (Glorot et al., 2011).

Table 2.3: Performance of BiLSTM Models on POS

Model	Accuracy	Error Reduction (words)
BiLSTM	96.1	–
BiLSTM-Char	97.21 (+1.11)	629
BiLSTM-CRF	**97.34** (+1.24)	702

Table 2.4: Performance of BiLSTM Models on NER CoNLL 2003 Data

Model	Precision	Recall	F1
BiLSTM	84.27	83.22	83.74
BiLSTM-Char	87.07	88.81	87.93 (+4.19)
BiLSTM-CRF	89.93	90.16	**90.05** (+6.31)

Table 2.5: Performance of BiLSTM Models on Chunking

Model	Precision	Recall	F1
BiLSTM	91.26	92.33	91.79
BiLSTM-Char	92.51	93.45	92.98 (+1.19)
BiLSTM-CRF	93.91	93.81	**93.86** (+2.07)

Table 2.3, Table 2.4 and Table 2.5 describe the final performance of BiLSTM models on the POS, NER and Chunking. Table 2.6 reports the decoding speed of BiLSTM models. In terms of performance, the fully structured BiLSTM model outperforms the other two. Compared to BiLSTM, the structure of Character Embeddings increases the performance, but BiLSTM networks do not heavily depend on features other than words. CRF layer is more helpful for NER than POS and Chunking. In terms of decoding speed, it is obvious

Table 2.6: BiLSTM Models Decoding Speed (words/sec)

Model	POS	NER (CoNLL 2003)	Chunking
BiLSTM-CRF	9009	10100	5390
BiLSTM-Char	13992 (1.55×)	11271 (×1.12)	6616 (×1.23)
BiLSTM	23036 (2.56×)	20740 (×2.05)	10321 (×1.91)

that the more features the model has, the slower it would be. Adding a CRF layer will increase the performance but decrease the decoding speed.

Chapter 3

Feedforward Neural Network Models

In this chapter, we the greedy tagging system with a feedforward network, which is denoted as Feedforward-History. In the experiments, we show the performance and decoding speed of feedforward models on POS and NER.

3.1 Feedforward-History Model

Inspired by the greedy parser system by Chen and Manning (2014), we present a similar greedy transition system for sequence tagging in this thesis. The greedy parser system employs a basic arc-standard system (Nivre and Scholz, 2004), which consists of three types of transitions (LEFT-ARC, RIGHT-ARC, and SHIFT), a stack, and a buffer. While the greedy parser system has three types of actions, the sequence tagging system only needs the SHIFT action which predicts the tag of the current word in the buffer and shifts the word to the stack. Syntaxnet from Google also uses the same model for POS and dependency parsing and achieves near the state-of-the-art performance on both tasks (Alberti et al., 2017).

In our greedy tagging system, we use a feedforward network to make decisions on individual words, and we assume that the word to be labeled depends mainly on its neighbor words instead of the whole sentence. Besides the word features, the system also takes into account the lexical composition of the words (spelling features), and the previous tag decisions (previous tag features). Thereby, we represent this architecture as the Feedforward-History model. The way Feedforward-History incorporates word features is the same with the window approach proposed in Collobert et al. (2011). The difference between these two models is that Feedforward-History uses spelling features and previous tag features while the model in Collobert et al. (2011) only uses word features.

Figure 3.1 illustrates the process of how a greedy tagging system uses Feedforward-History to decode a POS example. Since the focused word only depends on its neighbors

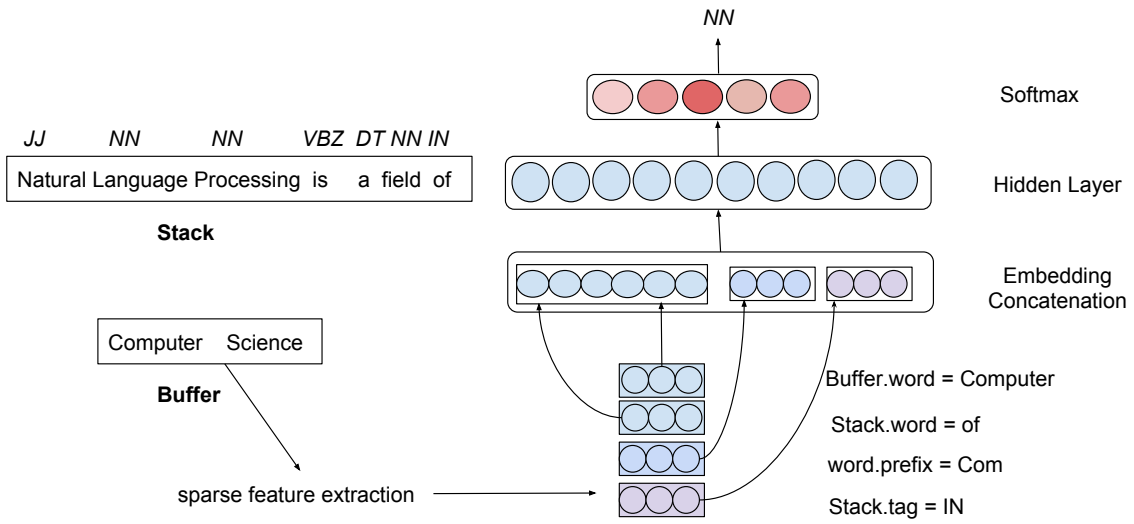


Figure 3.1: The architecture of a greedy tagging system using the Feedforward-History model on a POS example.

in this greedy tagging system, we use a fixed size window around the current word to generate features. In order to generate features for the words at the beginning and at the end of the sentence, we add special padding words at the beginning and at the end. To generate dense word features, we convert each word in the input sequence to a d-dimensional vector representation e_{w_i}. Meanwhile, we have a full vocabulary embedding dictionary E_w. Given a word w_i, we look up its embedding e_{w_i} in E_w. According to Ratnaparkhi et al. (1996), spelling features of a word can help predict the part-of-speech tag. The examples of spelling features are upper and lower case features, prefix and suffix features, digit features, etc. Each spelling feature of a word is also associated with an embedding vector e_{s_i}, and it can be looked up from a embedding vector dictionary E_s. Besides word features and spelling features, we incorporate the previous tag features in this model. Each previous tag decision is represented as e_{ht_i} and can be looked up from a tag dictionary represented as E_{HT}.

