Jointly Learning Representations for Low-Resource Information Extraction

by

Nanyun Peng

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

July, 2017

Abstract

This thesis explores information extraction (IE) in *low-resource* conditions, in which the quantity of high-quality human annotations are insufficient to fit statistical machine learning models. Such conditions increasingly arise in domains where annotations are expensive to obtain, such as biomedicine, or domains that are rapidly changing, such as social media, where annotations easily become out-of-date. It is crucial to leverage as many learning signals and as much human knowledge as possible to mitigate the problem of inadequate supervision.

In this thesis, we focus on two typical IE tasks: named entity recognition (NER) and entity relation extraction (RE). We explore two directions to help information extraction with limited supervision: 1). learning representations/knowledge from heterogeneous sources using deep neural networks and transferring the learned knowledge; and 2). incorporating structural knowledge into the design of the models to learn robust representations and make holistic decisions. Specifically, for the application of NER, we explore transfer

ABSTRACT

learning, including multi-task learning, domain adaptation, and multi-task domain adaptation, in the context of neural representation learning in order to help transfer learned knowledge from related tasks and domains to the problem of interest.

For the applications of entity relation extraction and joint entity recognition and relation extraction, we explore incorporating linguistic structure and domain knowledge into the design of the models, thus yielding more robust systems with less supervision.

Committee:

Mark Dredze

Jason Eisner

Kevin Duh

Acknowledgments

I would like to express my sincere thanks to my advisor, Mark Dredze, who guided my through the five years adventure of my Ph.D study. Mark gave me so much freedom to explore the research directions, yet was so helpful whenever I faced obstacles. He inspired me in many ways, among which I was most gratefully learned (and continue learning) is to strike balances between different objectives: theoretical neatness and practical deployability, technical depth and application breath, work and family...I believe the things I learned from Mark will be a life-long treasure for me.

I had the fortune to have Jason Eisner, Kevin Duh, alongside Mark Dredze, as my thesis committee members. Jason is a long-time mentor for me, who talks super fast yet is very patient in explaining. He always has good advises to share and challenges people in a positive way. I remember learning semirings, finite state machines and much more from the projects with Jason. His passion attitude of pushing everything to perfection is enduringly inspiring. Kevin has also been an amazing mentor. He is always upbeat and

encouraging. He always gives sparkling and practical suggestions for the projects, and always generous to share hands-on experiences. I wish Kevin could join CLSP earlier so that I can work with him more.

Beyond my committee, I am lucky to have worked with and learned from Jiri Navratil, Yaser Al-onaizan during my internships at IBM Watson Center, and Hoifung Poon, Chris Quirk, Scott Yih, Kristina Toutanova and Ming-Wei Chang during my internships at Microsoft Research. They broadened my research horizon and showed me the wide open world outside the school. I also benefited greatly from the intellectual discussions with them. I am also thankful for my former advisor, Houfeng Wang, who gave me the opportunity to continue the study and explore in the fields of NLP and machine learning as a person with only linguistics background.

I was fortunate to get into the PhD program at CLSP and be affiliated with the HLT-COE (the two NLP centers at JHU). I benefited tremendously from the support and advice of many faculty over the years, especially Ben Van Durme, David Yarowsky, Suchi Saria, Dan Povey, Sanjeev Khudanpur, Philipp Koehn, and Raman Arora. I also owe a lot to my fellow CLSP students whose helpful discussions have helped me along the way: Ryan Cotterell, Michael Paul, Frank Ferraro, Matt Gormley, Anni Irvine, Travis Wolfe, Nick Andrews, Yiming Wang, Hainan Xu, Dingquan Wang, Tongfei Chen, Sheng Zhang, Shuoyang Ding, Adrian Benton, Rebecca Knowles, Tao Chen, Zachary Wood-Doughty,

ACKNOWLEDGMENTS

Rachel Rudinger, Naomi Saphra, Gaurav Kumar, Chu-Cheng Lin, Keisuke Sakaguchi, Adithya Renduchintala, Jonathan Jones, Ke Li, Chunxi Liu, Xiaohui Zhang, Hongyuan Mei, Adam Poliak, Pegah Ghahremani, Adam Teichert, Tim Vieira, Nathaniel Filardo, Svitlana Volkova, Xuchen Yao, Guoguo Chen, Yuan Cao, Puyang Xu, Ming Sun, and certainly others.

I met a lot of great people during my internships and at conferences, such as He He, Yi Luan, Max Ma, Marjan Ghazvininejad, Daniel Fried, Zachary Lipton, Hao Chen, Haoruo Peng, Zi Yang, Diyi Yang... I constantly learn from the conversations with them.

Finally, I am grateful for the love and support from my family and friends, especially my husband K.W. Chang, who was so busy yet always put me in the first priority whenever I needed him; and my parents Y.Z. Peng and G.H. Wang, who gave me so much trust and encouragement to pursue whatever I wanted.

Dedication

To the numerous naive decisions that led me here.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Vast quantities of human knowledge are carried in written languages. While this knowledge is crucial for many applications, such as question answering, market analysis, and precision medicine, such unstructured knowledge is mostly inaccessible to computers and overwhelming for human experts to absorb. Computer systems rely on structured representations (e.g., databases) to organize and retrieve knowledge. Therefore, **Information Extraction (IE)** – processing *raw* text to produce machine understandable *structured* information – is a crucial step to dramatically increase the accessibility of knowledge through search engines, interactive AI agents, and medical research tools. To handle the

ambiguity and nuance of human languages, IE tasks require deep understanding of language, including words, phrases, and their underlying structures to form sentences, paragraphs and, discourses.

Figure 1.1 demonstrates an example of a typical IE task: relation extraction from articles on biomedical cancer genomics. In this task, a computer extracts entities (human genes, mutations, and drugs) and their relations (i.e., respond, resistant, sensitive) from *raw* texts. Extremely specialized knowledge, such as a biomedical graduate degree, is usually required for a person to understand these relations; our goal is to enable a computer to perform this same task.

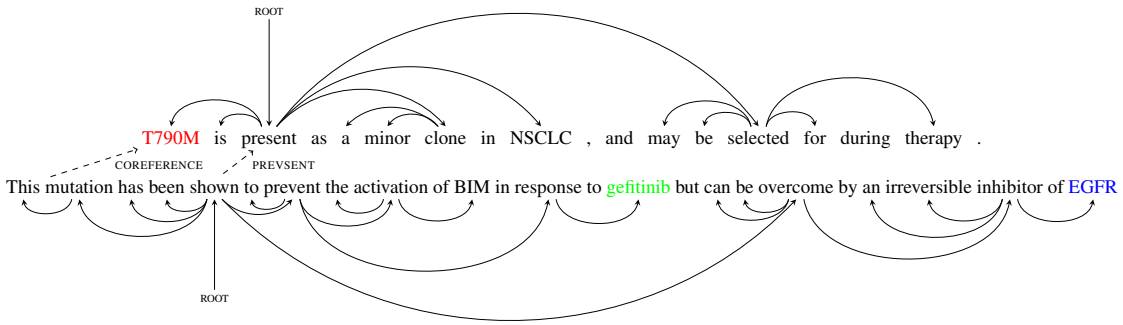


Figure 1.1: An example paragraph expressing a ternary relation among drug, gene, and mutation (tumors with a T790M mutation in the EGFR gene resist gefitinib treatment). The edges denote the underlying linguistic structures of the sentences, which are important for understanding the relation.

Traditional closed-domain IE systems usually formulate IE tasks as supervised learn-

ing problems, which assumes the availability of abundant high-quality human annotations to train machine learning models. Such system also typically utilize NLP tools to produce linguistic analysis such as tokenization, part-of-speech tagging, and syntactic parsing as features to help train the machine learning models. However, many highly specialized domains, such as biomedicine, lack such resources/tools and human annotations; this problem is also common in social media (and other rapidly changing domains), where the annotations soon become out-of-date. Besides the domain shift problem, there are languages lack abundant NLP resources, termed low-resource languages. Such language lack high quality NLP pre- processing tools to analyze the raw text and produce linguistic structures. This hinders the scaling up of information extraction to many languages, domains, and tasks.

This thesis explores *information extraction in low-resource settings*, for which we strive to reduce the required availability of resources and human annotations for IE systems. This is an interesting and challenging direction for several reasons. First, most IE tasks require comprehensive understandings of sentences, discourse structures, and domain knowledge to perform well; thus, IE tasks provide a challenging testbed for natural language understanding. Second, most information extraction problems are naturally structural, with predictions involving many interdependent variables that would be better to determine jointly. This raises modeling and computational challenges. Third, informa-

tion and knowledge are growing at an exploding rate; new entities, relations, events, and facts emerge everyday, from different domains, different languages, and in different forms. This invalidates the assumptions in most traditional IE systems of fully supervised learning with abundant resources and annotations. Robust methods that require fewer human annotations and are applicable to many domains, languages, and settings are desirable.

1.2 Thesis Statement

This thesis demonstrates that learning low-dimensional representations for characters, words, and multi-word units with joint models can overcome the challenges posed by small training corpora to information extraction tasks.

The joint models leverage *unlabeled data*, annotations for *related tasks* and *related domains*, *multiple steps in the NLP pipeline*, and *linguistic structures*, thus yielding robust performance for *low-resource* IE. The jointly learned representations encapsulate distributional semantics of languages, the knowledge from other tasks and domains, and comprehensive information from linguistics structures. Therefore, they can easily generalize to new data, reducing the demands of repeatedly annotating many data for each new task, domain, and language.

1.3 Contributions

This dissertation advances two approaches to learning better representations for IE tasks. One is transfer learning from unlabeled texts, annotations from other tasks and other domains. Another is learning robust representations with rich structure modeling. The following two sections give more details about each direction and highlight their contributions.

1.3.1 Representation Learning from Heterogeneous Sources

For the first direction, we explored two new aspects of transferring learned representations: higher level sharing of representations, and more flexible sharing settings.

Previous work on transfer learning usually involved learning representations for words and conducted transfer learning thereafter. However, we looked at the higher level representations of *words in context*, which are learned by bidirectional Long Short- Term Memory networks (LSTMs). Transferring higher level representations effectively shares more information and obtains better performance; combining both lower and higher level representations yields even more gains.

We also proposed a general framework that combines two well-studied settings in the literature on transfer learning: 1). multi-task learning (MTL) and 2). multi-domain learn-

ing (MDL), in order to leverage information from *both* data annotated for other tasks *and* data obtained from other domains. This framework outperformed either MTL or MDL alone. Under this framework, we pioneered a novel mismatch setting, where the annotations from a different task *and* a different domain are used to learn transferable representations to help a certain task. The experiments demonstrated that it is possible to learn useful representations from mismatched annotations.

We experimented with this framework on sequence tagging models, which are widely applicable in Natural Language Processing (NLP). The applications to the problem of Named Entity Recognition (NER; identifying person, organization, and location names from texts) in Chinese social media showed that using annotations as small as 1,500 messages from Weibo (a Chinese microblogging site), we achieved more than 50% relative improvement on F1 score over a strong baseline, as well as over the Stanford NER system (Finkel et al., 2005), using the multi-task and multi-domain transfer learning framework.

1.3.2 Deep Learning with Explicit Structure Modeling

Language has deep structure (e.g., syntactic and semantic relations), hidden in the written form of texts. Many linguistic theories have been proposed to express the underlying structure of languages, including grammar, coreference, and discourse dependencies. Traditional machine learning methods for information extraction benefited from incorporating

these linguistic structures as features, or building structured models that encode linguistic dependencies. Recent advances in deep neural networks, on the other hand, have largely focused on end- to-end models that directly learn from raw data, disregarding linguistic structures.

Nonetheless, blindly and exclusively learning from data does not work well when the amount of training data is insufficient. Explicitly encoding structural information into neural architectures strikes a balance between adding useful prior knowledge and learning generalizable representations. Therefore, we propose to *jointly learn* robust representations by incorporating linguistic structures into the design of neural networks, and making *joint inference* of several stages in the IE pipeline to make holistic decisions.

We develop a graph-based long short-term memory networks (graph LSTMs) model that builds recurrent neural networks on pre-defined graphs, the edges of which incorporate grammar, coreference, and discourse dependencies between words. By explicitly modeling linguistic structures, the model learns word representations that comprehensively encode information from different aspects of the contexts. We apply the model to the cancer genomics domain to extract from PubMed articles ternary relations among drugs for cancer treatments, genes, and mutations. The proposed model yields significant improvements over both traditional feature-based methods with linguistic features and neural architectures that do not consider linguistic structures.

We also propose to conduct joint named entity recognition and entity relation extraction by both jointly learning representation via recurrent neural networks (RNNs) *and* joint inference through integer linear programming (ILP). The ILP formulations efficiently incorporate domain knowledge about the entities and their relations to make holistic decisions; the jointly trained RNNs also learn more robust representations that are suitable to be used as features for both NER and RE tasks. The combination of ILP joint inference and joint learning through DNNs combines advantages of both. In the applications to ACE 2005 entity and relation extraction shared task, our model achieves better results than separately trained and inferred models.

1.4 Dissertation Outline

The rest of this dissertation is arranged as follows. Chapter 2 starts with an overview of related work on information extraction, machine learning models for IE, and learning under low-resource conditions. Chapter 3 summarizes the work we have done on multi-task learning for Chinese social media NER with different neural architectures and different levels of parameter sharing. Chapter 4 introduces a novel recurrent neural network architecture that enables joint learning of representations and sharing parameters for multiple domains *and* multiple tasks to achieve multi-task domain adaptation. A novel idea of simultaneously sharing representations among all tasks and domains, tying parameters be-

tween decoders for the same tasks in different domains is proposed. Both of these two chapters fall into the direction introduced in Section 1.3.1: representation learning from heterogeneous sources.

Chapters 5 and 6 explore another direction: representation learning with explicit structure modeling. Chapter 5 proposes a graph-LSTM model that builds recurrent neural networks on pre-defined graphs, the edges of which incorporate grammar, coreference and discourse structures between words to learn more robust representations. Chapter 6 proposes a joint learning and inference framework for entity and relation extraction that combines ILP joint inference with joint representation learning to make holistic decisions of entity and relation types. Chapter 7 then concludes the thesis with discussions and outlook for future work.

1.5 Other Publications

During my PhD. study, I have published other research related to the topics of information extraction applications, and joint learning and joint inference. Some of these research led to improved state-of-the-art of certain IE tasks, others developed useful software and tools, or yielded mathematical and modeling tools that can facilitate IE tasks. Here, I briefly overview the work with references.

In the space of IE applications, Peng et al. (2015c) studied name variances in Chinese

and explored methods for matching names referring to the same entity using only the name strings. Better name matching, I demonstrated, can facilitate certain down-stream IE tasks such as cross-document coreference resolution. Finin et al. (2015) explored entity detection and linking for both English and Chinese, and achieved good performance on the TAC KBP 2015 shared task. Peng et al. (2015b) created a pipeline to automatically annotate Chinese that includes the IE tasks of named entity recognition and relation extraction.

Work in the space of joint learning and inference including Peng et al. (2014), which proposed joint learning of polylingual topic models from code-switched tweets. The joint model learned to simultaneously identify language and assign topic for the tokens in the code-switched tweets. Cotterell et al. (2014) formulated a graphical model for the joint learning and inference of the underlying forms of morphemes based on the observed word forms. Peng et al. (2015a) explored exact inference methods for graphical models over string-valued variables, where the variables' domain is an infinite space of discrete structures which brought special challenges. We proposed a dual decomposition inference method to achieve exact MAP inference for this challenging problem.

Chapter 2

Background

This chapter provides background necessary for better understanding of this dissertation, including the problem setup, the methodology, and the application scenario. This chapter is organized as follows:

Section 2.1, overviews the research on information extraction (IE). I begin by introducing the IE pipeline, a standard natural language processing process for extracting knowledge from narrative text. I then discuss two common IE tasks the are the main focus of this thesis: 1). name entity recognition (NER) and 2). relation extraction (RE). I demonstrate the challenges of handling variances in domain and language, which cause severe drops in performance when annotation is scarce.

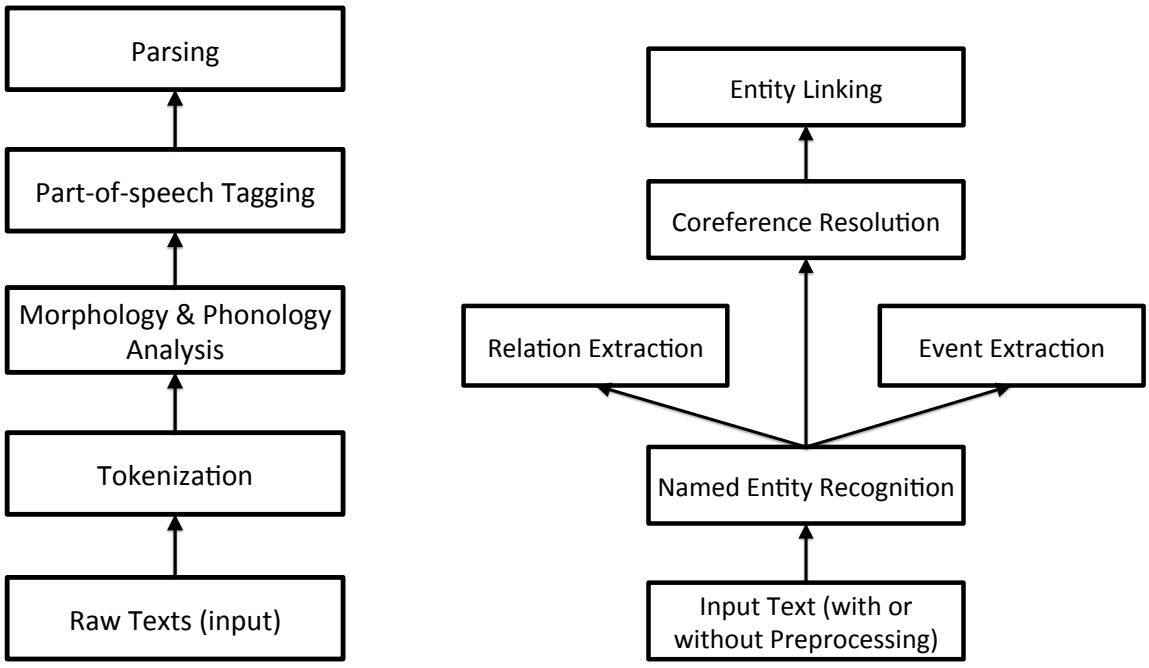
Section 2.2 introduces the important basics of statistical models for information extrac-

tion, including structured models, word representation learning, and the recent advances of deep neural network models for NLP. These techniques serve as stepping stones for the new models proposed in this thesis, which combine representation learning with structured models and explore the benefit of joint inference and learning.