The input layer $X = (x_1, x_2, \ldots x_n)$ to the feedforward network is obtained by concatenating the word feature vectors, spelling feature vectors, and previous tag feature vectors. In general, generating a lot of hand-engineered features for sequence tagging is expensive: selecting useful features is an empirical process based on trials and errors, and computing feature vectors requires searching feature strings in huge dictionaries and concatenating them together. We try to use as little hand-engineered features as possible to save the time from feature generation while keeping the model accurate. In our implementation, we extract the following spelling features of each word: whether it starts with a capital letter; whether it has all capital letters; whether it has a mix of letters and digits; whether if has

punctuation; and letter prefixes and suffices of length two and three. We also extract 4 previous tags as parts of input features.

While the input layer is the concatenation of the feature vectors of the focused word, the the output layer is a probability distribution over tags. The input unit x_i is mapped to a hidden unit h_i through the rectifier activation function ($ReLU$) (Nair and Hinton, 2010):

$$ReLU(x) = Max(0, x) \tag{3.1}$$

$$h_i = ReLU\left(W_1^w x_i^w + W_1^s x_i^s + W_1^l x_i^l + b_1\right), \tag{3.2}$$

where x^w represents the word features, x^s represents the spelling features, x^l represents previous tag features, W_1 is the weight parameters in the hidden layer, and b_1 is the bias of in the hidden layer. The output of the network is a probability distribution over tags, and its dimension is the size of all tags. The tag probability distribution is modeled by a softmax layer:

$$p_i = softmax(W_2 h_i + b_2), \tag{3.3}$$

where W_2 is the weight parameter in the softmax layer and b_2 is the bias in the softmax layer. The network is trained by minimizing a negative log likelihood over the training data. Embedding vectors are trained together with weight vectors and bias in the model. We denote all trainable parameters as θ. Given a sequence predictions, $Y = (y_1, y_2, \ldots y_n)$, the score of the output sequence is the sum of the probability of each decision y_i:

$$S(Y) = \sum_i^n p_i(y_i), \tag{3.4}$$

and the loss function is:

$$L(\theta) = -log\left(\sum_i^n p_i(y_i)\right), \tag{3.5}$$

3.2 Experiments and Results

In POS experiments, we train the models using the Penn Treebank training data and development data. Then, we decode the test data using trained models and record the per word accuracy and decoding time. In NER experiments, we train the models using the CoNLL 2003 training data and development data. Then we decode the test data using the trained models, and record the F1 score and decoding time. In Chunking experiments, we train the models using the CoNLL 2000 training data and development data. Then we decode the test data using the trained models, and record the F1 score and decoding time.

The speed is measured by the average number of words decoded per second. We are also interested in the robustness of the Feedfoward-History model, so we build a model using a feedforward network with only word features, which serves as the baseline model in our experiments. The tagging system using a feedforward network with only word features is denoted as Feedforward in this thesis.

Table 3.1: Hyperparameters used in Feedforward Models

Hyperparameters	Values
window size	8
word embedding size	100
feedforward layer	1
feedforward layer size	200
optimizer	Adam
learning rate	0.01
batch size	32

Table 3.2: Performance of Feedforward Models on POS

Model	Accuracy	Error Reduction (words)
Feedforward	95.89	–
Feedforward-History	97.28 (+1.39)	788
BiLSTM-CRF	97.34 (+1.45)	822

Table 3.3: Performance of Feedforward Models on NER CoNLL 2003

Model	Precision	Recall	F1
Feedforward	82.99	83.59	83.29
Feedforward-History	87.68	87.27	87.47 (+4.18)
BiLSTM-CRF	89.93	90.16	90.05 (+6.76)

Table 3.4: Performance of Feedforward Models on Chunking

Model	Precision	Recall	F1
Feedforward	89.35	91.55	90.43
Feedforward-History	92.52	92.69	92.61 (+2.18)
BiLSTM-CRF	93.91	93.81	93.86 (+3.43)

The hardware specifications are listed in Table 2.1. In NER experiments, because of the small amount of training data, we use GloVE pretrained word embedding where each word corresponds to a 100-dimensional embedding vector. We use the hyperparameters defined in Syntaxnet (Alberti et al., 2017), which contains a greedy POS tagging system shown to achieve near the state-of-the-art performance. Specifically we use Adam (Kingma and Ba, 2014) for stochastic optimization, and set the learning rate to be 0.01. We use 1 hidden layer and set the hidden layer size to be 200. We have a batch implementation which can

Table 3.5: Feedforward Models Decoding Speed (words/sec)

Model	POS	NER (CoNLL 2003)	Chunking
Feedforward	30967	26819	16920
Feedforward-History	19474 (-1.59×)	17609 (-1.52×)	10067 (-1.68×)
BiLSTM-CRF	9009 (-3.43×)	10100 (-2.65×)	5390 (-3.14×)

process multiple sentences at the same time, and we set the batch size to be 32 in the experiments. Table 3.1 shows the hyperparameters. We initialize all the weight parameters using Xavier initialization (Glorot et al., 2011). The experiments specification is shown in Table 2.1.

Table 3.2 presents the POS results achieved by greedy feedforward models and BiLSTM-CRF. Table 3.3 presents the final benchmark results including Precision, Recall, and F1 score of feedforward models on NER. Table 3.4 presents the performance of the models on Chunking. In all three sequence tagging tasks, BiLSTM-CRF outperforms Feedforward-History. BiLSTM-CRF improves the NER performance the most because of the dependencies among output tags. Table 3.5 shows the decoding speed of Feedforward and Feedforward-History. Feedforward-History has faster decoding speed then BiLSTM-CRF on all three sequence tagging tasks.

Chapter 4

Mention2Vec Models

In this chapter, we introduce multitask models based on Mention2Vec. We conduct experiments using the multitask models on sequence tagging tasks and compare their performance and decoding speed with Feedforward and BiLSTM-CRF

4.1 Model Description

4.1.1 Feedforward-Mention2Vec for NER

Mention2Vec is a neural network model for NER, which uses BiLSTMs to predict boundaries and entity types separately (Stratos, 2016). We summarize Mention2Vec into two steps. The first step uses a BiLSTM to generate hidden embeddings the same way in the BiLSTM-CRF model and predicts the boundaries using hidden embeddings. Unlike usual NER labels (B-LOC, I-LOC, ...), boundary labels in this step do not have NER types attached. Since the boundary labels have strong correlations, Mention2Vec uses CRF to capture correlations and produce boundary label sequences.

Figure 4.1 illustrates the first step of Mention2Vec on an NER example. We denote the input word sequence as $W : \{w_1, w_2, \ldots, w_n\}$, input embeddings as $X : \{x_1, x_2, \ldots, x_n\}$ which are the concatenations of the character embeddings and the word embeddings, and the hidden embeddings as $H : \{h_1, h_2, \ldots, h_n\}$.

The output boundary label probability distribution is denoted as p_i for word w_i, and the gold boundary label sequence is denoted as Y_{label}. In each training step, the boundary detection loss is given by the negative log likelihood of the gold boundary label sequence, shown in Equation 4.1

$$L_1(\theta_1) = -\log\left(p\left(Y_{label}|h_1, h_2 \ldots h_n\right)\right) \tag{4.1}$$

The second step of Mention2Vec is type prediction which finds actual types for named entity spans in a sequence. It makes use of the hidden embeddings and the entity boundaries from the first step. The model first looks up the hidden embeddings for the words in entity

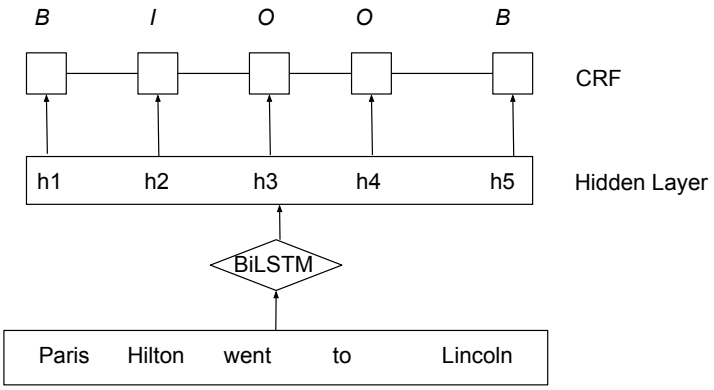


Figure 4.1: The first step of Mention2Vec for NER

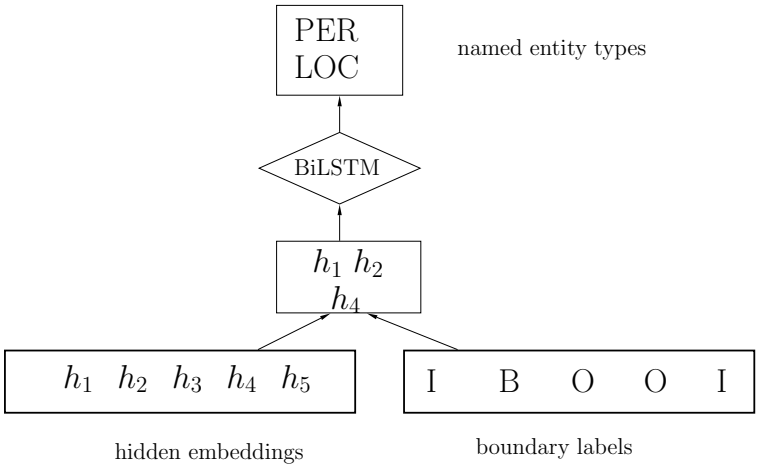


Figure 4.2: The second step of Mention2Vec for NER

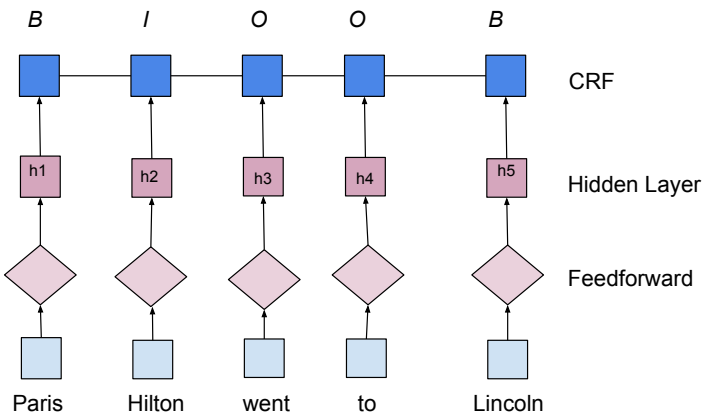


Figure 4.3: The first step of Feedforward-Mention2Vec

spans. Then, it feeds the hidden embeddings into an additional BiLSTM and obtains the type probability distributions. Figure 4.2 illustrates the third step of Mention2Vec.

The gold type output of an input sequence is denoted as Y_{type}. Assuming there are l named entities in a sequence, and the index of the first word in a named entity is represented as s, and the index of the last word in a named entity is represented as e, the type prediction loss is computed by 4.2:

$$L_2(\theta_2) = -\sum_l \log P\left(r^l | h_s^l \ldots h_e^l\right) \tag{4.2}$$

During training, the model uses the gold boundaries and gold types to compute the boundary detection loss and the type prediction loss. In each training step, the boundary detection loss and the type prediction loss are minimized jointly: the training objective is to find boundary sequences and type sequences that minimize the sum of L_1 and L_2.

In order to further speed up Mention2Vec as well as capture the correlation between boundary tags, we consider using a different network for boundary detection. Shown in Chapter 3, feedforward networks can produce relatively good performance on sequence tagging with faster speed than BiLSTM. We then replace the BiLSTM in the first step of Mention2Vec with a feedforward network. We denote this new model as Feedforward-Mention2Vec. Feedforward-Mention2Vec still has two steps where the second step is the same in Mention2Vec. The first step uses a feedforward network to produce the hidden embeddings, which is illustrated in Figure 4.3.

In both Feedforward-Mention2Vec and Mention2Vec, the run time of decoding boundaries is constant with the number of types, since there are only three types of boundaries (I, O, B) to decode in the CRF layer. In other CRF based models, like BiLSTM-CRF, the decoding time grows quadratically in the number of named entity types.

4.1.2 BPE-Mention2Vec for POS

In POS, each word in the input sentence is assigned a unique part-of-speech tag. Since there is no tag span existing in POS, it's not necessary to use a multitask model on POS. However, we propose a way to convert POS into an NER-like task through the help of Byte Pair Encoding. Inspired by the successful results obtained by using BPE in machine translation (Sennrich et al., 2016), we initially wanted to use BPE to capture morphological decomposition of the words and replace spelling features like prefixes and suffixes. BPE is a compression algorithm which replaces frequent pairs with an unused byte. Sennrich et al. (2016) presents a way to adapt BPE for word segmentation: using BPE to segment words into subword units of different length, and building a vocabulary dictionary using word frequency. In POS tagging system, we first learn BPE merge operations on the training data. To segment training data and test data, we first split each word in characters and then apply BPE to merge characters into larger chunks. In order to restore the words, we use the "IOB" label scheme to label the subword units. Since there can be multiple subword units sharing the same tag, POS becomes a task similar with NER. Figure 4.4 shows an example of using BPE to segment a sequence with POS tags.