Section 2.3 discusses the problem of learning with limited annotation, which is a practical problem in many real-world applications, including information extraction. Techniques such as distant supervision, transfer learning including multi-task and multi-domain transfer, are introduced to facilitate a later discussion of bridge representation learning with heterogeneous supervision signals.

2.1 Information Extraction

Information Extraction aims to extract structured knowledge from unstructured text (e.g., news articles, social media texts, scientific publications, medical records, product reviews) to construct knowledge databases. IE comprises many subtasks that loosely form a pipeline, the flow of which this section introduces with a focus on two essential tasks: named entity recognition and relation extraction. I then discuss the problem of variances in domain and language in real-world IE applications.



(a) NLP pre-processing pipeline for IE.

(b) Information extraction pipeline.

Figure 2.1: An illustration of the NLP and IE pipelines.

2.1.1 Pipelines

Before detailing the information extraction pipeline, let us step back and look at the NLP pre-processing pipeline for IE. The NLP pre-processing pipeline contains many basic tasks, the outputs of which serve as useful features for the IE tasks.

2.1.1.1 Pre-processing Pipeline

Figure 2.1a illustrates the NLP pre-processing pipeline for IE. Given raw text input, the pre-processing pipeline first tokenizes the text into words. This step is easier for some languages than for others. Specifically, the Indo-European languages have natural delimiters in their writing systems, facilitating tokenization. On the other hand, many Asian languages, such as Chinese, lack natural delimiters in writing, raising special challenges for processing those languages. Chapter 3 discusses how to handle such situation in more details.

After tokenization, morphological and phonological analyses can be conducted on the words to obtain their inner structures – stems and inflectional parts – with respect to writing forms and sounds, respectively. These analyses reduce the number of distinct word forms, enhancing the generalizability of the models. During my PhD study, I have researched morphological and phonological modeling for inflectional languages (Cotterell et al., 2015), and improved joint inference techniques for morphology / phonology models (Peng et al., 2015a). Though these will not be elaborated here for coherence, readers are encouraged to peruse these papers.

One step further in the pre-processing pipeline, part-of-speech tagging and parsing analyze the syntactic aspects of the input sentences, providing important information about the syntactic taxonomy of the words and the dependencies between them. In our work

on relation extraction for biomedical publications, covering in Chapter 5, we utilized the syntactic structure in the design of the neural architectures to build a more robust model for relation extraction.

2.1.1.2 Information Extraction Pipeline

Here I first overview the steps in the information extraction pipeline, and then detail in the following sections two essential tasks, named entity recognition and relation extraction.

Figure 2.1b shows an overview of the information extraction pipeline. This pipeline takes as input the representations of the raw text, which can come from the aforementioned NLP pre-processing pipeline or be directly extracted from the raw text using deep neural networks. Note that despite recent studies showing that deep neural networks can learn good representations of raw text to facilitate the end task with sufficient annotated training data, I will demonstrate in this thesis that when annotated training data are scarce, the information produced by the NLP pipeline is especially beneficial.

The first step in the information extraction pipeline is entity recognition (Collins and Singer, 1999), which detects boundaries of text chunks referring to certain entities and decides their types.

Relation extraction (Bunescu and Mooney, 2005; Banko et al., 2007), coreference resolution (Soon et al., 2001), and event extraction (Ananiadou et al., 2010) all rely on the

results of entity recognition. Relation extraction detects and classifies the semantic relationships among entity mentions in the text, thus producing structured knowledge about the entities. Coreference resolution decides whether certain entity mentions refer to some entities mentioned in prior contexts, enriching the information about any single entity mention. Entity linking (Dredze et al., 2010b; Ratinov et al., 2011) aims to link entity mentions in contexts to some structured knowledge base, thus obtaining comprehensive information about those entity mentions. Event extraction detects event mentions from the texts. An event usually involves several entities (arguments) and a trigger. Event extraction is done by deciding the extents and triggers of the event mentions.

This thesis mainly focuses on the tasks of entity detection and relation extraction, because they reside at the lower level of the IE pipeline where improvements catalyze the downstream tasks. Moreover, the results of the extraction will be useful for populating existing relational knowledge bases for other research fields.

The divide of the tasks in the pipeline is admittedly sometimes artificial, implemented for computational efficiency and modularization. Many tasks in the pipeline are mutually informative and can be learned together to make holistic decisions. Entity and relation extraction is a great example of this type. In Chapter 6, we explore joint learning and inference of entity mentions and relations in the raw text, yielding exciting discoveries.

2.1.2 Named Entity Recognition

U.S. intelligence officials now believe North Korea is developing the technology to make nuclear warheads small enough to fit atop the country 's growing arsenal of missiles potentially putting Tokyo and U.S. troops based in Japan at risk , according to officials who have received the intelligence reports .

Figure 2.2: An example sentence from a CNN news report with entity mention annotations.

Named entity recognition, or the more general task entity mention detection, is the first step in and an essential component of the information extraction pipeline. It involves detecting the boundaries of the noun or pronoun phrases that correspond to entities [1], and determining their entity types.

The most common form of entity mentions is *named* entities (e.g., United States, Apple, Jane Austin). Pronouns (e.g., he, she, who, it, they) and nominal mentions (the girl, mother, the company, etc.) are also prevalent. Nominal mentions often contain a word that is designated as the head word. For instance, the head word of the nominal mention "the former president" is "president". For named and pronoun mentions, the head words

[1]Entities usually include person, organization, location and geo-political entities, but other types are possible, depending on the application domain. For instance, in the example in Figure 1.1, the entity types are drug, gene, protein, and mutation

are usually the mentions themselves (can contain several words). For certain downstream tasks, such as relation extraction and coreference resolution, it usually suffices only to consider the entity mention heads.

Figure 2.2 gives example annotations of entity mentions for a sentence from a CNN news report. The entity types are color-coded, with red indicating geo-political entities, green indicating organizations, blue indicating persons, and purple indicating weapons. Different darknesses in the same color family represent different types [2] of mentions.

A long line of work has focused on NER in both formal and informal domains (Collins and Singer, 1999; McCallum and Li, 2003; Nadeau and Sekine, 2007; Jin and Chen, 2008; He et al., 2012a), with recent efforts turning towards social media (Finin et al., 2010; Liu et al., 2011; Ritter et al., 2011; Li et al., 2012; Liu et al., 2012a; Fromreide et al., 2014). While work on NER has been conducted for several languages, work on social media NER has largely focused on English language data.[3]

In this thesis, we expand the scope to Chinese social media, both because of the popularity of the service and the special challenges posed by the Chinese language. Moreover, this is a first step towards robust IE systems that are applicable to more domains and languages with fewer annotations.

[2]The darkest to the lightest color refer to named, pronoun, and nominal mention, respectively

[3]Etter et al. (2013) considered Spanish Twitter, which is quite similar to English from the standpoint of building models and features.

2.1.3 Relation Extraction

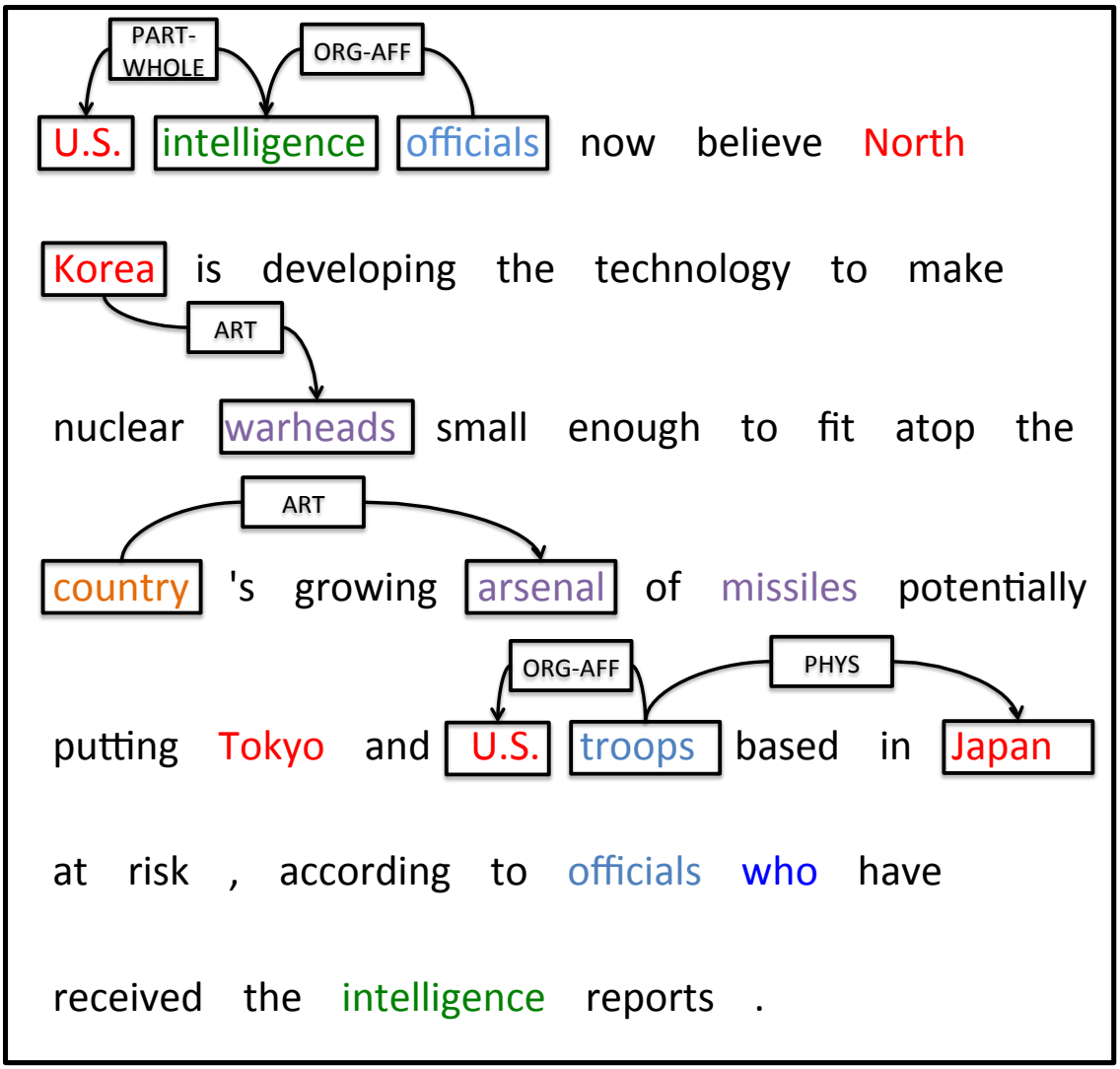


Figure 2.3: An example sentence from a CNN news article with entity and relation mention annotations.

Relation Extraction detects and classifies semantic relationships among entity men-

tions in a certain context. Figure 2.3 shows some example relation mentions for the same sentence as in Figure 2.2. The relations are directional, with "PART-WHOLE" standing for the "part and whole" relation, "ORG-AFF" meaning the "organization and affiliation" relation, "ART" referring to the "person and artifact" relation, and "PHYS" standing for the "physically located in" relation.

There are several different formulations for conducting relation extraction. Open class RE (Banko et al., 2007) extracts the *relation words* along with the involved entities from given sentences; closed class RE (Bunescu and Mooney, 2005) pre-defines a relation set which may not appear in the given sentence, and decides which relation (or no relation) exists for the entities; and slot filling initiated by the TAC-KBP shared task aims to complete the information about each entity by filling the slots of relations. This dissertation focuses on closed class relation extraction.

Most work on closed class RE has been applied to binary relations of entities in a single sentence. However, some work, including this thesis (in Chapter 5), has explored beyond this scope to investigate n-ary *and* cross-sentence relation extraction. In this section, I briefly review prior work on binary, single-sentence relation extraction, n-ary relation extraction, and cross-sentence relation extraction. These problems are usually formulated as multi-class classification problems. Given the boundaries and types of the involved entities, a classifier need to decide whether the entities has certain semantic relations among

them or not.

2.1.3.1 Binary Relation Extraction

This widely used task setting is most frequently applied to extraction from a single sentence. Traditional feature-based methods rely on carefully designed features to learn good models, and often integrate diverse sources of evidence, such as word sequences and syntax context (Kambhatla, 2004; Zhou et al., 2005; Boschee et al., 2005; Suchanek et al., 2006; Chan and Roth, 2010; Nguyen and Grishman, 2014). Kernel-based methods design various subsequence or tree kernels (Mooney and Bunescu, 2005; Bunescu and Mooney, 2005; Qian et al., 2008) to capture structured information. More recently, along with the revival of deep neural networks, various neural architectures have been proposed to approach the relation extraction problem. Models for relation extraction are discussed in more details in Section 2.2.2.

2.1.3.2 N-ary Relation Extraction

Early work on extracting relations among more than two arguments was done in MUC-7 benchmark, with a focus on fact and event extraction from news articles (Chinchor, 1998). Semantic role labeling in the Propbank (Palmer et al., 2005) or FrameNet (Baker et al., 1998) style are also instances of n-ary relation extraction, with extraction of events

expressed in a single sentence. McDonald et al. (2005) extracted n-ary relations in the biomedical domain by first factoring the n-ary relations into pair-wise relations between all entity pairs and then constructing maximal cliques of related entities. Most of this work has looked at relations expressed within a single sentence and has focused on feature-engineered models.

2.1.3.3 Cross-Sentence Relation Extraction

Several relation extraction tasks have gone beyond sentence boundaries, including MUC fact and event extraction (Swampillai and Stevenson, 2011), record extraction from web pages (Wick et al., 2006), extraction of facts for biomedical domains (Yoshikawa et al., 2011), and extensions of semantic role labeling to cover implicit inter-sentential arguments (Gerber and Chai, 2010). These prior works have either relied on explicit co-reference annotation or assumed that the whole document refers to a single coherent event, simplifying the problem and reducing the need for powerful representations of multi-sentential contexts of entity mentions. Recently, cross-sentence relation extraction models have been learned with distant supervision, and used integrated contextual evidence of diverse types without relying on these assumptions (Quirk and Poon, 2017).

2.1.4 Variance in Domains and Languages

Most work on information extraction had focused on English news articles, facilitated by benchmark data comprising CoNLL (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), MUC (Grishman and Sundheim, 1996; Chinchor, 1998), ACE (Doddington et al., 2004; Walker et al., 2006), and DEFT-ERE (Linguistics Data Consortium, 2014).

However, researchers have long noticed that other domains, such as scientific articles (Bodenreider, 2004; Bundschus et al., 2008; Bui et al., 2011; Bodnari et al., 2012), online reviews and discussion forums (Collins and Singer, 1999; McCallum and Li, 2003; Nadeau and Sekine, 2007; Jin and Chen, 2008; He et al., 2012a), and recent emerging domain social media (Finin et al., 2010; Liu et al., 2011; Ritter et al., 2011; Li et al., 2012; Liu et al., 2012a; Fromreide et al., 2014), has demonstrated different usage of languages, entities, and the patterns of expressing relations, coreferences and events.

Regarding language, aside from English, many benchmarks, such as CoNLL, ACE, DEFT-ERE, and TAC-KBP, also comprise data for other languages, including German, Spanish, Russian, and Chinese. Several SIGHAN shared tasks have focused on Chinese NER (Zhang et al., 2006; Jin and Chen, 2008; He et al., 2012b; Zhu et al., 2003; Fang et al., 2004; Zhang et al., 2006), acknowledging the popularity of and special challenges posed by the Chinese language.

This dissertation recognizes variances in domain and language as an imperative and challenging problem in IE; the capability of high-quality extraction in domains and languages other than English news articles is indispensable. Therefore, we initiated two tasks, namely NER on Chinese social media (Chapters 3 and 4), and cross-sentence N-ary relation extraction in English Biomedical research publications (Chapter 5). We released data and code to promote research in related areas.

2.2 Machine Learning for Information Extraction

Research on information extraction usually involves designing machine learning algorithms to approach the problems. This section provides important background to understand the basics of machine learning (Section 2.2.1), and explains several machine learning models for NER and RE (Sections 2.2.2 and 2.2.3) upon which the later chapters of this dissertation build. Table 2.1 overviews the notation used throughout the thesis.

Notation	
$\boldsymbol{x} \in \mathbb{R}^n$	Input instance
$y \in \{1, 2, ..., k\}$	Output (multi-class)
$\boldsymbol{y} \in \mathcal{Y}$	Output (structured)
$h_\theta(\mathbf{x})$	Decision function
$\hat{\mathbf{y}}$	Prediction
$D = \{\boldsymbol{x}_d, \boldsymbol{y}_d\}_{d=1}^D$	Dataset
$\phi(\boldsymbol{x}, \boldsymbol{y})$	Feature function
$\lambda, \Phi, \Lambda, \theta, W$	Weight parameters
$\ell(\hat{\mathbf{y}}, \mathbf{y})$	Loss function

Table 2.1: Notational Overview.