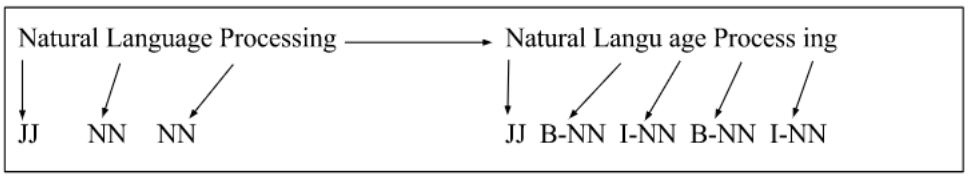


Figure 4.4: An example of using BPE for word segmentation

In our proposed BPE-Mention2Vec model for POS, there are three main steps. In the first step, given a sequence, we use BPE to segment the words and convert the corresponding tags using the "IOB" label scheme. After the first step, we have an NER-like task with known boundary of each entity span, so we can apply the same method in Feedforward-Mention2Vec to predict the POS types for entity spans. In the second step, we use a feedforward network to produce the hidden embeddings of the input words. CRF is not used here because the boundary labels are known in both training and decoding. The third step of BPE-Mention2Vec uses a BiLSTM to predict the POS tags for each entity span based on the hidden embeddings and the known boundaries. Figure 4.5 describes the process of using BPE-Mention2Vec to find POS tags for a sequence.

4.2 Experiments and Results

We empirically evaluate the Mention2Vec model and the Feedforward-Mention2Vec model for NER, and the BPE-Mention2Vec model for POS. The hardware specifications are listed

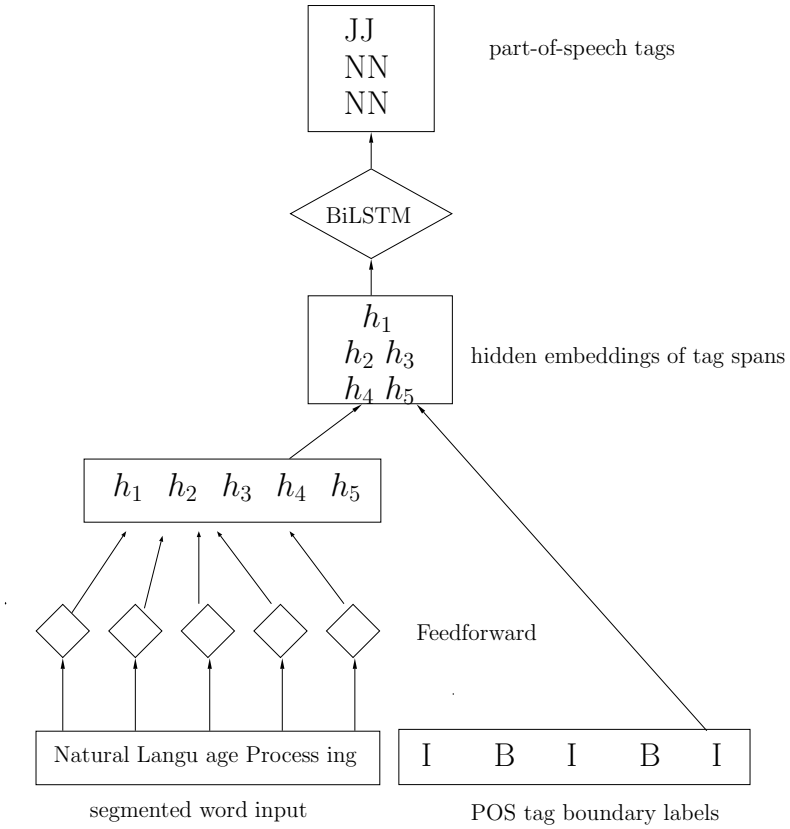


Figure 4.5: An example of using BPE-Mention2Vec to find POS tags

in Table 2.1. The experiments were conducted on a Linux server with a single 10G Nvidia GeForce GPU. The experiments specification is shown in Table 2.1. In the implementation of Mention2Vec, we use the same set of hyperparameters from the origin model (Stratos, 2016). We perform minimum hyperparameter tuning for Feedforward-Mention2Vec and BPE-Mention2Vec over hidden layer size, learning rate, and dropout rate. The hyperparameters we use in Feedforward-Mention2Vec and BPE-Mention2Vec are shown in Table 4.1.

Table 4.1: Hyperparameters used in Feedforward-Mention2Vec and BPE-Mention2Vec

Hyperparameters	Values
character embedding size	50
word embedding size	100
feedforward layer size	200
feedforward layer	1
optimizer	Adam
learning rate	0.001
batch size	32
dropout rate	0.5

Table 4.2: NER F1 Score and Decoding Speed Comparison on CoNLL 2003

Model	Precision	Recall	F1	Speed
Feedforward	82.99	83.59	83.29	26819
BiLSTM-CRF	89.93	90.16	**90.05** (+6.76)	10100 (-2.65×)
Mention2Vec	89.14	88.99	89.06 (+5.77)	9701 (-2.76×)
Feedforward-Mention2Vec	88.98	88.21	88.6 (+5.31)	13450 (-1.99×)

Table 4.3: Chunking F1 Score and Decoding Speed Comparison on CoNLL 2000

Model	Precision	Recall	F1	Speed
Feedforward	89.35	91.55	90.43	16920
BiLSTM-CRF	93.91	93.81	**93.86** (+3.43)	5390 (-3.14×)
Mention2Vec	93.03	93.13	93.08 (+2.65)	5465 (-3.09×)
Feedforward-Mention2Vec	92.61	92.62	92.61 (+2.18)	7601 (-2.23×)

In Chunking and NER experiments, we compare Feedforward-Mention2Vec with Mention2Vec on the CoNLL 2003 data set. We use the BiLSTM-CRF and Feedforward as baseline models because BiLSTM-CRF achieves the highest F1 score and Feedforward has the fastest decoding speed among the previous models we built. Table 4.2 shows the NER performance and decoding speed of these neural network models on the CoNLL 2003 data set.Table 4.3 shows the Chunking performance and decoding speed of these neural network models on the CoNLL 2000 data set. Mention2Vec and Feedforward-Mention2Vec obtain lower F1 score than BiLSTM-CRF. Feedforward-Mention2Vec is faster than Mention2Vec

Table 4.4: NER F1 Score and Decoding Speed Comparison on OntoNotes

Model	Precision	Recall	F1	Speed(words/sec)
Feedforward	82.98	62.09	71.03	22829
BiLSTM-CRF	86.59	85.21	**85.90**	7667 (-2.97×)
Mention2Vec	86.24	84.25	85.23	8433 (-2.71×)
Feedforward-Mention2Vec	85.40	79.92	82.57	10812 (-2.11×)

Table 4.5: Per-label F1 Score on OntoNotes

Labels	Feedforward	Feedforward-Mention2Vec	BiLSTM-Char-CRF
CARDINAL	72.38	78.03	80.33
DATE:	72.45	80.61	81.24
EVENT:	30.01	45.81	61.40
FAC:	29.08	46.27	57.94
GPE:	89.62	92.84	94.84
LANGUAGE:	45.16	43.42	51.43
LAW:	20.02	43.62	57.97
LOC:	52.79	64.71	73.51
MONEY:	77.73	81.66	86.98
NORP:	84.11	87.80	92.15
ORDINAL:	71.26	72.77	77.24
ORG:	72.05	79.92	85.27
PERCENT:	84.43	89.27	88.73
PERSON:	82.72	86.98	90.65
PRODUCT:	51.66	50.00	64.38
QUANTITY:	65.78	74.77	81.13
TIME:	41.68	56.44	58.85
WORK OF ART:	26.29	37.63	47.21

and BiLSTM-CRF. The empirical results demonstrate that the Feedforward-Mention2Vec model performs competitively on Chunking and NER while being faster than original Mention2Vec model and the state-of-the-art model. Feedforward is still the fastest but it has the lowest F1 score on both tasks.

Different NER corpus may have different named entity types. We have noted that Feedforward-Mention2Vec and Mention2Vec scale linearly in the number of named entity types while BiLSTM-CRF grows quadratically. We want to show the time difference of the same model on NER data sets with different numbers of named entity types. We have obtained the performance and decoding speed of these models on CoNLL 2003 which has 4 types of named entity. Then we conduct the experiments on the OntoNotes English data set which has 18 types of named entity. Table 4.4 reports the final results of the experiments. The same model obtains lower F1 score on OntoNotes. Table 4.5 shows the per label performance of Feedforward-Mention2Vec, BiLSTM-CRF, and Feedforward. They reveal that some labels in OntoNotes are classified poorly, such as "WORK OF ART" and

"LANGUAGE". We suspect this is due to OntoNotes is more noisy than CoNLL 2003 as OntoNotes data is extracted from a wide variety of sources with more named entity types. In terms of decoding speed, while Feedforward-Mention2Vec is 1.3 times faster than the BiLSTM-CRF model on the data set with 4 named entity types, Feedforward-Mention2Vec is 1.4 times faster on the data set with 18 named entity types. Mention2Vec is slightly slower than BiLSTM–CRF on CoNLL 2003, but it's faster than BiLSTM-CRF on OntoNotes.

Table 4.6: POS Tagging Systems Performance and speed Comparison

Model	Accuracy	Error Reduction	Speed
Feedforward	95.89	–	30967
BPE-Mention2Vec	96.04 (+0.15)	85	4923 (-6.29×)
BILSTM-CRF	97.34 (+1.45)	822	9009 (-3.43×)

In the POS experiments, we compare BPE-Mention2Vec with BiLSTM-CRF which is the state-of-the-art model for POS, and with Feedforward which is the fastest among the models we have built. Table 4.6 demonstrates the performance and decoding speed of them on the Penn Treebank data set. BPE-Mention2Vec obtains lower accuracy than BiLSTM-CRF and higher accuracy than Feedforward. Since using word segmentation increases the number of words to be processed and introduces more preprocessing time, BPE-Mention2Vec is slower than the BiLSTM-CRF and Feedforward. The empirical results conclude that BiLSTM-CRF is a better model than BPE-Mention2Vec on POS.

Chapter 5

Discussion

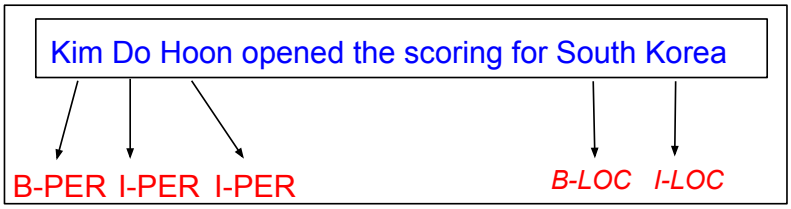
In this Chapter, we put together the final experiment results of the network models presented in this thesis, and provide an analysis of the results.

Table 5.1 records the performance of the neural network models for POS, NER, and Chunking. The POS performance is measured on the Penn Treebank test data set, the NER performance is measured on the CoNLL 2003 test data set, and the Chunking performance is measured on CoNLL 2000 test data set. Table 5.1 also includes the results from the state-of-the-art models. While our implementations obtain lower accuracy scores and F1 scores than the state-of-art results, we emphasize that our main goal of this thesis is to compare different neural networks. Since the performance scores of the state-of-the-art model are already high, we are willing to trade in some performance scores for speed improvement. Figure 5.1 and Figure 5.2 display two NER examples extracted from CoNLL 2003 test data where BiLSTM-CRF performs better than Feedforward-History and Feedforward-Mention2Vec.

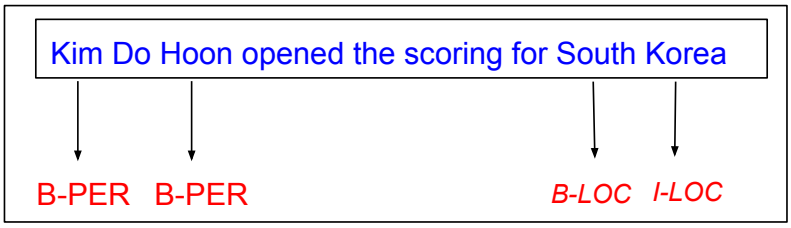
Table 5.1: Neural Network Models Accuracy and F1 Score

Model	Penn Treebank	CoNLL 2003	CoNLL 2000
Ling et al. (2015)	**97.78**	–	–
Lample et al. (2016)	–	**90.94**	–
Shen and Sarkar (2005)	–	–	**95.23**
Indig and István (2016)			95.08
Feedforward	95.89	83.29	90.43
Feedforward-History	97.28	87.47	92.61
BiLSTM	96.1	83.74	91.79
BiLSTM-Char	97.21	87.93	92.98
BiLSTM-CRF	*97.34*	*90.05*	*93.86*
Mention2Vec	–	89.06	93.08
Feedforward-Mention2Vec	–	88.6	92.61
BPE-Mention2Vec	96.04	–	–