2.2.1 Learning and Inference

A machine learning algorithm aims to **learn** a function $f(\boldsymbol{x}; W)$ that maps an input space $\mathcal{X}$ to an output space $\mathcal{Y}$. Depending on how a learning problem is formulated, the output variable can be categorical ($y \in \{1, 2, ..., k\}$, with binary as a special case), or structured ($\boldsymbol{y} = y_1, y_2, ..., y_n$, where each y_i is categorical, and y_is are usually interdependent, with $\boldsymbol{y} \in \mathcal{Y}$ denoting a set of feasible structures). The mapping function is usually parameterized with feature weights W. With recent advances in deep representation learn-

ing, some researchers also parameterize the feature function $\phi(\boldsymbol{x}, \boldsymbol{y})$ (or $\phi(\boldsymbol{x})$ as a special case) to automatically learn low-dimensional representations from data as features.

Machine learning assumes the data $D = \{\boldsymbol{x}^d, \boldsymbol{y}^d\}_{d=1}^D$ are drawn from some underlying distribution $P(\boldsymbol{x}, \boldsymbol{y})$, with each instance an independent and identically distributed (i.i.d.) sample. The goal of *supervised learning*[4] is to **learn** the parameters of the mapping function using training data $D = \{\boldsymbol{x}^d, \boldsymbol{y}^d\}_{d=1}^D$, to minimize the expected loss:

$$E[f] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(h_\theta(\boldsymbol{x}), \boldsymbol{y}) dP(\boldsymbol{x}, \boldsymbol{y}). \tag{2.1}$$

In practice, since the underlying distribution is usually unknown, the expected loss is approximated by the empirical loss:

$$J(D) = \frac{1}{D} \sum_{d=1}^{D} \ell(h_\theta(\boldsymbol{x}_d), \boldsymbol{y}_d) \tag{2.2}$$

The goal of learning is to find the parameters $\theta^* = \arg\min_\theta J(D)$, which is essentially an optimization problem. In NLP, the most popular optimization algorithms for learning the parameters are gradient-based methods, such as stochastic gradient descent (SGD) (Polyak and Juditsky, 1992), adaptive subgradient methods (Adagrad) (Duchi et al., 2011), and Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm (Byrd et al., 1995).

After learning the parameters, the unobserved variables (e.g., output variables) are estimated by **inference**. This is especially important in the structured case, because the

[4]We confine the discussion within the scope of supervised learning throughout the thesis.

output variables are interdependent and the set of possible output structures (the set $\mathcal{Y}$) is large. The two major types of inference are *maximum a posteriori (MAP) inference* and *marginal inference*. MAP inference computes the single highest- probability joint assignment of all unobserved variables:

$$\boldsymbol{y}^* = \arg\max S(\boldsymbol{x}, \boldsymbol{y}; W), \tag{2.3}$$

where $S(.)$ denotes a scoring function. Marginal inference computes the posterior marginal distributions of these variables.

There are many algorithms for inference, such as (loopy) belief propagation (Yedidia et al., 2003), variational Bayesian methods (Waterhouse et al., 1996), and Gibbs sampling (Gelfand and Smith, 1990) for marginal inference[5], and dual decomposition (Everett III, 1963), and integer linear programming (ILP) (Lenstra Jr, 1983) for MAP inference.

2.2.2 Log-linear Models

Since this dissertation focuses on the applications of named entity recognition and relation extraction, this section introduces two popular machine learning models for NER and RE. Both are log-linear models.

[5]The algorithms for marginal inference can usually be easily adapted for MAP inference.

2.2.2.1 Logistic Regression

The stand-alone relation extraction task is often formulated as a classification problem[6]. In this formulation, the boundaries and types of entities are given, and a classifier must decide whether the entities have a relation, and if so, which type.

Logistic regression is a discriminative classification model that defines the conditional probability of the output label y given the input $\boldsymbol{x}$ as:

$$p(y = k|\boldsymbol{x}; \theta) = \frac{\exp(\theta_k^T \phi(\boldsymbol{x}))}{Z(\boldsymbol{x}; \theta)}, \tag{2.4}$$

$$Z(\boldsymbol{x}; \theta) = \sum_{k=1}^{K} \exp(\theta_k^T \phi(\boldsymbol{x})), \tag{2.5}$$

where θ denotes the model parameters that can be learned, K denotes the number of classes, $\phi(\boldsymbol{x})$ denotes the feature function, and $Z(\boldsymbol{x}; \theta)$ is a normalization constant called the partition function. Since the logarithm of the conditional probability is an affine function with respect to the parameters θ, this is also called a log-linear model.

The parameters can be learned by maximizing the (log) conditional probability of the output labels $p(y_d|\boldsymbol{x}_d; \theta)$ of the training data:

$$\max_{\theta} \sum_{d=1}^{D} \log p(y_d|\boldsymbol{x}_d; \theta). \tag{2.6}$$

This objective function is convex, and can be solved by standard gradient-based optimization algorithms.

[6]In Chapter 6, We will discuss joint named entity recognition and relation extraction, which is a more complex structured prediction problem.

In Chapter 6, we combine logistic regression with deep neural network representation learning to conduct cross-sentence n-ary relation extraction.

2.2.2.2 Conditional Random Fields

Many IE problems including named entity recognition, coreference resolution, and event extraction, involve structured output in which the output variables are interdependent and form sequences, clusters, trees, or arbitrary graphs. NER is formulated as a sequence tagging problem: given an input sentence, the output is a sequence of labels, each associated with a word (or character in some cases). In one popular encoding schema, each tag indicates whether the tagged word or character is the **B**eginning, **I**nside, or **O**utside of an entity (the BIO tag schema). Another popular encoding strategy is to indicate whether it is the **B**eginning, **I**nside, **L**ast, **O**utside of an entity, or a **U**nit entity (the BILOU tag schema). The tag set can also be expanded by taking a Cartesian product with the types of entities (e.g., person, organization, or location).

Conditional random fields (CRFs) is a type of sequence tagging model that define the conditional probability of the output sequence $\boldsymbol{y}$ given the input $\boldsymbol{x}$ as:

$$p(\boldsymbol{y}|\boldsymbol{x};\theta) = \frac{\exp(\theta^T \phi(\boldsymbol{x},\boldsymbol{y}))}{Z(\boldsymbol{x};\theta)}, \tag{2.7}$$

$$Z(\boldsymbol{x};\theta) = \sum_{\boldsymbol{y}\in\mathcal{Y}} \exp(\theta^T \phi(\boldsymbol{x},\boldsymbol{y})), \tag{2.8}$$

where, again, θ denotes the model parameters that can be learned, $\phi(\boldsymbol{x},\boldsymbol{y})$ denotes the

feature function, and $Z(x;\theta)$ is the partition function. Equations 2.7 and 2.8 appear very similar to equations 2.4 and 2.5, because both logistic regression and CRFs are discriminative models that model the conditional probability. Specifically, they are both log-linear models. Their major differences are that the feature function of CRFs involves the output label $\boldsymbol{y}$, as the model capturing the inter-dependencies between adjacent labels; the partition function must sum over all possible output sequences. Marginal inference (e.g., the forward algorithm) is required to compute the partition function. At test time, MAP inference (e.g., the Viterbi algorithm) is required to compute the highest-probability output sequence given the input.

2.2.3 Deep Neural Networks

Recent advances in deep neural networks have revolutionized many NLP tasks, including machine translation, sentiment analysis, and sequence tagging problems including chunking, part-of-speech tagging and NER, and relation extraction. The major advantages of deep neural networks are the abilities to learn nonlinear mapping functions from inputs to outputs, and to automatically learn representations for characters, words, phrases, and sentences as features. This section introduces two basic building blocks of deep neural networks that are very prevalent in NLP: word embeddings and recurrent neural networks (RNNs).

2.2.3.1 Word Embeddings

One advantage of using deep neural networks for NLP is that they utilize lexical embeddings as input, rather than the original words and characters. Such embeddings were motivated by the linguistic theory of distributional semantics, according to which words appearing in similar contexts tend to have similar semantics. The NLP community has a long history of studying low-dimensional word representations, including matrix factorization algorithms (e.g., LSA, PCA, CCA, NNMF) over term frequency matrices (Hotelling, 1933; Thompson, 2005), Brown clustering (Brown et al., 1992), topic models (Hofmann, 1999; Blei et al., 2003; Mnih and Hinton, 2007) and vector space models (Salton et al., 1975) used in information retrieval. Recently, several popular algorithms have been proposed for deep neural networks to learn such embeddings (Bengio et al., 2006; Collobert and Weston, 2008; Turian et al., 2010; Mikolov et al., 2013; Pennington et al., 2014).

One popular neural network architecture for learning word embeddings is call the "skip-gram model" (Mikolov et al., 2013), which optimizes the log probability of the observed data:

$$\mathcal{L}(\mathcal{X}) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t), \tag{2.9}$$

where the conditional probability $p(w_{t+j}|w_t)$ is modeled as:

$$p(w_o|w_i) = \frac{\exp\left(\boldsymbol{e}_{wo}^T\boldsymbol{e}_{wi}\right)}{\sum_{w\in W}\exp\left(\boldsymbol{e}_{w}^T\boldsymbol{e}_{wi}\right)},$$

and where $\boldsymbol{e}_{wo}$ and $\boldsymbol{e}_{wi}$, respectively, denote the output and input embeddings for word w. W is the set of total words. We use this model in Chapters 3 and 4 to pre-train and jointly train with other IE tasks to produce word embeddings.

Word embeddings as features have proven helpful for many NLP tasks, such as NER (Miller et al., 2004; Turian et al., 2010), chunking (Turian et al., 2010), dependency parsing (Koo et al., 2008), semantic role labeling (Roth and Woodsend, 2014), and relation extraction (Sun et al., 2011; Plank and Moschitti, 2013; Nguyen and Grishman, 2015b). However, the most efficient way to use embeddings in NLP tasks is to combine them with recurrent neural networks to produce contextual representations, fine-tuning them for the specific task (Turian et al., 2009, 2010; Zhang et al., 2013b; Nguyen and Grishman, 2014; Roth and Woodsend, 2014; Nguyen et al., 2015; Cherry and Guo, 2015).

2.2.3.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) (Elman, 1990) is a family of neural networks in which the connections between units form a directed cycle. Their core is a recursively defined parameterized function that takes as inputs the current input x_t and the previous state s_{t-1}, and produces the current state s_t. There is usually also an output function that maps the current state to the current output vector. When given an input of a finite length sequence, an RNN can be unrolled into a chain structure. Figure 2.4 illustrates a basic unit

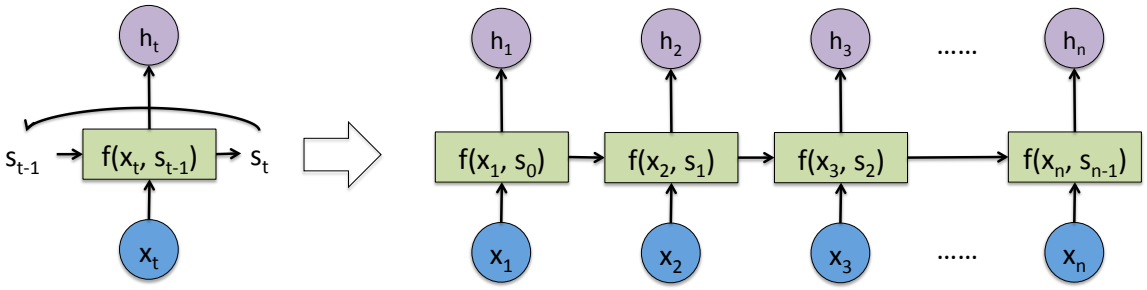


Figure 2.4: An illustration of recurrent neural networks (RNNs).

(left) and the corresponding unrolled structure (right) of an RNN.

Since RNNs enable the encoding of arbitrary length structured inputs into a fixed-length vector while considering the structured properties of the input, RNNs are suitable to model sequences.

The vanilla RNN defines the recurrent and output functions as:

$$s_t = R(x_t, s_{t-1}) = g(W^i x_t + W^h s_{t-1} + b), \tag{2.10}$$

$$h_t = O(s_t) = s_t, \tag{2.11}$$

where W^i, W^h, b are the model parameters. $W^i \in \mathbb{R}^{d \times h}, W^h \in \mathbb{R}^{h \times h}$, and $b \in \mathbb{R}^h$, where d and h are the dimensions for the input embeddings and the output vectors, respectively. $g(.)$ denotes the activation function, which in this case is the sigmoid function. The vanilla RNNs suffer from gradient diffusion or explosion, where error signals (gradients) in later steps in the sequence diminish or explode in the back-propagation process, and cannot reach earlier inputs, making training very difficult (Bengio et al., 1994; Pascanu et al., 2013).

Long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) have been proposed to combat the problem of vanishing or exploding gradients. This special type of RNNs uses a series of gates (input, forget, and output gates) to control the flow of information in the hidden states of the model, thus avoiding amplifying or suppressing gradients during back-propagation. Equations 2.12 to 2.17 define the basic recurrent unit in an LSTM, where Ws $\in \mathbb{R}^{d\times h}$, Us $\in \mathbb{R}^{h\times h}$, and bs $\in \mathbb{R}^{h}$ are the model parameters; $i_t, f_t, o_t \in \mathbb{R}^h$ are the input, forget, and output gates, respectively, controling the information flow; and $\odot$ denotes element-wise multiplication. Following the notation above, $s_t = [c_t; h_t]$, where $[;]$ denotes concatenation.

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \tag{2.12}$$

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \tag{2.13}$$

$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \tag{2.14}$$

$$\tilde{c_t} = \tanh(W^c x_t + U^c h_{t-1} + b^c) \tag{2.15}$$

$$c_t = i_t \odot \tilde{c_t} + f_t \odot c_{t-1} \tag{2.16}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.17}$$

LSTMs are currently the most successful type of RNNs for NLP applications. The models in this dissertation which involve RNNs are all the LSTM variant.

There are many other variants of RNNs. The gated recurrent unit (GRU) (Cho et al.,

2014) is one of the most significant variants, shown to achieve comparable results to LSTMs on several (non-textual) datasets (Chung et al., 2014). Like LSTMs, the GRU is also based on gating mechanisms, but has substantially fewer gates and does not have a separate memory component. Equations 2.18 to 2.21 define the basic recurrent unit for a GRU, where $W\text{s} \in \mathbb{R}^{d \times h}$ and $U\text{s} \in \mathbb{R}^{h \times h}$ are model parameters; r_t, z_t are reset and update gates, respectively, controlling information flow; and $s_t = h_t$, following the notation above.

$$r_t = \sigma(W^r x_t + U^r h_{t-1}) \tag{2.18}$$

$$z_t = \sigma(W^z x_t + U^z h_{t-1}) \tag{2.19}$$

$$\tilde{h_t} = \tanh\left(W^h x_t + U^h(r_t \odot h_{t-1})\right) \tag{2.20}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h_t} \tag{2.21}$$

2.3 Learning Under Low-Resource Conditions

While machine learning models have achieved notable success in NLP and IE applications, most have assumed high-resource, fully supervised conditions. That is, the training and test data are drawn from the same distribution; high-quality tools are available for the pre-processing pipeline; and there are adequate labeled data to train good models. These assumptions are too strict for real-world applications.

Language is widely recognized to have domains. Social media posts, biomedical publications, legal provisions, and online reviews all form different domains, each associated with different distributions of words (both word frequencies and word semantic distributions). Moreover, different tasks are proposed to achieve different goals, and there are 6909 living languages in the world. It would be impossible to collect adequate annotations to train fully supervised models for each domain, task, and language.

I recognized the problem of limited annotation as the next significant challenge in natural language processing and information extraction, and proposed to leverage smart supervision signals (Chapter 5) and transfer knowledge from other domains and tasks using representation learning (Chapter 3 and 4) in order to combat the problem of inadequate supervision.

This section introduces several common techniques typically employed when there is inadequate training data, including distant supervision, multi-domain learning, and multi-task learning.

2.3.1 Distant Supervision

Distant supervision is a family of algorithms that rely on available knowledge bases or (semi-)structured data, such as Wikipedia, to *automatically* obtain *noisy* annotations for training, thereby obtaining smart supervision signals with little human annotative effort.

Mintz et al. (2009) first coined the name "distant supervision" for relation extraction, but the same idea has been explored in slightly different forms for entity and relation extraction (Craven et al., 1998, 2000; Morgan et al., 2004; Snow et al., 2005; Wu and Weld, 2007).

These algorithms involve noisy matching of knowledge base entries in the raw text to compose training data. Because the facts (entries) in the knowledge bases often form a small set, the training instances are usually redundant (multiple instances discussing the same fact). Strategies such as anonymizing the matched fact mentions are usually employed to avoid overfitting. Hoffmann et al. (2011); Surdeanu et al. (2012) also explored using the traits of multiple instances to combat noise in the training data.

More recently, distant supervision has been extended to capture implicit reasoning via matrix factorization (Riedel et al., 2013) or knowledge base embedding (Toutanova et al., 2015, 2016).

2.3.2 Multi-domain Learning

Multi-domain learning and its special case domain adaptation aim to learn a *single* model for data from different domains to help the domains with less annotated data. It focuses on handling *changes in data distributions among domains*, such as different word usage and semantics. Formally, multi-domain learning aims at modeling the changes in $p(\mathbf{x})$, or termed as covariate shift.