Table 5.2 records the decoding speed of different neural network models on the Penn Treebank, CoNLL 2003 and CoNLL 2000 test data. The models are sorted by their decoding

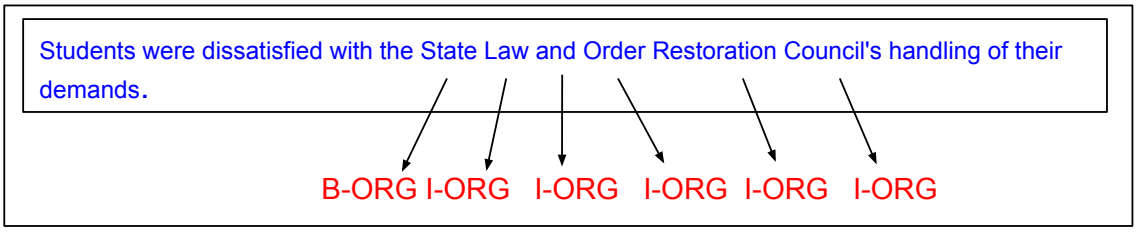


(a) Result Obtained by BiLSTM-CRF

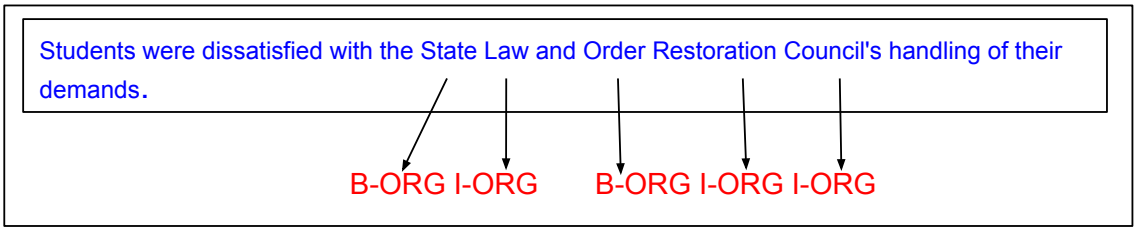


(b) Result Obtained by Feedforward-History

Figure 5.1: An Comparison between BiLSTM-CRF and Feedforward-History on an NER Example



(a) Result Obtained by BiLSTM-CRF



(b) Result Obtained by Feedforward-Mention2Vec

Figure 5.2: An Comparison between BiLSTM-CRF and Feedforward-Mention2Vec on an NER Example

Table 5.2: Neural Network Models Decoding Speed

Model	Penn Treebank	CoNLL 2003	CoNLL 2000
Feedforward	30967	26819	16920
BiLSTM	23036	20740	10321
Feedforward-History	19474	17609	10067
Feedforward-Mention2Vec	–	13450	7601
BiLSTM-Char	13992	11271	6616
BiLSTM-CRF	9009	10100	5390
BPE-Mention2Vec	4923	–	–

speed in descending order. The decoding speed is measured by the number of words decoded per second in the test time. It is obvious that the fewer features used in the same model the faster the decoding speed of the model will be. In general, the greedy tagging systems using feedforward models are faster then the systems using BiLSTM models. Since the CRF based models introduce a transition matrix and uses dynamic programming algorithm to decode the sequence, they are slower than the models without the CRF layer.

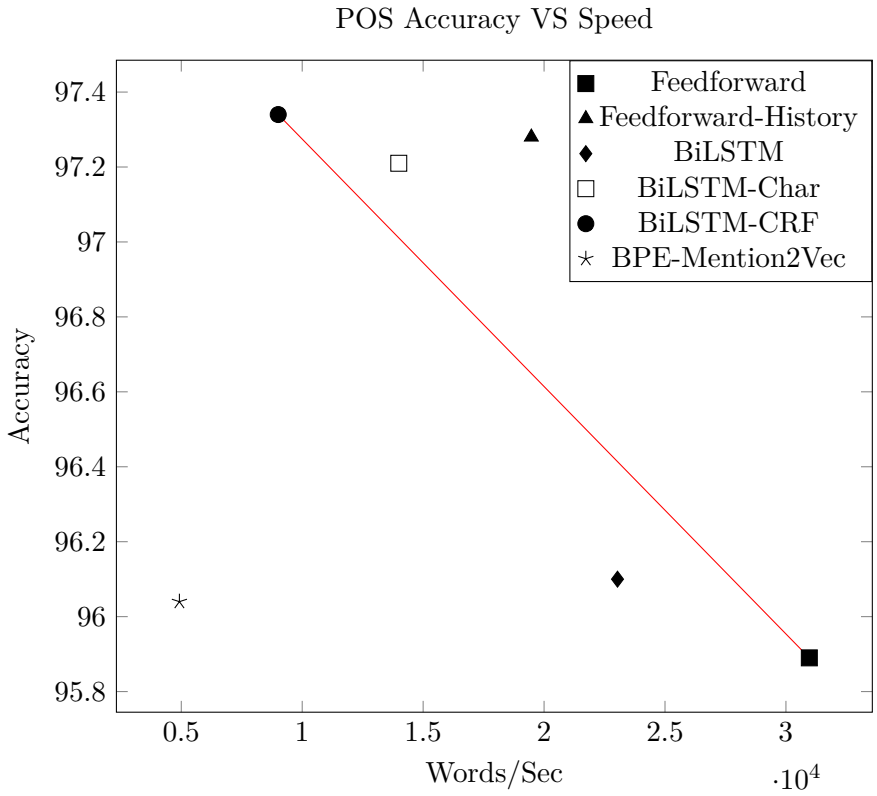


Figure 5.3: Results of the POS system using different Neural Network Models

Figure 5.3 illustrates the trade-off between performance and decoding speed in POS systems using different neural network models. Among the neural network models for POS, BiLSTM-CRF achieves the best per word accuracy 97.34%, and Feedforward-History ob-

tains the second best per word accuracy 97.28%. Feedforward achieves the fastest decoding speed since it employs a simple neural network without extra features. The line in Figure 5.3 connects BiLSTM-CRF which is the most accurate model and Feedforward which is the fastest model. The models above the line are faster but perform slightly worse than BiLSTM-CRF, such as Feedforward-History and BiLSTM-Char. Models below the line such as BPE-Mention2Vec are slower and less accurate, which makes them less ideal for POS. Feedforward-History is the fastest model with competitive performance on POS, and it is about 2 times faster than BiLSTM-CRF.