Some domains have more resources and annotated data, often termed the source domain, while other domains have fewer annotations, often termed the target domain. Multi-domain learning leverages the annotations in source domains to facilitate the predictions in target domains, thus mitigating the problem of inadequate annotations.

In NLP, multi-domain learning is very practical because the available resources and annotations are mostly in the domain of English news. Many applications suffer significant drops in performance when applied to new domains, such as social media, discussion forums, or biomedicine.

Work on domain adaptation mostly follows two approaches: 1). parameter tying, or linking similar features during learning (Dredze and Crammer, 2008; Daumé III, 2009b,a; Finkel and Manning, 2009; Kumar et al., 2010; Dredze et al., 2010a) and 2). learning cross domain representations (Blitzer et al., 2006, 2007; Glorot et al., 2011; Chen et al., 2012; Yang and Eisenstein, 2015), where representations can arise from feature design or be learned directly from data.

Supervision signals for domain adaptation may be either unsupervised (Blitzer et al., 2006) or supervised (Daumé III, 2009b), depending on whether or not some training data exists in the target domain. Unsupervised methods attempt to learn the shift of data distribution from a vast amount of unlabeled data taken from the corresponding domains, while supervised methods better calibrate the shift of data distribution for a specific task

by leveraging few annotations in the target domain.

2.3.3 Multi-task Learning

Multi-task learning (Caruana, 1993, 1997) is not specifically designed to handle the problem of limited annotation in machine learning. Rather, it has been proposed to jointly learn several tasks to yield better inductive bias and avoid over-fitting, thereby improving the performance of each task (Ando and Zhang, 2005; Collobert et al., 2011; Setiawan et al., 2015; Pham et al., 2015; Søgaard and Goldberg, 2016a; Liu et al., 2016a, 2017; Pasunuru and Bansal, 2017; Rei, 2017; Peng et al., 2017). Formally, multi-task learning *jointly* learn several models for different tasks. The different tasks associate with a change in $p(\boldsymbol{y}|\boldsymbol{x})$, where the set $\mathcal{Y}$ of possible outputs changes.

Multi-task learning has been shown to efficiently handle the inadequate annotation problem for many applications including classification tasks (Vijayaraghavan et al., 2017; Augenstein and Søgaard, 2017), sequence tagging tasks (Cheng et al., 2015; Peng and Dredze, 2016; Yang et al., 2016), machine translation (Shah and Specia, 2016), among others (Cdesouza de Souza et al., 2015; Cummins et al., 2016; Liu et al., 2016c), especially when combined with representation learning. There are several reasons for this:

First, multi-task learning implicitly augments the training data. If the datasets are disjoint for different tasks, multi-task learning implicitly pools the datasets for different

but related tasks to learn. Second, machine learning algorithms introduce inductive bias to reduce the assumption space and efficiently solve the problems. Multi-task learning promotes the inductive bias that prefer robust feature representations with better generalizability to different tasks. This helps the learned model generalize to new tasks and unseen data, a trait which suits the scenario of limited annotation. Moreover, in multi-task learning, each task acts as a regularizer to avoid over-fitting to training data. With few annotations, over-fitting is a serious concern.

Work on multi-task learning has mostly focused on learning predictive feature representations and sharing them among different tasks. A transformation is learned to produce the feature representation (Ando and Zhang, 2005). recently, multi-task learning has been extensively explored under the deep learning (feature learning) framework. There are two major methods for multi-task learning with deep learning: 1). hard parameter tying, which simply shares some layers of neural networks among tasks and 2). soft parameter regularization, which regularizes the parameters for learning representations of different task to be similar.

wiwit

Chapter 3

Multi-task Representation Learning

A Case Study on Chinese Social Media NER

This chapter investigates the efficiency of multi-task representation learning for low-resource IE. We use Chinese social media entity mention detection [1] as a case study, which is a typical case of low-resource IE. This chapter is based on materials from Peng and Dredze (2015) and Peng and Dredze (2016).

The chapter progressively explores two multi-task learning strategies. The first one explores multi-task learning with *unlabeled data* and language modeling to help Chinese social media NER (Section 3.3). This is an extension of the widely used fine-tuning strat-

[1] We sometimes use the term NER for simplicity.

egy, which *initializes* the embeddings using a pre-trained model (usually language model) on unlabeled data, and then adjust them according to NER annotated data (Collobert et al., 2011; Zheng et al., 2013a; Yao et al., 2014; Pei et al., 2014). The multi-task learning strategy improves over fine-tuning since it dynamically adjust the embeddings for all the words even if they have not appeared in the NER annotated data. This is especially helpful for dealing with out-of-vocabulary words (OOVs) during test time.

The second part goes one step further to conduct multi-task learning of Chinese NER and Chinese word segmentation (Section 3.4). These are two highly-correlated tasks. Since the two tasks can both be formulated as sequence tagging problems and use RNNs to learn feature representations, they can efficiently share the "contextual representations" produced by RNNs while conduct multi-task learning. The empirical results showed that sharing higher-level contextual representations is more efficient than just sharing lower-level word embeddings.

Section 3.5 then discusses about the improvements of data quality after the release of our dataset. We provide the performances of our methods on the quality-improved dataset in comparison to several other work that used our data (the improved version).

The contributions of this chapter including:

- We initiated the task of Chinese social media NER, which is a typical case of low-resource IE. We composed and released the dataset to facilitate research in the re-

lated fields.

- We evaluated three types of embeddings for representing Chinese text and using as inputs for Chinese NER, and found the character-positional embedding proposed to be the most efficient one.

- We proposed and empirical evaluated two multi-task learning strategies for Chinese social media NER, which are both proven efficient to help the NER task.

- Our results reveal that the gap between social media and traditional text for Chinese is much larger than similar corpora for English, suggesting this task as an interesting area of future work

This chapter is organized into three main parts. The first part (Section 3.1 and 3.2) introduces the task of Chinese social media NER and the weibo dataset. The second and third parts investigate the two multi-task learning strategies respectively.

3.1 NER for Chinese Social Media

Named entity recognition, or more generally entity mention detection, is an essential component of the IE pipeline. As is discussed in Section 2.1.2, the long line of work in Chinese NER has focused on formal domains, and NER for social media has been largely

restricted to English. In this chapter, we expand the scope to NER on Chinese social media from the popular Sina Weibo service. There are four major reasons make this an important and challenging problem. First, Sina Weibo is the most popular microblog service (comparable in size to Twitter and previously used in NLP research (Ling et al., 2013)) for Chinese language users. There are abundant timely information carried in the Weibo messages. Second, there are special challenges faced in processing Chinese language data since it uses logograms instead of alphabets, and lacks many of the clues that a word is a name, e.g. capitalization and punctuation marks, etc. The lack of explicit word boundaries further confuses NER systems. Moreover, as is the case for other languages, social media informality introduces numerous problems for NLP systems, such as spelling errors, novel words, and ungrammatical constructions. Last but not the least, it is a low-resource setting where neither enough annotated data nor high-quality NLP pre-processing tools are available for building NER systems. Figure 3.1 shows some examples that demonstrate the challenges.

The baseline system for our task is our own implementation of Mao et al. (2008), which is the current state-of-the-art on the SIGHAN 2008 shared task (Jin and Chen, 2008). They use a CRF tagger with a BIOSE (begin, inside, outside, singleton, end) encoding that tags individual characters, not words, since word segmentation errors are especially problematic for NER (Zhang et al., 2006). Features include many common English NER

有好多好多的话想对你说李巾凡想要瘦瘦瘦成李帆我是想切开云朵的心

Have many many words to say to you Jinfan Li wanna thin thin thin to Fan Li I am a heart that want to cut the cloud

美得呀～顾天池苦逼青年杨素晗闵日记肖立伟嘻嘻嘻嘻嘻嘻美啊

Beautiful Tianchi Gu bitter youth Suhan Yang Riji Min Liwei Xiao hahahahahaha beautiful

看见前女友和她的新欢走在一起的时候，已经无处可躲了，只好硬着 头皮上去打招呼哎呀，好久不见，你儿子都这么高了。

When saw ex-girl friend and her new partner coming across, nowhere to hide, have to say hello, long time no see, your son grown up.

Figure 3.1: Examples of Weibos messages and translations with named (red) and nominal (blue) mentions.

features, e.g. character unigrams and bigrams, with context windows of size 5. See Mao et al. (2008) for complete details on their system.

Mao et al. (2008) use a two pass approach, training a CRF first for mention detection and using the resulting predictions as a feature for an NER system. Furthermore, they make extensive use of gazetteer features. For simplicity, we exclude the first pass mention detection and the gazetteer features, which make only small improvements to their overall performance. More specifically we used the following feature templates:

Unigram: $C_n(n = -2, -1, 0, 1, 2)$,

Bigram: $C_nC_{n+1}(n = -2, -1, 0, 1)$ and $C_{-1}C_1$,

where C_0 is the current character, C_1 the next character, C_2 the second character after C_0,

C_{-1} the character preceding C_0, and C_{-2} the second character before C_0. We note that other implementations of this system (Zhang et al., 2013b) have been unable to match the performance reported in Mao et al. (2008). Similarly, our implementation yields results on SIGHAN 2008 similar to those reported in Zhang et al. (2013b).[2] Overall, we take this tagger as representative of state-of-the-art for Chinese NER.

3.2 Weibo NER Corpus

To facilitate research of the important and challenging problem of Chinese social media NER, we constructed a corpus of Weibo messages annotated for NER. We followed the DEFT ERE (Linguistics Data Consortium, 2014) [3] annotation guidelines for entities, which includes four major semantic types: person, organization, location and geo-political entity. We annotated both name and nominal mentions. Chinese pronoun mentions can be easily recognized with a regular expression. We used Amazon Mechanical Turk, using standard methods of multiple annotators and including gold examples to ensure high quality annotations (Callison-Burch and Dredze, 2010).

Our corpus includes 1,890 messages sampled from Weibo between November 2013 and December 2014. Rather than selecting messages at random, which would yield a

[2]Our implementation obtains an F1 of 88.63%.

[3]See Aguilar et al. (2014) for a comparison of DEFT ERE with other common standards.

	Mentions		
Entity Type	**Name**	**Nominal**	**Total**
Geo-political	243	0	243
Location	88	38	126
Organization	224	31	255
Person	721	636	1,357

Table 3.1: Mention statistics for the Weibo NER corpus.

small number of messages with entities, we selected messages that contained three or more (segmented) words that were not in a fixed vocabulary of common Chinese words. Initial experiments showed this gave messages more likely to contain entities.

Table 3.1 shows statistics of the final corpus. We divided the corpus into 7 folds, each with 127 messages, where each message corresponds to a single instance. We use the first 5 folds for train, the 6th for development, and the 7th for test. The annotated corpus is made publicly available.[4]

[4]https://github.com/hltcoe/golden-horse

3.3 Joint Learning of Embeddings and NER

3.3.1 Embeddings for Chinese

Lexical embeddings represent words in a continuous low dimensional space, which can capture semantic or syntactic properties of the lexicon: similar words would have similar low dimensional vector representations. Embeddings have been used to gain improvements in a variety of NLP tasks. In NER specifically, several papers have shown improvements by using pre-trained neural embeddings as features in standard NER systems (Collobert and Weston, 2008; Turian et al., 2010; Passos et al., 2014). More recently, these improvements have been demonstrated on Twitter data (Cherry and Guo, 2015). Embeddings are especially helpful when there is little training data, since they can be trained on a large amount of unlabeled data. This is the case for new languages and domains, the task we face in this Chapter.

However, training embeddings for Chinese is not straightforward: Chinese is not word segmented, so embeddings for each word cannot be trained on a raw corpus. Additionally, the state-of-the-art systems for downstream Chinese tasks, such as NER, may not use words.

We present three types of Chinese embeddings that will be trained on a large corpus of Weibo messages. These embeddings will be used as features in the NER system by adding

a (real valued) feature for each dimension of the embedding for the current word/character.

Word Embeddings We train an embedding for each word type, the standard approach in other languages. We run a Chinese word segmentation system[5] over the raw corpus of Weibo messages. To create features, we first segment the NER data, and then lookup the embedding that matches the segmented word. Since the NER system tags characters, we add the same word embedding features to each character in the word.

Character Embeddings We learn an embedding for each character in the training corpus (Sun et al., 2014; Liu et al., 2014).This removes the dependency on pre-processing the text, and better fits our intended use case: NER tagging over characters. Since there are many fewer characters than words, we learn many fewer embeddings. On the one hand, this means fewer parameters and less over-fitting. However, the reduction in parameters comes with a loss of specificity, where we may be unable to learn different behaviors of a character in different settings. We explore a compromise approach in the next section. These embeddings are directly incorporated into the NER system by adding embedding features for each character.

Character and Position Embeddings Character embeddings cannot distinguish between uses of the same character in different contexts, whereas word embeddings fail to make use of characters or character n-grams that are part of many words. A compromise is to use character embeddings that are sensitive to the character's position in the word

[5]We use Jieba for segmentation: `https://github.com/fxsjy/jieba`

(Chen et al., 2015b). We first word segment the corpus. For each character in each word, we add a positional tag, e.g. the first/second/etc. character in the word, yielding multiple embeddings per character. We learn separate embeddings for each positionally tagged character. To use these embeddings as features, we segment the NER text, obtain position tags for each character, and add features for the corresponding embedding.

These three methods lead to 179,809 word embeddings, 10,912 character embeddings, and 24,818 character with position embeddings.

3.3.2 Fine Tuning

For each of the embeddings, we fine-tune pre-trained embeddings in the context of the NER task. This corresponds to initializing the embeddings parameters using a pre-trained model, and then modifying the parameters during gradient updates of the NER model by back-propogating gradients. This is a standard method that has been previously explored in sequential and structured prediction problem (Collobert et al., 2011; Zheng et al., 2013a; Yao et al., 2014; Pei et al., 2014).

3.3.3 Joint Neural-CRF NER and Language Modeling

Fine-tuning has a disadvantage: it can arbitrarily deviate from the settings obtained from training on large amounts of raw text. Recent work has instead tuned embeddings

for a specific task, while maintaining information learned from raw text. Yu and Dredze (2014) use multi-part objectives that include both standard unlabeled objectives, such as skip-gram models in word2vec, and task specific objectives. Jointly training the embeddings with the multi-part objectives allows the fine-tuned embeddings to further influence other embeddings, even those that do not appear in the labeled training data. This type of training can help improve OOVs (Yu and Dredze, 2015), an important aspect of improving social media NER.

We propose to jointly learn embeddings for both language models and the NER task. The modified objective function (log-likelihood) for the CRF is given by:

$$\begin{aligned}&\mathcal{L}_s(\boldsymbol{\lambda}, \boldsymbol{e_w}) \\ =&\frac{1}{K}\sum_k \left[\log \frac{1}{Z(x)^k} + \sum_j \lambda_j F_j(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{e_w})\right],\end{aligned}$$

where K denotes the number of instances, $\boldsymbol{\lambda}$ represents the weight vector, $\boldsymbol{x}^k$ and $\boldsymbol{y}^k$ are the words and labels sequence for each instance, $\boldsymbol{e_w}$ is the embedding for a word/character/character-position representation w, $Z(x)^k$ is the normalization factor for each instance, and $F_j(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{e_w}) = \sum_{i=1}^{n} f_j(y_{i-1}^k, y_i^k, \boldsymbol{x}^k, \boldsymbol{e_w}, i)$ represents the feature function in which j denotes different feature templates and i denotes the position index in a sentence. The feature template generate m dimensional lexical features at each position i, and the feature function concatenate them with the lexical embedding of character i to generate the final feature. We used the feature template introduced on Section 3.1. This differs from a traditional CRF in

that the feature function depends on the additional variables $\boldsymbol{e_w}$, which are the embeddings (as defined above). As a result, the objective is no longer log-linear, but log-bilinear [6].

The second term is the standard skip-gram language model objective (Mikolov et al., 2013):

$$\mathcal{L}_u(\boldsymbol{e_w}) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t), \tag{3.1}$$

where

$$p(w_i|w_j) = \frac{\exp\left(\boldsymbol{e}_{\boldsymbol{w_i}}^T \boldsymbol{e}_{\boldsymbol{w_j}}\right)}{\sum_{i'} \exp\left(\boldsymbol{e}_{\boldsymbol{w_{i'}}}^T \boldsymbol{e}_{\boldsymbol{w_j}}\right)}.$$

The first objective is notated $\mathcal{L}_s$ for "supervised" (trained on labeled NER data), and the second is $\mathcal{L}_u$, "unsupervised" (trained on raw text.) Both objectives share the same variables $\boldsymbol{e_w}$. The overall goal is to maximize their weighted sum:

$$\arg\max_{e_w} = \mathcal{L}_s(\boldsymbol{\lambda}, \boldsymbol{e_w}) + C\mathcal{L}_u(\boldsymbol{e_w}) \tag{3.2}$$

where C is a tradeoff parameter.

3.3.4 Parameter Estimation

We pre-trained embeddings using word2vec (Mikolov et al., 2013) with the skip-gram training objective and NEC negative sampling. We constructed a corpus of unlabeled mes-

[6] It is log-bilinear because the log-likelihood takes the form $f(x, y) = axy + bx + cy$, where x, y are variables and a, b, c are coefficients. In this case, x is the feature weight and y is the embedding; both of them are vectors. Taking the partial derivative with respect to any one of the variables, one gets a constant (wrt that variable). This satisfies the definition of log-bilinear functions.

sages for training the embeddings. We randomly selected 2,259,434 messages from the same time period when the NER annotated data are selected. Unless otherwise stated, we used word2vec's default parameter settings. All embeddings were 100-dimensional, and we used the same embeddings for the input and output parameters in the skip-gram objective. We optimized the joint objective (3.2) using an alternative optimization strategy: we alternated 30 iterations of CRF training on the NE labeled data and 5 multi-threaded passes through both the labeled and unlabeled data for the skip-gram objective. We avoided overfitting using early-stopping. For simplicity, we set $C = 1$ for (3.2). The CRF was trained using stochastic gradient descent with an L2 regularizer. All model hyper-parameters were tuned on dev data.

We use the off-the-shelf tool word2vec to do skip-gram training for language model, and implement our own CRF model to modify the embeddings. We optimize (3.2) by alternating the optimzation of each of the two objectives.

3.3.5 Experiments

We evaluate our methods under two settings: training on only name mentions, and training on both name and nominal mentions. We re-train the Stanford NER system (Finkel et al., 2005) as a baseline; besides, we also evaluate our implementation of the CRF from Mao et al. (2008) as described in Section 3.1 as *Baseline Features*. To this baseline,

	Dev						Test					
Method	**Without Fine Tuning**			**With Fine Tuning**			**Without Fine Tuning**			**With Fine Tuning**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Stanford	63.51	23.27	34.06	**N/A**			55.70	22.86	33.06	**N/A**		
Baseline Features	63.51	27.17	38.06	**N/A**			56.98	25.26	35.00	**N/A**		
+ word	65.71	26.59	37.86	70.97	25.43	37.45	56.82	25.77	35.46	64.94	25.77	36.90
+ character	53.54	30.64	38.97	58.76	32.95	42.22	56.48	31.44	40.40	57.89	34.02	42.86
+ character+position	60.87	32.37	42.26	61.76	36.42	**45.82**	61.90	33.51	43.48	57.26	34.53	43.09
Joint (cp)	**N/A**			57.41	35.84	44.13	**N/A**			57.98	35.57	**44.09**
Stanford	72.39	31.80	44.19	**N/A**			63.96	22.19	32.95	**N/A**		
Baseline Features	71.94	33.22	45.45	**N/A**			60.16	23.87	34.18	**N/A**		
+ word	69.66	33.55	45.29	70.67	35.22	47.01	59.40	25.48	35.67	60.68	22.90	33.26
+ character	58.76	32.95	42.22	66.88	35.55	46.42	58.28	28.39	38.18	55.15	29.35	38.32
+ character+position	73.43	34.88	47.30	69.38	36.88	48.16	65.91	28.06	39.37	62.33	29.35	39.91
Joint (cp)	**N/A**			72.55	36.88	**48.90**	**N/A**			63.84	29.45	**40.38**

Table 3.2: NER results for name mentions (top) and name + nominal mentions (bottom).

we add each of our three embedding models: *word*, *character*, *character+position* (as described in Section 3.3.1), and report results on the modified CRF model with and without fine-tuning. We also report results for the joint method trained with the character+position model (cp), which performed the best on dev data for joint training.

3.3.5.1 General Results

Table 3.2 shows results for both dev (tuned) and test (held out) splits. First, we observe that the results for the baseline are significantly below those for SIGHAN shared tasks as well as the reported results on Twitter NER, showing the difficulty of this task. In particular, recall is especially challenging. Second, all embeddings improve the baseline on test data, but the character + position model gets the best results. Fine-tuning improves embedding results, but seems to overfit on dev data. Finally, our joint model does the best in both conditions (name and name+nominal) on test data, improving over fine-tuning, yielding up to a 9% (absolute) improvement over a strong baseline.

3.3.5.2 Effect of Embeddings

We expect improvements from embeddings to be larger when there is less training data. Figure 3.2 shows F1 on dev data for different amounts of training data, from 200 instances up to 1400, for the character + position embeddings versus the baseline model. We see that for both settings, we see larger improvements from embeddings for smaller training sets.

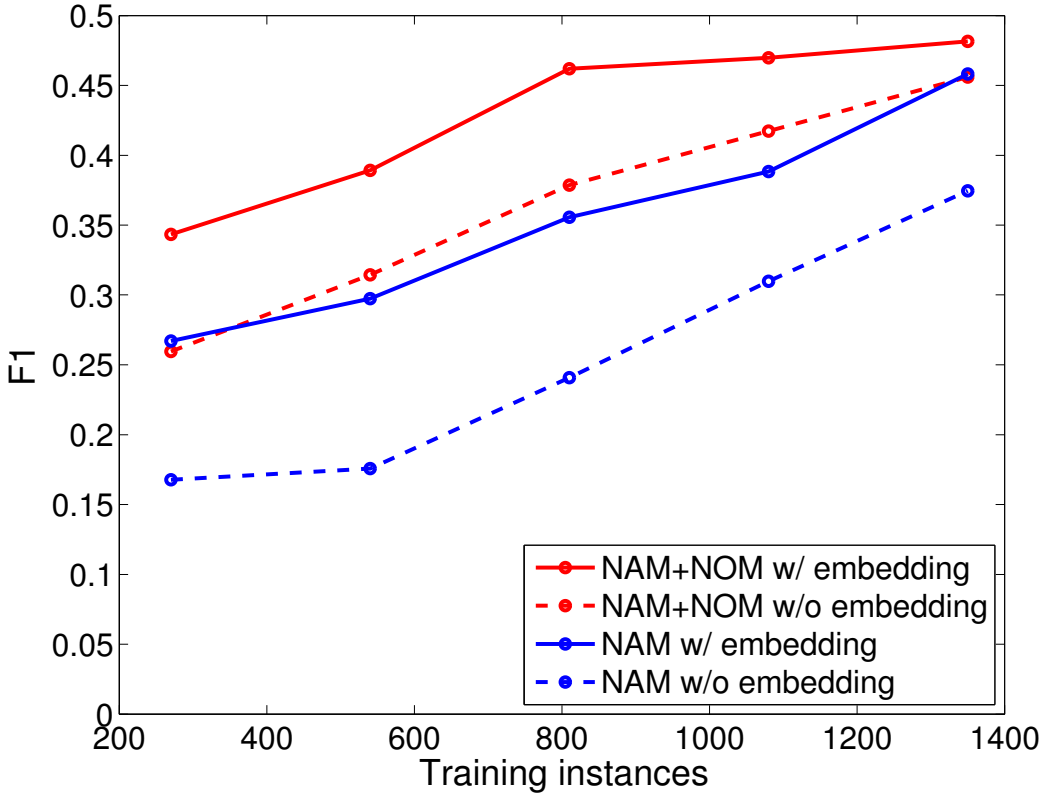


Figure 3.2: Dev F1 for varying number of training instances.

3.3.5.3 Error Analysis

Since the results are relatively low, we conducted an error analysis by randomly sampling 150 error items and manually looking through them. Among the 150 examples, 65 are annotation errors, majorly cause by annotators neglecting some mentions, this contributes 43% of the errors. The second largest error source are the person names: Chinese person names are very flexible and nearly every character can be used in given names, this

makes recognizing person names challenging and contributes to 9% of our errors. The following largest source of error are transliterated foreign names, which contributes to 7% of the errors. Other sources including boundary error, type error, name abbreviation, nicknames, etc.

3.4 Multi-task Learning of Chinese Word Segmentation and NER

The fact that character-positional embeddings performed better than character embeddings when we jointly learned the embeddings with unlabeled data and language modeling, indicates that word segmentation information is essential for NER. Therefore, in this section, we investigate better ways to incorporate word boundary information into an NER system for Chinese social media. We combine the state-of-the-art Chinese word segmentation system (Chen et al., 2015a) with the NER model we introduced in the last section. The proposed multi-task learning strategy allows for jointly training *contextual* representations, providing more information to the NER system about hidden representations learned from word segmentation, as compared to features based on segmentation output.

As will be shown in the experimental section, the integrated model sharing higher-level information achieves nearly a 5% absolute improvement over the multi-task learning

strategy proposed in Section 3.3.

3.4.1 LSTM for Word Segmentation

Chen et al. (2015a) proposed a single layer, left to right LSTM for Chinese word segmentation. As is introduced in Section 2.2.3.2, an LSTM is a recurrent neural network (RNN) which uses a series of gates (input, forget and output gate) to control how memory is propagated in the hidden states of the model. For the Chinese word segmentation task, each Chinese character is initialized as a d dimensional vector, which the LSTM will modify during its training. For each input character, the model learns a hidden vector $\boldsymbol{h}$. These vectors are then used with a biased-linear transformation to predict the output labels, which in this case are **B**egin, **I**nside, **E**nd, and **S**ingleton. A prediction for position t is given as:

$$y^{(t)} = W_o h^{(t)} + b_o \tag{3.3}$$

where W_o is a matrix for the transformation parameters, b_o is a vector for the bias parameters, and $h^{(t)}$ is the hidden vector at position t. To model the tag dependencies, they introduced the transition score A_{ij} to measure the probability of jumping from tag $i \in T$ to tag $j \in T$.

We used the same model as Chen et al. (2015a) trained on the same data (segmented Chinese news article). However, we employed a different training objective. Chen et al.

(2015a) employed a max-margin objective, however, while they found this objective yielded better results, we observed that maximum-likelihood yielded better segmentation results in our experiments[7]. Additionally, we sought to integrate their model with a log-bilinear CRF, which uses a maximum-likelihood training objective. For consistency, we trained the LSTM with a maximum-likelihood training objective as well. The maximum-likelihood CRF objective function for predicting segmentations is:

$$\mathcal{L}_s(\boldsymbol{y}_s; \boldsymbol{x}_s, \Theta) = \frac{1}{K} \sum_k \big[\log \frac{1}{Z(\boldsymbol{x}_s)^k} + \sum_i \big(T_s(y^k_{i-1}, y^k_i) + s(y^k_i; \boldsymbol{x}^k_s, \boldsymbol{\Lambda}_s) \big) \big] \tag{3.4}$$

Example pairs $(\boldsymbol{y}_s, \boldsymbol{x}_s)$ are word segmented sentences, k indexes examples, and i indexes positions in examples. $T_s(y^k_{i-1}, y^k_i)$ are standard transition probabilities learned by the CRF[8]. The LSTM parameters $\boldsymbol{\Lambda}_s$ are used to produce $s(y^k_i; \boldsymbol{x}^k_s, \Lambda_s)$, the emission probability of the label at position i for input sentence k, which is obtained by taking a soft-max over (3.3). We use a first-order Markov model.

[7]Chen et al. (2015a) preprocessed the data specifically for Chinese word segmentation, such as replacing English characters, symbols, dates and Chinese idioms as special symbols. Our implementation discarded all these preprocessing steps, which while it achieved nearly identical results on development data (as inferred from their published figure), it lagged in test accuracy by 2.4%. However, we found that while these preprocessing steps improved segmentation, they hurt NER results as they resulted in a mis-match between the segmentation and NER input data. Since our focus is on improving NER, we do not use their preprocessing steps in our experiments.

[8]The same functionality as A_{ij} in the model of Chen et al. (2015a).

3.4.2 Log-bilinear CRF for NER

In Section 3.3, we proposed a log-bilinear model for Chinese social media NER. The model used standard NER features along with additional features based on lexical embeddings. By jointly training the embeddings with a word2vec objective, the resulting model is log-bilinear.

In this section, we apply the same idea but go one step further to explore augmenting the traditional CRF sequence tagging model with LSTM learned representations. We enable interaction between the CRF and the LSTM parameters. More details are described in (Section 3.4.3).

3.4.3 Using Segmentation Representations to Improve NER

The improvements provided by character position embeddings demonstrated in Section 3.3.5 indicated that word segmentation information can be helpful for NER. Embeddings aside, a simple way to include this information in an NER system would be to add features to the CRF using the predicted segmentation labels as features.

However, these features alone may overlook useful information from the segmentation model. Previous work showed that jointly learning different stages of the NLP pipeline helped for Chinese (Liu et al., 2012b; Zheng et al., 2013b). We thus seek approaches for deeper interaction between word segmentation and NER models. The LSTM word seg-

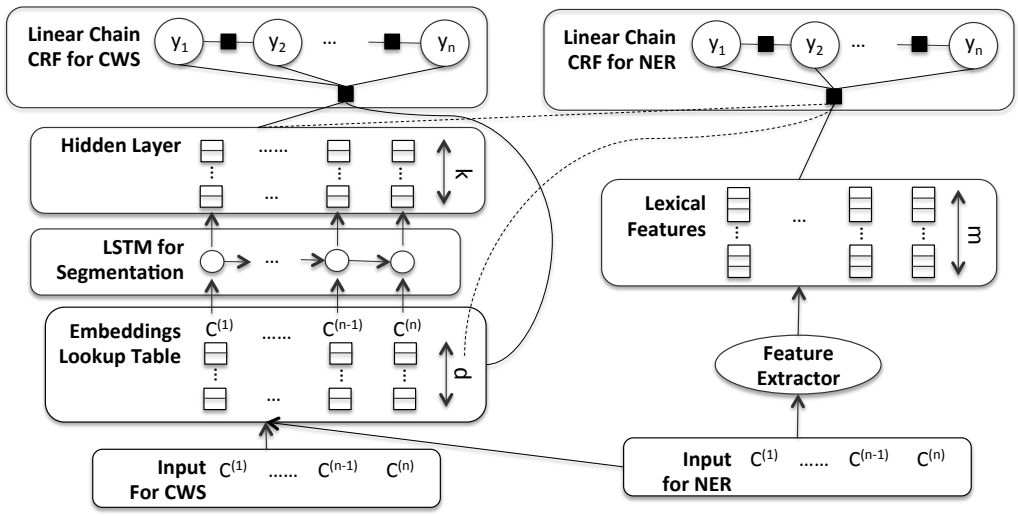


Figure 3.3: The joint model for Chinese word segmentation and NER.

mentor learns two different types of representations: 1) embeddings for each character and 2) hidden vectors for predicting segmentation tags. Compressing these rich representations down to a small feature set imposes a bottleneck on using richer word segmentation related information for NER. We thus experiment with including both of these information sources directly into the NER model.

Since the log-bilinear CRF already supports joint training of lexical embeddings, we can also incorporate the LSTM output hidden vectors as dynamic features using a joint objective function.

First, we augment the CRF with the LSTM parameters as follows:

$$\mathcal{L}_n(\boldsymbol{y}_n; \boldsymbol{x}_n, \Theta) = \frac{1}{K} \sum_k \big[\log \frac{1}{Z(\boldsymbol{x}_n)^k} + \sum_j \Lambda_j F_j(\boldsymbol{y}_n^k, \boldsymbol{x}_n^k, \boldsymbol{e}_w, \boldsymbol{h}_w) \big], \tag{3.5}$$

where k indexes instances, j positions, and

$$F_j(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{e_w}, \boldsymbol{h_w}) = \sum_{i=1}^{n} f_j(y_{i-1}^k, y_i^k, \boldsymbol{x}^k, \boldsymbol{e}_w, \boldsymbol{h}_w, i)$$

represents the feature functions. These features now depend on the embeddings learned by the LSTM ($\boldsymbol{e}_w$) and the LSTM's output hidden vectors ($\boldsymbol{h}_w$). Note that by including $\boldsymbol{h}_w$ alone we create dependence on all LSTM parameters on which the hidden states depend (i.e. the weight matrices). We experiment with including input embeddings and output hidden vectors independently, as well as both parameters together.

An illustration of the integrated model is shown in Figure 3.3. The left hand side is an LSTM module for word segmentation, and the right hand side is a traditional feature-based CRF model for NER. Note that the linear chain CRF for NER has both access to the feature extractor specifically for NER and the representations produced by the LSTM module for word segmentation. The CRF in this version is a log-bilinear CRF, where it treats the embeddings and hidden vectors inputs as variables and modifies them according to the objective function. As a result, it enables propagating the gradients back into the LSTM to adjust the parameters. Therefore, the word segmentation and NER training share all the parameters of the LSTM module. This facilitates the joint training.

3.4.3.1 Joint Training

In our integrated model, the LSTM parameters are used for both predicting word segmentations and NER. Therefore, we consider a joint training scheme. We maximize a (weighted) joint objective:

$$\mathcal{L}_{joint}(\Theta) = \lambda \mathcal{L}_s(\boldsymbol{y}_s; \boldsymbol{x}_s, \Theta) + \mathcal{L}_n(\boldsymbol{y}_n; \boldsymbol{x}_n, \Theta) \tag{3.6}$$

where λ trades off between better segmentations or better NER, and Θ includes all parameters used in both models. Since we are interested in improving NER we consider settings with $\lambda < 1$.

3.4.4 Parameter Estimation

We train all of our models using stochastic gradient descent (SGD.) We train for up to 30 epochs, stopping when NER results converged on dev data. We use a separate learning rate for each part of the joint objective, with a schedule that decays the learning rate by half if dev results do not improve after 5 consecutive epochs. Dropout is introduced in the input layer of LSTM following Chen et al. (2015a). We optimize two hyper-parameters using held out dev data: the joint coefficient λ in the interval $[0.5, 1]$ and the dropout rate in the interval $[0, 0.5]$. All other hyper-parameters were set to the values given by Chen et al. (2015a) for the LSTM and follow Section 3.3 for the CRF.

We train the joint model using an alternating optimization strategy. Since the segmentation dataset is significantly larger than the NER dataset, we subsample the former at each iteration to be the same size as the NER training data, with different subsamples in each iteration. We found subsampling critical and it significantly reduced training time and allowed us to better explore the hyper-parameter space.

We initialized LSTM input embeddings with pre-trained character-positional embeddings trained on 112,971,734 Weibo messages to initialize the input embeddings for the LSTM. We used word2vec with the same parameter settings as used in Section 3.3 to pre-train the embeddings.