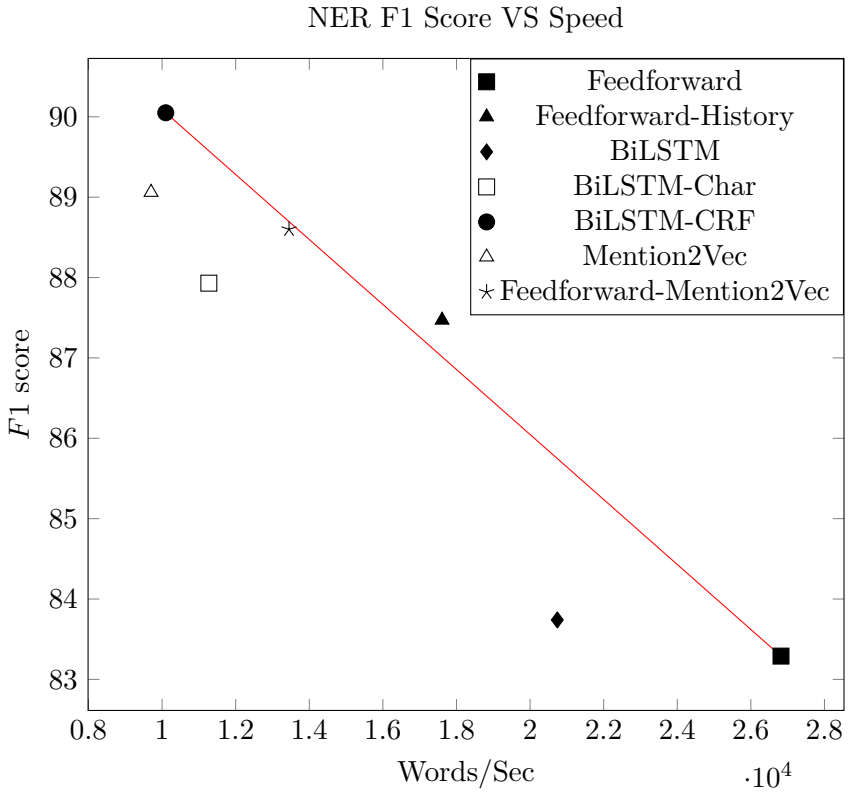


Figure 5.4: Results of the NER system using different Neural Network Models on CoNLL 2003

Figure 5.4 illustrates the trade-off between performance and speed of using different neural network models on CoNLL 2003. BiLSTM–CRF achieves the highest F1 Score 90.05, and Mention2Vec obtains the second best F1 Score 89.06. Feedforward achieves the fastest decoding speed since it employs a simple neural network with no extra features. The line in Figure 5.4 connects the BiLSTM-CRF model which is the most accurate model and the Feedforward model which is the fastest model. The models above the line are faster but perform slightly worse than BiLSTM-CRF, such as Feedforward-History. Our new model, Feedforward-Mention2Vec, comes very close to the line, and it achieves 88.6 F1 score which

is close to the best performance, and it is 1.3 times faster than the fully structured BiLSTM model.

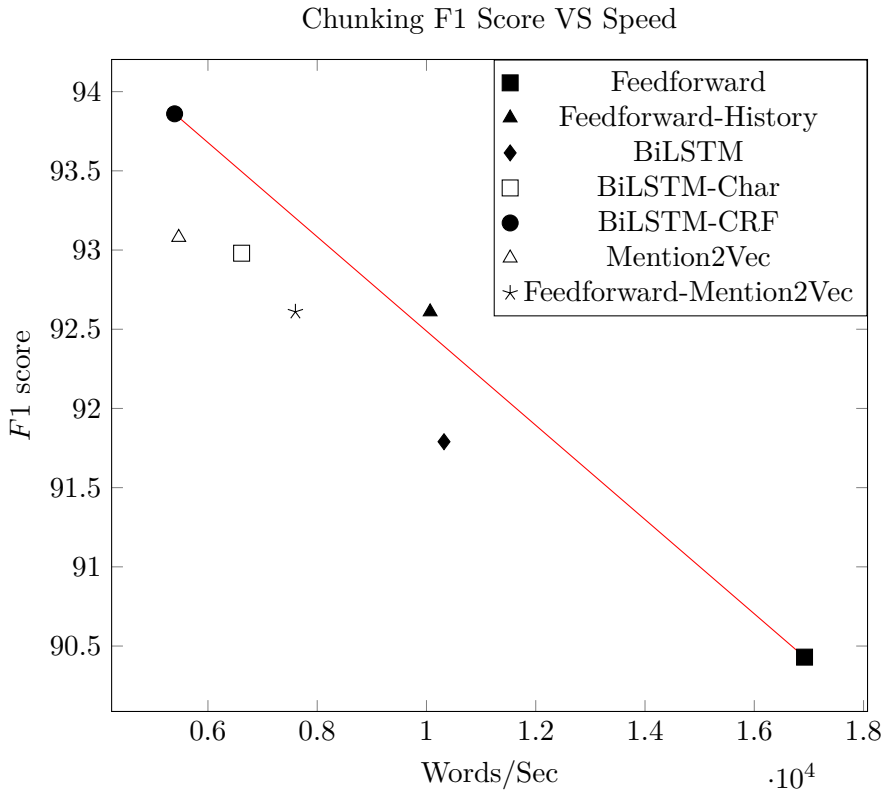


Figure 5.5: Results of the Chunking system using different Neural Network Models on CoNLL 2000

Figure 5.5 illustrates the trade-off between performance and speed of using different neural network models on CoNLL 2000. BiLSTM-CRF achieves the highest F1 Score 93.86, and Mention2Vec obtains the second best F1 Score 89.06. Feedforward achieves the fastest decoding speed since it employs a simple neural network with no extra features. The line in Figure 5.4 connects the BiLSTM-CRF model which is the most accurate model and the Feedforward model which is the fastest model. The models above the line are faster but perform slightly worse than BiLSTM-CRF, such as Feedforward-History. Models below the line are slower and less accurate, which makes them less ideal for Chunking.

As illustrated in Figure 5.3, Figure 5.4 and Figure 5.5, the greedy sequence tagging systems using a feedforward network (Feedforward-History) can achieve comparable performance and faster speed than the systems using recurrent models; the multitask model (Feedforward-Mention2Vec) performs competitively with the state-of-the-art model on NER.