3.4.5 Experiments and Analysis

3.4.5.1 Datasets

The segmentation data is taken from the SIGHAN 2005 shared task. We used the PKU portion, which includes 43,963 word sentences as training and 4,278 sentences as test. We did not apply any special preprocessing. We use the same training, development and test splits as Chen et al. (2015a)

We use the dataset and the split strategy introduced in Section 3.2 for NER. We note

		Named Entity						Named + Nominal Mention					
		Dev			Test			Dev			Test		
	Method	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
1	CRF with baseline features	60.27	25.43	35.77	57.47	25.77	35.59	72.06	32.56	44.85	59.84	23.55	33.80
2	+ Segment Features	62.34	27.75	38.40	58.06	27.84	37.63	58.50	38.87	46.71	47.43	26.77	34.23
3	Joint w/ Embeddings Best Results	57.41	35.84	44.13	57.98	35.57	44.09	72.55	36.88	48.90	63.84	29.45	40.38
4	+ Segment Features	47.40	42.20	44.65	48.08	38.66	42.86	76.38	36.54	49.44	63.36	26.77	37.64
5	Joint w/ Char Embeddings	58.76	32.95	42.22	57.89	34.02	42.86	66.88	35.55	46.42	55.15	29.35	38.32
6	+ Segment Features	51.47	40.46	45.31	52.55	37.11	43.50	65.43	40.86	50.31	54.01	32.58	40.64
7	Pipeline Seg. Repr. + NER	64.71	38.14	**48.00**	64.22	36.08	46.20	69.36	39.87	50.63	56.52	33.55	42.11
8	Jointly LSTM w/o feat.	59.22	35.26	44.20	60.00	35.57	44.66	60.10	39.53	47.70	56.90	31.94	40.91
9	Jointly Train Char. Emb.	64.21	35.26	45.52	63.16	37.11	46.75	73.55	37.87	50.00	65.33	31.61	42.61
10	Jointly Train LSTM Hidden	61.86	34.68	44.44	63.03	38.66	47.92	67.23	39.53	49.79	60.00	33.87	43.30
11	Jointly Train LSTM + Emb.	59.29	38.73	46.85	63.33	39.18	**48.41**	61.61	43.19	**50.78**	58.59	37.42	**45.67**

Table 3.3: NER results for named and nominal mentions on dev and test data.

that the word segmentation dataset is significantly larger than the NER data, which motivates our subsampling during training (Section 3.4.4).

3.4.5.2 Results and Analysis

Table 3.3 shows results for NER in terms of precision, recall and F1 for named (left) and nominal (right) mentions on both dev and test sets. The hyper-parameters are tuned on dev data and then applied on test. We now explain the results.

We begin by establishing a CRF baseline (#1) and show that adding segmentation fea-

tures helps (#2). However, adding those features to the full model (with embeddings) proposed in Section 3.3 (#3) did not improve results (#4). This is probably because the character-positional embeddings already carry segmentation information. Replacing the character-positional embeddings with character embeddings (#5) gets worse results than (#3), but benefits from adding segmentation features (#6). This demonstrates both that word segmentation helps and that character-positional embeddings effectively convey word boundary information.

We now consider our model of jointly training the character embeddings (#9), the LSTM hidden vectors (#10) and both (#11). They all improve over the best published results (#3). Jointly training the LSTM hidden vectors (#10) does better than jointly training the embeddings (#9), probably because they carry richer word boundary information. Using both representations achieves the single best result (#11): 4.3% improvement on named and 5.3% on nominal mentions F1 scores.

Finally, we examine how much of the gain is from joint training versus from pre-trained segmentation representations. We first train an LSTM for word segmentation, then use the trained embeddings and hidden vectors as inputs to the log-bilinear CRF model for NER, and fine tune these representations. This (#7) improved test F1 by 2%, about half of the overall improvements from joint training.

3.5 Dataset Improvements

After our paper (Peng and Dredze, 2015) introduced the task of named entity recognition (NER) on Chinese social media and released an accompanying dataset, there were several follow-up work used our dataset and experimental setup to conduct their research. Since our NER annotations on Weibo messages was constructed using Amazon Mechanical Turk[9], and the final annotations were generated by merging labels from multiple different Turkers using heuristics. This inevitably lead to inconsistencies and errors in the dataset. He and Sun (2017a) manually corrected the annotations which resulted in a much cleaner dataset[10]. They also reported results based on the improved dataset.

This section reports the results of our methods on the improved dataset. We also conduct thorough significant test and error analysis on this improved dataset. Table 3.4 gives an overview of the the results. We can see that the results follow the same trends as we discovered in our papers: multi-task learning of embeddings and named entity recognition performed much better than the feature-based baseline. Among the different em-

[9] `https://www.mturk.com/mturk/welcome`

[10] The improved dataset along with the code from our original paper is available on github.`https://github.com/hltcoe/golden-horse`. The best hyper-parameter settings are also given on that website.

[11] For the named + nominal mention case, the best results was significantly better than the third best, but not the second best

	Named Entity						Name + Nominal Mention					
	Dev			Test			Dev			Test		
Method	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
CRF with baseline features	63.83	35.50	45.63	68.81	34.72	46.15	72.99	44.47	55.27	73.39	38.65	50.63
+ Word Embeddings	73.56	37.87	50.00	71.82	36.57	48.47	67.75	53.47	59.77	64.08	47.83	54.77
+ Char Embeddings	54.11	46.75	50.16	58.06	41.67	48.52	65.47	51.67	57.76	62.66	46.62	53.46
+ Char-Pos Embeddings	69.16	43.79	53.62	69.16	39.81	50.74	70.14	51.93	59.68	68.16	47.59	56.05
Multi-task w/ Embeddings	74.00	43.79	55.02	74.78	39.81	51.96	70.50	53.98	61.14	70.14	48.07	57.18
Multi-task w/ Segmentation.	62.04	50.30	**55.56**	66.67	47.22	**55.28***	68.87	55.61	**61.53**	70.47	50.72	**58.99***

Table 3.4: NER results for named and named+nominal mentions on dev and test data. The bold numbers are the best. We conducted paired permutation test on the test predictions, and use * to mark the method that significantly better than other methods[11].

bedding variations, character-positional embeddings performed the best when being fine-tuned with CRF models; the joint learning strategy performed even better than just fune-tuning. Furthermore, the multi-task learning strategy that described in Section 3.4 with joint training on word segmentation dataset brings additional gain. Overall, on both development and test data, the multi-task learning with word segmentation dataset obtained the best results.

We conducted the paired permutation test on the test predictions obtained by different models, to gauge the significance of the improvements. With $p < 0.01$, the best model on the named entity recognition is significantly better than the second best model. On the whole dataset, however, the improvements obtained by the multi-task learning with word

Models	Named Entity			Nominal Mention			
	Prec	**Recall**	**F1**	**Prec**	**Recall**	**F1**	**Overall**
He and Sun (2017a)	66.93	40.67	50.60	66.46	53.57	59.32	54.82
He and Sun (2017b)	61.68	48.82	54.50	74.13	53.54	62.17	58.23
Peng and Dredze (2015)	74.78	39.81	51.96	71.92	53.03	61.05	57.18
Peng and Dredze (2016)	66.67	47.22	55.28	74.48	54.55	62.97	58.99

Table 3.5: Test results for Peng and Dredze (2015) and Peng and Dredze (2016) on the updated Chinese Social Media NER dataset. We got much better results than the originally reported number. We also listed the results in He and Sun (2017a) and He and Sun (2017b) for comparison purposes.

segmentation dataset was not significant over the multi-task learning with word embeddings. It is significantly better than the third best method with $p < 0.01$.

For fair comparisons, we also compare the best results of our proposed model from Section 3.3 and Section 3.4 with He and Sun (2017a) and He and Sun (2017b) on the improved dataset. This will allow future work to compare to our results using the cleaner dataset. For these comparisons we tuned the hyper-parameters as was discussed in Section 3.3.4 and Section 3.4.4. We only show the best results obtained from the best model

in Section 3.3 and Section 3.4. Table 3.5 shows the results [12].

It is clear that on the improved dataset, our methods in Peng and Dredze (2015) and Peng and Dredze (2016) out-performed He and Sun (2017a) and He and Sun (2017b).

3.6 Conclusion

In this chapter, we explored two multi-task learning strategies for a typical low-resource IE task: Chinese social media NER. We initiated the task, and released the dataset we composed to facilitate the research in related area. Our results demonstrated several interesting empirical findings:

1. NER for Chinese social media remains a challenging task, results lag behind both formal Chinese text and English Twitter.

2. Our multi-task learning strategies, jointly learning with unlabeled data and annotations for Chinese word segmentation, provide large improvements over the classic supervised NER models.

3. When two tasks are highly related, sharing high-level representations (e.g. contextual representation produced by RNNs is more efficient than just sharing low-level word embeddings.

[12] To facilitate comparison with He and Sun (2017a), we report results using their format.

4. In the second multi-task learning strategy, our segmentation data is from the news domain, whereas the NER data is from social media. While experiments showed large improvements in directly applying multi-task learning to these different domains, expanding our model to explicitly include domain adaptation mechanisms will be an interesting future work since it is well known that segmentation systems trained on news do worse on social media (Duan et al., 2012).

Chapter 4

Multi-task Domain Adaptation for Sequence Tagging

The previous chapter has shown the effectiveness of multi-task learning for low-resource information extraction. A surprising finding is that multi-task learning would work even if the data for the different tasks are from different domains.

In this chapter, we investigate whether explicitly modeling domains along with multi-task learning would help information extraction in the low-resource setting. The underlying intuition lies in the fact that both multi-task learning and domain adaptation are efficient methods for combating the problem of insufficient supervisions. A principled way to combine the benefit of each method is desirable. This chapter is based on materials

from Peng and Dredze (2017).

We propose a neural network framework that supports domain adaptation for multiple tasks *simultaneously*, and learns shared representations that better generalize for different domains. We apply the proposed framework to sequence tagging problems considering two tasks: Chinese word segmentation and named entity recognition, and two domains: news article and social media. Experiments show that multi-task domain adaptation works better than disjoint domain adaptation for each task, or multi-task learning disregard the domain mismatch. We achieve the state-of-the-art results for both tasks in the social media domain.

4.1 Introduction

Many natural language processing tasks have abundant annotations in formal domain (news articles) but suffer a significant performance drop when applied to a new domain, where only a small number of annotated examples are available. The idea behind domain adaptation is to leverage annotations from high-resource (source) domains to improve predictions in low-resource (target) domains by training a predictor for a single task across different domains.

Domain adaptation work tends to focus on changes in data distributions (e.g. different words are used in each domain). Formally, it focuses on modeling the change of distribu-

tion of the input $\mathbf{x} \in \mathbb{R}^n$. Domain adaptation methods include unsupervised (Blitzer et al., 2006) and supervised (Daumé III, 2009b) variants, depending on whether there exists no or some training data in the target domain. This chapter considers the case of supervised domain adaptation, where we have a limited amount of target domain training data, but much more training data in a source domain.

Work on domain adaptation mostly follows two approaches: parameter tying (i.e. linking similar features during learning) (Dredze and Crammer, 2008; Daumé III, 2009b,a; Finkel and Manning, 2009; Kumar et al., 2010; Dredze et al., 2010a), and learning cross domain representations (Blitzer et al., 2006, 2007; Glorot et al., 2011; Chen et al., 2012; Yang and Eisenstein, 2015). Often times, domain adaptation is formulated as learning a single model for the same task across domains, although with a focus on maximizing target domain performance.

Multi-task learning (MTL) (Caruana, 1997), on the other hand, jointly learns models for several tasks by learning a single data representation common to each task (Ando and Zhang, 2005; Collobert et al., 2011; Liu et al., 2016c; Peng and Dredze, 2016; Yang et al., 2016; Liu et al., 2016a). Formally, it focuses on modeling the changes in the output space $\mathbf{y} \in \mathcal{Y}$ by learning a common representation for the input. Given the similarity between domain adaptation and MTL, it is natural to ask: can domain adaptation benefit from jointly learning across several tasks?

This chapter investigates how MTL can induce better representations for domain adaptation. There are several benefits. First, learning multiple tasks provides more training data for learning. Second, MTL provides a better inductive learning bias so that the learned representations better generalize. Third, considering several tasks in domain adaptation opens up the opportunities to adapt from a different domain *and* a different task, a mismatch setting which has not previously been explored. We present a representation learning framework based on MTL that incorporates parameter tying strategies common in domain adaptation. Our framework is based on a bidirectional long short-term memory network with a conditional random fields (BiLSTM-CRFs) (Lample et al., 2016) for sequence tagging. We consider sequence tagging problem since they are common in NLP applications and have been demonstrated to benefit from learning representations (Lample et al., 2016; Yang et al., 2016; Peng and Dredze, 2016; Ma and Hovy, 2016).

This chapter makes the following contributions:

- A neural MTL domain adaptation framework that considers several tasks *simultaneously* when doing domain adaptation.
- A new domain/task mismatch setting: where you have two datasets from two different, but related domains and tasks.
- State-of-the-art results on Chinese word segmentation and named entity recognition in social media data.

4.2 Model

We begin with a brief overview of our model, and then instantiate each layer with specific neural architectures to conduct multi-task domain adaptation for sequence tagging. Figure 4.1 summarizes the entire model presented in this section.

A representation learner that is shared across all domains and tasks, and learns robust data representations for features. This feeds a domain projection layer, with one projection for each domain that transforms the learned representations for different domains into the same shared space. As a result, the final layer of task specific models, which learns feature weights for different tasks, can be shared across domains since the learned representations (features) for different domains are now in the same space. The framework is flexible in both the number of tasks and domains. Increasing the number of domains linearly increases domain projection parameters, with the number of other model parameters unchanged. Similarly, increasing the number of tasks only linearly increases the number of task specific model parameters. If there is only one domain, then the framework reduces to a multi-task learning framework, and similarly, the framework reduces to a standard domain adaptation framework if there is only one task.

The shared representation learner, domain projections and task specific models can be instantiated based on the application. In this chapter, we focus on sequence tagging problems. We now introduce our instantiated neural architecture for multi-task domain

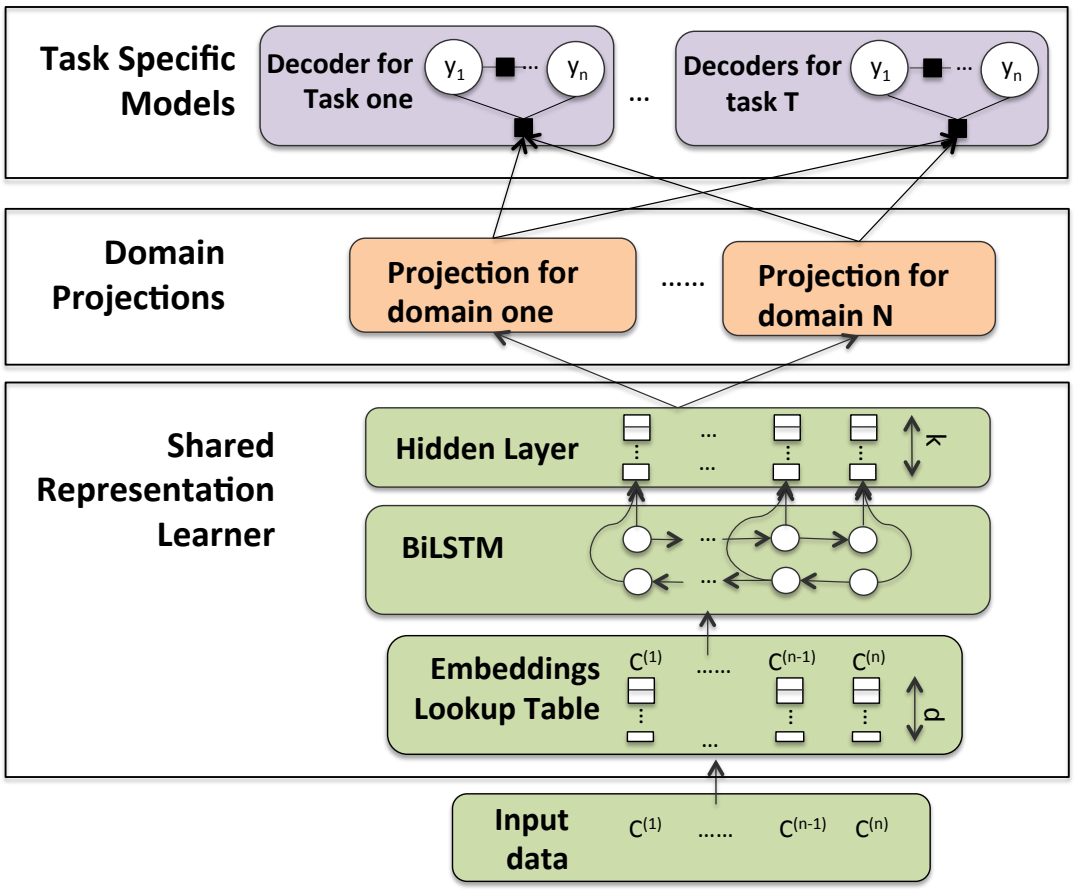


Figure 4.1: An overview of our proposed model framework. The bottom layer is shared by all tasks and domains. The domain projections contain one projection per domain and the task specific models (top layer) contain one model per task.

adaptation for sequence tagging.

4.2.1 BiLSTM for representation learning

Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN) that models interdependencies in sequential data. It addresses the vanishing or exploding gradients (Bengio et al., 1994; Pascanu et al., 2013) problems of vanilla RNNs by using a series of gates (input, forget and output gates) to control how memory is propagated in the hidden states of the model, and thus effectively captures long-distance dependencies between the inputs.

Many NLP applications use bi-directional LSTMs (BiLSTM) (Dyer et al., 2015) to scan both left-to-right and right-to-left, which capture left and right context. The hidden vectors produced by both LSTMs are concatenated to form the final output vector $h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t}$. BiLSTMs have become a common building block for learning representations in NLP and have achieved impressive performance in problems such as sequence tagging (Lample et al., 2016; Yang et al., 2016; Ma and Hovy, 2016), relation classification (Xu et al., 2015b; Zhang et al., 2015), and syntactic parsing (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016). We use a BiLSTM as our representation learner. It produces a hidden vector for each token in the sentence, which we denote as:

$$h_t = \text{BiLSTM}(x_{1:n}, t) \tag{4.1}$$

where $x_{1:n}$ denotes the whole input sequence of length n, and t denotes the t-th position. The representation for the whole sequence is thus denoted as $\boldsymbol{h} = h_{1:n}$.

4.2.2 Domain Projections

Domain adaptation requires learning a shared representation that generalizes across domains. Ideally, parameter estimation of the BiLSTM should learn to produce such robust features. However, this may place a heavy burden on the BiLSTM; it does not know the identity of each domain yet must still learns how to map two heterogeneous input types to the same representation. To reduce this burden, we introduce a domain projection layer, which relies on explicit domain specific transformation functions to produce shared representations. We place this transformation between the representation learner and the task specific predictor to alleviates pressure on the representation learner to learn cross domain representations. Note that the domain projection layer works *jointly* with the representation learner to produce shared representations. We experiment with two simple strategies for domain projections which are based on previous lines of work in domain adaptation.

4.2.2.1 Domain Mask

The first strategy is inspired by Daumé III (2009b) and Yang and Hospedales (2014), which split the representations into several regions, with one region shared among domains, and others specific for each domain. As a result, the BiLSTM representation learner will learn to put the features that are suitable to be shared across domains into the shared region, and domain specific features to the corresponding region for the domain.

We implement this strategy by defining domain masks $\boldsymbol{m}_d$, which is a vector for the dth domain. The mask $\boldsymbol{m}_d$ has value 1 for the effective dimensions of domain d and domain shared region, and 0 for all other dimensions. For example, assume we have two domains and a k dimensional hidden vector for features, the first $k/3$-dimensions is shared between the two domains, while the $k/3+1$ to $2k/3$ dimensions are used only for domain 1, and the remaining dimensions for domain 2. The mask for domain 1 and domain 2 would be:

$$\boldsymbol{m}_1 = [\vec{1}, \vec{1}, \vec{0}], \quad \boldsymbol{m}_2 = [\vec{1}, \vec{0}, \vec{1}]. \tag{4.2}$$

We can then apply these masks directly to the hidden vectors $\boldsymbol{h}$ learned by the BiLSTM to produce a projected hidden state $\hat{\boldsymbol{h}}$:

$$\hat{\boldsymbol{h}} = \boldsymbol{m}_d \odot \boldsymbol{h}, \tag{4.3}$$

where $\odot$ denotes element-wise multiplication. Since only a subset of the dimensions are used as features in each domain, the BiLSTM will be encouraged to learn to partition the dimensions of the output hidden vectors into domains.

Note that in Daumé III (2009b), the domain masks operate on hand engineered features, thus only affect feature weights. However, here the domain masks will change the parameters learned in BiLSTMs as well, changing the learned features. Therefore, training data from one domain will also change the other domains' representation. When we jointly train with data from all domains, the model has to balance the training objectives for all domains simultaneously.

4.2.2.2 Linear Transformation

The second domain adaptation strategy we explore is a linear transformation to each domain. Many previous work had used linear transformation to capture the shift of meanings/distributions in word embeddings (Bolukbasi et al., 2016; Hamilton et al., 2016). We follow their settings. The domain specific linear transformation matrix is denoted as T_d. Given a k-dimensional vector representation $\boldsymbol{h}$, T_d is a $k \times k$ matrix that projects the learned BiLSTM hidden vector to a common space that can be used by a shared task specific model. We use the transformation:

$$\hat{\boldsymbol{h}} = T_d \boldsymbol{h}. \tag{4.4}$$

We learn one T_d for each domain jointly with other model parameters, in the hope that the design of the whole architecture, and the sharing of the domain-specific models (discuss in details in next section), will encourage the learned T_d to transform the different distributions for different domains into the same shared distribution space.

While this model has greater freedom in learning representations across domains, it relies on the training data to learn a good transformation, and does not explicitly partition the representations into domain regions.

4.2.2.3 Discussion

Other possible domain modeling strategies: The domain mask is a reminisce of

Daumé III (2009b)'s frustratingly easy domain adaptation idea, and the linear transformation matrix is a reminisce of several prior work on using linear transformation to capture the shift of meanings (Bolukbasi et al., 2016; Hamilton et al., 2016). The major differences between our proposal and theirs lie in the fact that our joint learning neural architectures encourage the representation learner (in our case a Bi-LSTM) to automatically learn representations that have a notion of domains. The deep learning framework makes the learning of the domain transformation more convenient.

Other famous domain adaptation methods such as structural correspondence learning (SCL) (Blitzer et al., 2007) can also be easily integrated to our framework by using learning SCLs as an initialization of our domain transformation layer.

The placement of domain projection layer: It is arguable that there are several desirable way to put the domain projections, since they aim at projecting the different distribution of the inputs $\mathbf{x}$ into the same space. One possible position would be put the projections right after the word embedding layer. We experimented with this setting in our preliminary study, however, it did not perform as good as putting them after the hidden layer of the RNNs. One possible reason could be that the RNNs learned some feature representations that reflect the combinations of words, and the distribution of those combinations also shifts across domains. Placing the domain projection layer on the top helped capture these shifts.

4.2.3 Task Specific Neural-CRF Models

Multi-task domain adaptation *simultaneously* considers several tasks adapting domains since the related tasks would help induce more robust data representations for domain adaptation. Additionally, it enables leveraging more data to learn better domain projections. The goal of a task specific model is to learn parameters to project the shared representations to the desired outputs for the corresponding task. Different tasks that define different output spaces need separate task specific models.

For our applications to sequence tagging problems, we choose Conditional Random Fields (CRFs) (Lafferty et al., 2001) as task specific models, since it is widely used in previous work and is shown to benefit from learning representations (Lample et al., 2016; Ma and Hovy, 2016). These "Neural-CRFs" define the conditional probability of a sequence of labels given the input as:

$$p(\boldsymbol{y}^k|\boldsymbol{x}^k; W) = \frac{\prod_{i=1}^{n} \exp\left(W^T F(y_{i-1}^k, y_i^k, \psi(\boldsymbol{x}^k))\right)}{Z^k},$$

where i indexes the position in the sequence, F is the feature function, and $\psi(\boldsymbol{x}^k)$ defines a transformation of the original input, in our case $\psi(\boldsymbol{x}^k) = BiLSTM(\boldsymbol{x}^k)$. Z^k is the partition function defined as:

$$Z^k = \sum_{\boldsymbol{y} \in \mathcal{Y}} \prod_{i=1}^{n} \exp\left(W^T F(y_{i-1}^k, y_i^k, \psi(\boldsymbol{x}^k))\right).$$

4.2.3.1 Sharing Task Specific Models

We could create a CRF decoder for each task and domain. This is the practice of some previous work. Yang and Hospedales (2014) considered domain adaptation (or multi-domain learning) as a special case of MTL, and learn separate models for the same task from different domains.

Instead, we argue that learning a single model for a task regardless of the number of domains draws strong connections to the traditional domain adaptation literature. It enjoys the benefit of increasing the amount of training data for each task by considering different domains, and better handles the problem of shifts in data distributions by explicitly considering different domains. Therefore, we use a single CRF per *task*, shared across all domains.

4.3 Parameter Estimation

The proposed neural architecture for multi-task domain adaptation can be trained end-to-end by maximizing data log-likelihood. As there are $D \times T$ [1] datasets, the final loss function is a linear combination of the log-likelihood of each dataset. For simplicity, we give each dataset equal weight when forming the linear combination.

[1] D denotes the number of domains and T the number of tasks

4.3.1 Training

Model training is a straightforward application of gradient based back-propagation. We use *alternating optimization* among each dataset with stochastic gradient descent (SGD). To prevent training from skewing the model to a specific dataset due to the optimization order, we subsample the number of instances used in each epoch with a fraction λ w.r.t. the smallest dataset size, which is tuned as a hyper-parameter on development data. A separate learning rate is tuned for each dataset, and we decay the learning rate when results on development data do not improve after 5 consecutive epochs. We train for up to 30 epochs and use early stopping (Caruana et al., 2001; Graves et al., 2013) as measured on development data. We select the best model for each dataset based on hyper-parameter tuning. We use dropout on the embeddings and the BiLSTM output vectors as in Ma and Hovy (2016).

4.3.2 Initialization

We use pre-trained Chinese embeddings composed in Chapter 3 with dimension 100. We use the character-positional variation. These embeddings are updated during learning by fine-tuning. We use a single set of embeddings for all domains, however, for future research in multi-domain learning, we suggest using different embeddings learn from the corresponding domains as initialization, and update them separately for different domains.

All other model parameters are initialized uniformly at random in the range of $[-1, 1]$.

4.3.3 Inference

For training the CRFs, we use marginal inference and maximize the marginal probabilities of the labels in the training data. At test time, the label sequence with highest conditional probability $y^* = \arg\max p(y|x; \Omega)$ is obtained by MAP inference.

4.3.4 Hyper-parameters

Our hyper-parameters include the initial learning rate (per dataset, in the range of [0.005, 0.01, 0.02]), the dropout rate for the input embedding and the hidden vectors (in the range of [0, 0.1, 0.2]), and the subsample coefficient for each setting (in the range of [5, 10, 15]). We tune these hyper-parameter using beam search on development data. For convenience, the embedding and the LSTM hidden vector dimensions are set to 100 and 150 respectively.

4.4 Experimental Setup

We test the effectiveness of the multi-task domain adaptation framework on two sequence tagging problems: Chinese word segmentation (CWS) and named entity recog-

nition (NER). We consider two domains: news and social media, with news the source domain and social media the target domain.

4.4.1 Datasets

We consider two domains: news and social media for the two tasks: CWS and NER. This results in four datasets: news CWS data comes from the SIGHAN 2005 shared task (*SighanCWS*) (Emerson, 2005), news NER data comes from the SIGHAN 2006 shared task (*SighanNER*) (Levow, 2006), social CWS data (*WeiboSeg*) created by (Zhang et al., 2013a), and social NER data (*WeiboNER*) created in Chapter 3.

Both *SighanCWS* and *SighanNER* contain several portions[2]; we use those for simplified Chinese (PKU and MSR respectively). The datasets do not have development data, so we hold out the last 10% of training data for development. *SighanNER* contains three entity types (person, organization and location), while *WeiboNER* is annotated with four entity types (person, organization, location and geo-political entity), including named and nominal mentions. To match the two tag sets, we only use named mentions in *WeiboNER* and merge geo-political entities and locations. The 2000 annotated instances in *WeiboSeg* were meant only for evaluation, so we split the data ourselves using an 8:1:1 split for training, development, and test. Hyper-parameters are tuned on the development data and

[2]The portions are annotated by different institutes, and cover both traditional and simplified Chinese

Dataset	#Train	#Dev	#Test
SighanCWS	39,567	4,396	4,278
SighanNER	16,814	1,868	4,636
WeiboCWS	1,600	200	200
WeiboNER	1,350	270	270

Table 4.1: Datasets statistics.

we report the precision, recall, and F1 score on the test portion. Detailed data statistics is shown in Table 4.1.

4.4.2 Baselines

We consider two baselines common in domain adaptation experiments. The first baseline only considers a single dataset at a time (*separate*) by training *separate* models just on in-domain training data. The second baseline (*mix*) uses out-of-domain training data for the same task by mixing it with the in-domain data. For both the baselines, we use the BiLSTM-CRFs neural architecture (Lample et al., 2016), which achieved state-of-the-art results on NER and other sequence tagging tasks (Peng and Dredze, 2016; Ma and Hovy, 2016; Yang et al., 2016).

Settings	Datasets / Methods	CWS			NER		
		Prec	**Recall**	**F1**	**Prec**	**Recall**	**F1**
Baseline	Separate	86.2	85.7	86.0	57.2	42.1	48.5
	Mix	87.0	86.1	86.5	60.9	44.0	51.1
Domain Adapt	Domain Mask	88.7	87.1	87.9	68.2	48.6	56.8
	Linear Projection	88.0	87.5	87.7	73.3	45.8	56.4
Multi-task DA	Domain Mask	**89.7**	88.3	**89.0**	60.2	**52.3**	**59.9**
	Linear Projection	89.1	**88.6**	88.9	**68.6**	49.5	57.5

Table 4.2: Test results for CWS and Chinese NER on the target social media domain. The first two rows are baselines (Section 4.4.2,) followed by two domain adaptation models that only considers one task a time. The last two rows are the proposed multi-task domain adaptation framework building upon the two domain adaptation models, respectively. Domain adaptation models leverage out-of-domain training data and *significantly* improve over the *Separate* baseline, as well as the *Mix* baseline which trains with the out-of-domain data without considering domain shift. Multi-task domain adaptation further *significantly* improves over traditional domain adaptation on both domain adaptation models and achieved the new state-of-the-art results on the two tasks.

4.5 Experimental Results

4.5.1 Main Results

Table 4.2 presents the results for domain adaptation to the target domain (social media) test data . The baseline method *Mix* improves over *Separate* as it benefits from the increased training data. The single task domain adaptation models are a special case of the proposed multi-task domain adaptation framework: with only one task specific model in the top layer (CWS or NER). Both of our approaches (domain mask and linear projection) improve over the baseline methods. Knowing the domain of the training data helps the model better learn effective representations. Finally, we see further improvements in the multi-task domain adaptation setting. By considering additional tasks in addition to domains, we achieve new state-of-the-art results on the two tasks. We compare to the best published results from Zhang et al. (2013a) and Peng and Dredze (2016) with F1 scores of 87.5% (CWS) and 55.3% (NER), respectively.

4.5.2 Statistical Significance

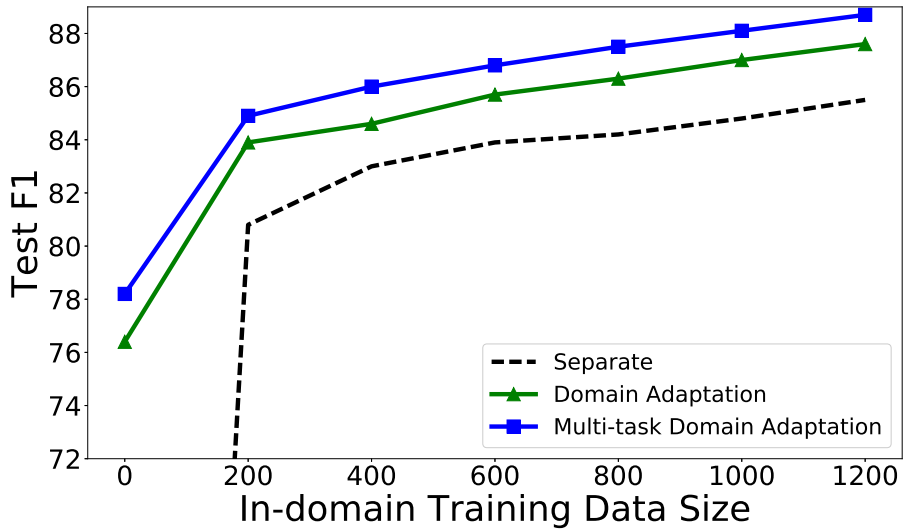
We measures statistical significance using McNemar's chi-square test (McNemar, 1947) for paired significant test. We treated the predicted spans (not tokens) that agreed with the ground truth as positive, otherwise negative. For the NER task, we only count the spans

that corresponds to named entities. We compare the best baseline (*mix*) and the two domain adaptation models, as well as between the domain adaptation models and their multi-task domain adaptation counterpart. Both the domain adaptation models *significantly* improved over the *mix* baseline ($p < 0.01$), and the multi-task domain adaptation methods *significantly* improved over their single task domain adaptation counterpart ($p < 0.01$). We cannot conduct paired significance tests with the best published results since we do not have access to their outputs.

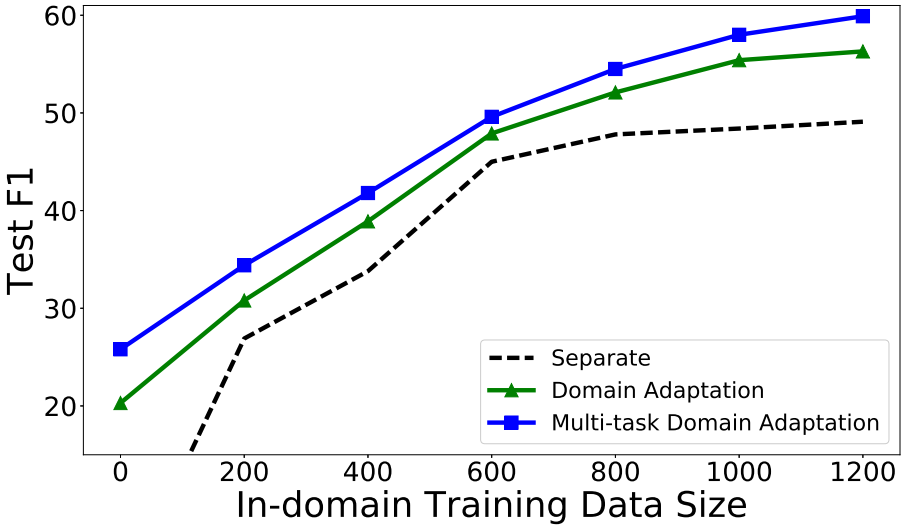
4.5.3 In-domain Training Data

We also conducted several experiments to show the flexibility of our multi-task domain adaptation framework and analyze the behavior of the models by varying the training data.