Table 5.3 compares the performance and decoding speed of BiLSTM-CRF and Feedforward-Mention2Vec on CoNLL 2003 and OntoNotes. It shows that Feedforward-Mention2Vec with a simpler architecture can achieve competitive performance with BiLSTM-CRF on CoNLL

Table 5.3: F1 Scores and Decoding Speed on CoNLL 2003 and OntoNotes

	F1 Score		Decoding Speed (Words/Sec)	
	CoNLL 2003	OntoNotes	CoNLL 2003	OntoNotes
Feedforward-Mention2Vec	88.6	82.57	13445 (1.3×)	10812 (1.4×)
BiLSTM-CRF	90.05	85.90	10100	7667

2003 and OntoNotes. The speed gap between Feedforward-Mention2Vec and BiLSTM-CRF grows as the number of named entity type increases. There is more speed benefit in using a multitask model like Feedforward-Mention2Vec when there are more named entity types to classify.

Chapter 6

Conclusion and Future Work

This thesis presents and compares different neural network models for sequence tagging tasks. The empirical results reveal that simple feedforward networks can achieve competitive results while being significantly faster than the BiLSTM networks. The empirical results also demonstrate that Feedforward-Mention2Vec performs well on Chunking and NER, and it is more scalable in the number of named entity types in NER.

6.1 Contribution

In this thesis, we first built the the state-of-the-art model for sequence tagging: BiLSTM-CRF. We ran the BiLSTM model with different configurations on three sequence tagging task: POS, Chunking and NER.

Then, we implemented the greedy feedforward sequence tagging system. We compared the decoding speed and performance between BiLSTM models and feedforward models. As the experiments show, BiLSTM models are more accurate then feedforward models in general. Since feedforward models have a simpler architecture and fewer parameters, feedforward models are faster then BiLSTM models. Experiments also show that the feedforward models are not strongly dependent on hand engineered features, and the models are able to automatically learn the useful features for making decisions.

In addition to feedforward models and BiLSTM models, we presented Feedforward-Mention2Vec for Chunking and NER which is a combination of feedforward models and Mention2Vec, and BPE-Mention2Vec for POS which is based on BPE and Mention2Vec. Both of these two models are multitask models. Feedforward-Mention2Vec predicts boundaries of tags and types of the tags separately: it uses a feedforward network and CRF for boundary detection, and a BiLSTM for type prediction. BPE-Mention2Vec first segments words using BPE, and predicts the part-of-speech tags for the subword units using a BiLSTM network. Feedforward-Mention2Vec performs slightly worse than the state-of-the-art model (BiLSTM-CRF), but its decoding speed is faster. In NER, as the number of named

entity types grows, the decoding time of Feedforward-Mention2Vec grows linearly while the decoding time of BiLSTM-CRF grows quadratically.

Lastly, we summarized all the experiment results and compared the performance and decoding speed of all the models presented in this thesis. We provided analysis on the speed and accuracy trade-off in different models. Feedforward is the fastest among all models, since it contains no extra features and employs a simple network architecture. Our re-implementation of BiLSTM-CRF achieves near state-of-the-art performance on POS and NER. Feedforward-History performs better on POS then on Chunking and NER with a faster decoding speed than BiLSTM-CRF. Feedforward-Mention2Vec performs slightly worse than BiLSTM-CRF on NER, but its decoding speed is faster and grows linearly in the number of named entity types.

6.2 Future Work

There are several potential extensions of this thesis we would like to work on in the future:

First, even though the main goal of this thesis is to compare different neural network models and reveal the speed vs accuracy trade-off, we would like to improve the performance of our models and minimize the accuracy gap between the our implementations and the state-of-the-art results, especially for Chunking. Since the number of training data in CoNLL 2000 for Chunking is limiting, the performance of Chunking can be improved by increasing the amount of training data by self-training. We can train a model on labeled training, run on lot of unlabeled data and then re-train on combined data sets. The same method can be applied on NER and POS.

Second, we would like to compare our implementations with the off-the-shelf sequence tagging tools, such as spaCy. We can further develop our models into applications with high data throughput. The applications can be used to analyze real time natural language data, such as Twitter, Facebook, Wikipedia, and even the web.

Third, we would like to explore multitasking models which combine POS, Chunking and NER. We can build a neural network tagging system to train and predict POS tags, Chunking tags, and NER tags jointly. The intermediate representations would benefit from considering linguistic hierarchies in the training process.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Alberti, C., Andor, D., Bogatyy, I., Collins, M., Gillick, D., Kong, L., Koo, T., Ma, J., Omernick, M., Petrov, S., Thanapirom, C., Tung, Z., and Weiss, D. (2017). Syntaxnet models for the conll 2017 shared task. *CoRR*, abs/1703.04929.

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M. (2016). Globally normalized transition-based neural networks. In *ACL (1)*. The Association for Computer Linguistics.

Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In Moschitti, A., Pang, B., and Daelemans, W., editors, *EMNLP*, pages 740–750. ACL.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In Wu, D., Carpuat, M., Carreras, X., and Vecchi, E. M., editors, *SSST@EMNLP*, pages 103–111. Association for Computational Linguistics.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, 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.

Dernoncourt, F., Lee, J. Y., and Szolovits, P. (2017). NeuroNER: an easy-to-use program for named-entity recognition based on neural networks. *EMNLP*.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., and Garcia, R. (2001). Incorporating second-order functional knowledge for better option pricing. In *Advances in neural information processing systems*, pages 472–478.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *ACL (1)*, pages 334–343. The Association for Computer Linguistics.

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.

Gage, P. (1994). A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

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.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *CoRR*, abs/1508.01991.

Indig, B. and István, E. (2016). Gut, Besser, Chunker – Selecting the best models for text chunking with voting. In *Computational Linguistics and Intelligent Text Processing - 17th International Conference, CICLing 2016, Konya, Turkey, April 3-9, 2016*. Springer.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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*, ConLL '00, pages 142–144, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In Knight, K., Nenkova, A., and Rambow, O., editors, *HLT-NAACL*, pages 260–270. The Association for Computational Linguistics.

Ling, W., Dyer, C., Black, A. W., Trancoso, I., Fermandez, R., Amir, S., Marujo, L., and LuÃŋs, T. (2015). Finding function in form: Compositional character models for open vocabulary word representation. In MÃărquez, L., Callison-Burch, C., Su, J., Pighin, D., and Marton, Y., editors, *EMNLP*, pages 1520–1530. The Association for Computational Linguistics.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

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.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Nivre, J. and Scholz, M. (2004). Deterministic dependency parsing of english text. In *Proceedings of the 20th international conference on Computational Linguistics*, page 64. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Ratnaparkhi, A. et al. (1996). A maximum entropy model for part-of-speech tagging. In *EMNLP*, volume 1, pages 133–142. Philadelphia, PA.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *ACL (1)*. The Association for Computer 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.

Stratos, K. (2016). Mention2vec: Entity identification as multitasking. *CoRR*, abs/1612.02706.

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.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.