We first consider the effect of in-domain training data size. Figure 4.2 shows the test F1 for the *Separate* baseline which only considers in-domain training data compared with both a single-task domain adaptation model and a multi-task domain adaptation model. For simplicity, we only show the curve for the *Domain Mask* variant. As expected, we observe diminishing returns with additional in-domain training data on both tasks, but domain adaptation and multi-task domain adaptation methods suffer less from the diminishing return, especially on the NER task (Figure 4.2a). The curves for domain adaptation and multi-task domain adaptation also appear to be smoother, as they leverage more data



(a)



(b)

Figure 4.2: The effect of training data size on social media CWS (top) and NER (bottom) tasks.

to learn input representations, and thus are more robust.

When we have no in-domain training data, the problem reduces to unsupervised domain adaptation. Our framework applies here as well, and multi-task domain adaptation achieves performance close to the *Separate* baseline with only 200 in-domain training examples.

Dataset Numbers	**Datasets** / **Methods**	**CWS**			**NER**		
		Prec	**Recall**	**F1**	**Prec**	**Recall**	**F1**
One Dataset	Separate	86.2	85.7	86.0	57.2	42.1	48.5
Two Datasets	Multi-task	87.7	86.2	86.9	59.1	44.9	51.1
	Domain Adaptation	88.7	87.1	87.9	68.2	48.6	56.8
	Mismatch	87.8	86.3	87.1	60.8	45.0	51.7
Four Datasets	All Multi-task	88.7	87.7	88.2	67.2	48.5	56.4
	Multi-task DA	89.7	88.3	89.0	60.2	52.3	59.9

Table 4.3: Model variations grouped by number of training datesets.

4.5.4 Model Variations

The multi-task domain adaptation framework is flexible regarding the number of domains and tasks, thus the number of datasets. Table 4.3 shows the results for several model

variations, grouped by the number of training datasets. With one dataset, it is just the standard supervised learning setting, which reduces to our *Separate* baseline.

With two datasets, the framework can do multi-task learning (with two datasets from the same domain but different tasks), single task domain adaptation (with two datasets for the same task but from different domains), and a novel mismatch setting (with two datasets from *both* different domains *and* different tasks). As shown in the second section of Table 4.3, including additional training data – no matter from another task, domain or both – always improves the performance. A hidden factor not shown in the table is the additional dataset's size. For multi-task learning, since we are look at the social media domain, the additional dataset size is small. This is probably the reason why the *Mismatch* setting leveraging data from a different task *and* domain surprisingly outperformed multi-task learning. *Domain adaptation* enjoys both the benefits of a large amount of additional training data and an aligned task, thus achieving the best results among the two dataset settings.

When conducting multi-task domain adaptation, we are leveraging four datasets. One concern is that the performance gains only come from additional training data, instead of the deliberately designed framework (Joshi et al., 2012). We thus also compare with a strategy which treats the same task for a different domain as a different task. The corresponding neural architecture is a shared BiLSTM with four separate task-specific models:

we call it the *All Multi-task* setting. The results show that explicitly modeling data domains gives extra benefit than blindly throwing in more training data. We found the same benefits when experimenting with three datasets (instead of 2 or 4).

4.5.5 Discussion

There is also an interesting observation from Table 4.3. Comparing the results that obtained by training on two datasets and separate training, multi-task learning gave the least gain, especially for the Chinese word segmentation task. One possible explanation could be that when several tasks form a pipeline in the processing hierarchy, the tasks that are later in the pipeline would benefit from multi-task learning more than the tasks earlier in the pipeline.

4.6 Related Work

The previous work on domain adaptation exclusively focused on building a unified model for *a* task across domain. However, we argue that a flexible framework for domain adaptation on several tasks simultaneously would be beneficial. To the best of our knowledge, the work that is closest to ours is Yang and Hospedales (2014), which provided a unified perspective for multi-task learning and multi-domain learning (a more general case of

domain adaptation) under the same perspective of representation learning. However, they only focused on exploring the common ground of multi-task learning and multi-domain learning, and did not explore the possibility of having multi-task learning to help domain adaptation. We briefly review previous work on domain adaptation and multi-task learning below.

4.6.1 Domain Adaptation

In domain adaptation, or more general multi-domain learning, the goal is to learn a single model that can produce accurate predictions for multiple domains. An important characteristic of learning across domains is that each domain represents data drawn from a different distribution, yet share many commonalities. The larger the difference between these distributions, the larger the generalization error when learning across domains (Ben-David et al., 2010; Mansour et al., 2009).

As a result, a long line of work in multi-domain learning concerns learning shared representations, such as through identifying alignments between features (Blitzer et al., 2007, 2006), learning with deep networks (Glorot et al., 2011), using transfer component analysis (Pan et al., 2011), learning feature embeddings (Yang and Eisenstein, 2015) and kernel methods for learning low dimensional domain structures (Gong et al., 2012), among others. Another line sought for feature weight tying (Dredze and Crammer, 2008; Daumé III,

2009b,a; Finkel and Manning, 2009; Kumar et al., 2010; Dredze et al., 2010a) to transfer the learned feature weights across domains.

We combined the two lines and explored joint learning with multiple tasks.

4.6.2 Multi-task Learning

The goal of MTL (Caruana, 1997; Ando and Zhang, 2005) is to improve performance on different tasks by learning them jointly.

With recent progress in deep representation learning, new work considers MTL with neural networks in a general framework: learn a shared representations for all the tasks, and then a task specific predictor. The representations shared by tasks go from lower level word representations (Collobert and Weston, 2008; Collobert et al., 2011), to higher level contextual representations learned by Recurrent Neural Networks (RNNs) (Liu et al., 2016b; Yang et al., 2016) or other neural architectures (Liu et al., 2016a; Søgaard and Goldberg, 2016b; Benton et al., 2017). For the higher level representation sharing, the parameters of the word embeddings and the shared neural representation learning are jointly learned. MTL has helped in many NLP tasks, such as sequence tagging (Collobert et al., 2011; Peng and Dredze, 2016; Søgaard and Goldberg, 2016b; Yang et al., 2016), text classification (Liu et al., 2016b,a), and discourse analysis (Liu et al., 2016c).

We expand the spectrum by exploring how multi-task learning can help domain adap-

tation.

4.7 Conclusion

We have presented a framework for multi-task domain adaptation, and instantiated a neural architecture for sequence tagging problems. The framework is composed of a shared representation learner for all datasets, a domain projection layer that learns one projection per domain, and a task-specific model layer that learns one set of feature weights per task. The proposed neural architecture can be trained end-to-end, and achieved the state-of-the-art results for Chinese word segmentation and NER on social media domain.

With this framework in mind, there are several interesting future directions to explore. First, we considered common domain adaptation schemas with our domain mask and linear projection. However, there are many more sophisticated methods that we can consider integrating into our model (Blitzer et al., 2007; Yang and Eisenstein, 2015). Second, we only experimented with Chinese language, which has the problem of word segmentation. However, many languages have natural delimiters in their writing systems. The multi-task learning with word segmentation is not applicable to those languages. We plan to explore the multi-task domain adaptation framework on other languages, to investigate whether other core NLP tasks such as morphological analysis, noun phrase chunking, part-of- speech tagging are helpful in multi-task learning. Third, we only experimented

with sequence tagging problems. However, the proposed framework is generally applicable to other problems such as text classification, parsing, and machine translation. We will explore these applications in the future. Finally, our work draws on two traditions in multi-domain learning: parameter sharing (on the task specific models) and representation learning (the shared representation learner). An interesting future direction is to explore how other domain adaptation methods can be realized in a deep architecture.

Chapter 5

Graph LSTM for Cross-Sentence N-ary Relation Extraction

The previous two chapters explored multi-task learning and domain adaptation for leveraging heterogeneous sources to learn representations to help IE in the low-resource setting. The applications are sequence tagging tasks in social media domain.

From this chapter on, we start the exploration of another main direction of this thesis: representation learning with explicit structure modeling for low-resource IE. We initiate the task of cross-sentence n-ary relation extraction in this chapter, which is a novel setting for relation extraction. As we reviewed in Chapter 2.1.3, the previous work on relation extraction focused on the single sentence binary relation case. However, in some high-

valued domain such as biomedicine, it is common to have relations that involve more than two entities and have multiple sentences describing the relation. This is a typical low-resource IE scenario, as we do not have any available annotated data to train the model. However, this is a common scenario whenever a new task is initiated.

This chapter composes a small noisy annotated dataset by distance supervision, and proposes a graph long short-term memory networks (graph LSTMs) to learn representations for relation extraction. The proposed graph LSTMs architecture can be easily extended to cross-sentence n-ary relation extraction. The graph formulation provides a unifying way to explore different LSTM approaches and incorporate various intra-sentential and inter-sentential structures, such as sequential, syntactic, and discourse relations. A robust contextual representation is learned for the entities, which serves as input to the relation classifier, making it easy for scaling to arbitrary relation arity n, as well as for multi-task learning with related relations.

We evaluate the proposed framework in two important domains in precision medicine. We demonstrate its effectiveness with both supervised learning and distant supervision. Cross sentence extraction produced more knowledge, and multi-task learning significantly improved extraction accuracy. A thorough analysis comparing various LSTM approaches yielded interesting insight on how linguistic analysis impacts the performance.

5.1 Introduction

Relation extraction has made great stride in the newswire and Web domains. Recently, there has been increasing interest in applying relation extraction to high-valued domains such as biomedicine. Craven and Kumlien (1999) first proposed to extract knowledge from biomedical publications using rule-based methods. Ravikumar et al. (2014) mined knowledge from literature for genetic pathway. Kim et al. (2009) composed the GENIA datasets as well as organized several shared tasks for information extraction in biomedical domain for cancer research. The advent of $1000 human genome[1] heralds the dawn of precision medicine, but progress in personalized cancer treatment is hindered by the bottleneck in interpreting genomic data using prior knowledge. For example, given a tumor sequence, the molecular tumor board needs to determine which genes and mutations are important, and what drugs are available to treat them. There is a wealth of relevant knowledge in the research literature, which grows at an astonishing rate. PubMed[2], the online repository of biomedical articles, adds two new papers per minute, or one million each year. It thus becomes imperative to advance relation extraction for machine reading.

There is a vast literature on relation extraction, but past work focuses primarily on binary relations in single sentences, which is not up for the new challenges. Consider the following example: "*The deletion mutation on exon-19 of* **EGFR** *gene was present in 16*

[1] `http://www.illumina.com/systems/hiseq-x-sequencing-system.html`
[2] `https://www.ncbi.nlm.nih.gov/pubmed`

patients, while the **L858E** *point mutation on exon-21 was noted in 10. All patients were treated with* **gefitinib** *and showed a partial response.*". Collectively, the two sentences convey the fact that there is a ternary interaction between the three entities in bold, which is not expressed in either sentence alone. Namely, tumors with *L858E* mutation in *EGFR* gene can be treated with *gefitinib*. Extracting such knowledge clearly requires moving beyond binary relations and single sentences.

n-ary relations and cross-sentence extraction have received relatively little attention in the past. Prior work on n-ary relation extraction focused on single sentences (Palmer et al., 2005; McDonald et al., 2005) or entity-centric attributes that can be extracted largely independently (Chinchor, 1998; Surdeanu and Heng, 2014). Prior work on cross-sentence extraction often used coreference to gain access to arguments in a different sentence (Gerber and Chai, 2010; Yoshikawa et al., 2011), without truly modeling inter-sentential relational patterns. (See Section 5.7 for more detailed discussion.) A notable exception is Quirk and Poon (2016), which applied distant supervision to general cross-sentence relation extraction, but was limited to binary relations.

In this chapter, we propose a general framework for cross-sentence n-ary relation extraction, using graph long short-term memory (graph LSTM). By adopting the graph formulation, our framework subsumes prior approaches based on chain or tree LSTMs, and can incorporate a rich set of linguistic analyses to aid relation extraction. Relation clas-

sification takes as input the entity representations learned from the entire text, and can easily scale to arbitrary relation arity n. Our approach also facilitates joint learning with sub-relations, for which supervision signal is often more abundant compared to the n-ary relation.

We conducted extensive experiments on two important domains in precision medicine. In both distant supervision and supervised learning, graph LSTM outperformed other neural architectures and a well-engineered feature-based classifier. Multi-task learning with sub-relations led to further improvement. Syntactic analysis had a significant impact on the performance of graph LSTM, especially when its quality was high.

In the molecular tumor board domain, PubMed-scale extraction using distant supervision from a small set of known interactions produced orders of magnitude more knowledge, and cross-sentence extraction tripled the yield compared to single-sentence extraction. Manual evaluation verified that the extraction accuracy is high despite the lack of annotated examples.

5.2 Cross-sentence n-ary relation extraction

Let $e_1, \cdots, e_m$ be entity mentions in text T. Relation extraction can be formulated as the classification problem of determining whether a relation R holds for $e_1, \cdots, e_m$ in T. For example, given a cancer patient with mutation v in gene g, the molecular tumor

board wants to know if this type of cancer would respond to drug d. Literature with such knowledge has been growing rapidly, and we can help the tumor board by extracting the `Respond` relation and see if it holds for the (d, g, v) triple.

Traditional extraction approaches focus on binary relations within single sentences (i.e., $m = 2$ and T is a sentence), and cannot handle the aforementioned ternary relations naturally. Moreover, as n increases when we focus on more complex relations, it becomes increasingly rare that the relation instances will be contained completely in a single sentence. In this chapter, we will generalize extraction to cross-sentence, n-ary relations, where $m > 2$ and T can contain multiple sentences. As will be seen in our experiments section, n-ary relations are crucial for high-valued domains such as biomedicine, and expanding beyond sentence boundary enables us to extract far more knowledge.

In the standard binary-relation setting, the dominant approaches generally revolve around the shortest dependency path between the two entities in question, either deriving rich features from the path or modeling it using deep neural networks. Generalizing this paradigm to n-ary setting is challenging, as there are $\binom{n}{2}$ paths. One way to combat this is to follow the Davidsonian semantics by identifying a trigger phrase to signify the whole relation, and reduce the n-ary relation to n binary relations between the trigger and an argument. However, challenges remain. It is often hard to specify a single trigger, as the relation is manifested by several. Moreover, it is expensive and time-consuming to anno-

tate training examples, especially with trigger designation, as is evident in prior annotation efforts such as GENIA (Kim et al., 2009). The realistic and widely adopted paradigm is to leverage indirect supervision, such as distant supervision (Craven and Kumlien, 1999; Mintz et al., 2009), where triggers will not be available at all.

Additionally, lexical and syntactic patterns signifying the relation will be quite sparse. To combat such sparsity, traditional feature-based approaches require extensive engineering effort. Unfortunately, this challenge becomes much more severe in cross-sentence extraction when the text spans multiple sentences.

To combat these challenges, we propose a general framework for cross-sentence, n-ary relation extraction based on graph LSTM. By learning a continuous representation for words and dependencies, LSTMs can handle sparsity effectively without requiring intense feature engineering. The graph formulation subsumes prior LSTM approaches based on chains or trees, and can incorporate rich linguistic analyses.

Our LSTM approach also opens up opportunities for joint learning with sub-relations. For example, the `Response` relation over d, g, v also implies a binary sub-relation over drug d and mutation v, with the gene underspecified. Even with distant supervision, supervision signal for n-ary relations will likely be sparser than their binary sub-relations. Our approach makes it very easy for multi-task learning with both the n-ary relations and their sub-relations.

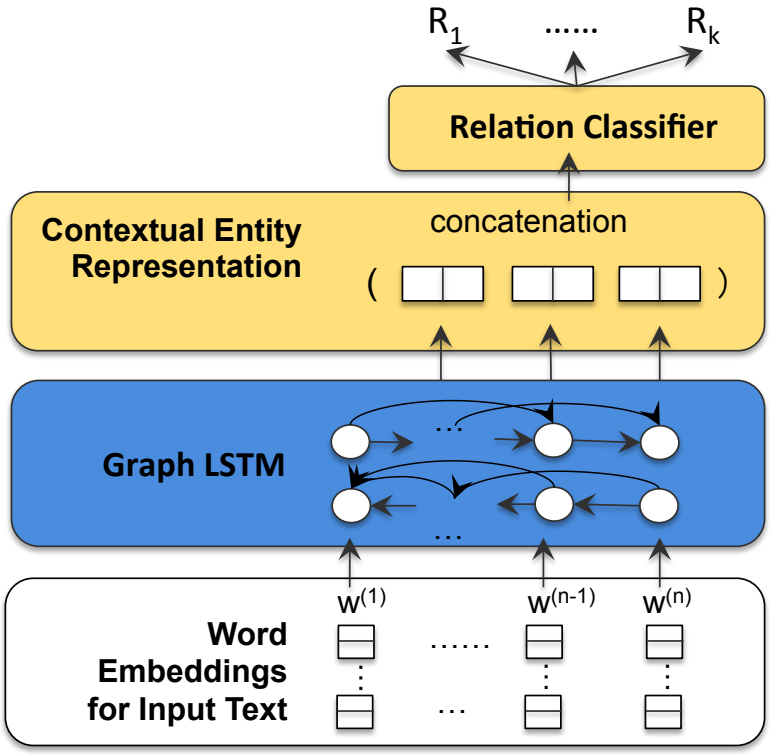


Figure 5.1: Our general architecture for cross-sentence n-ary relation extraction based on graph LSTM.

5.3 Graph LSTM

Learning a continuous representation can be effective for dealing with lexical and syntactic sparsity. For sequential data such as text, a popular choice is recurrent neural network (RNN), which resembles a hidden Markov model (HMM), except that discrete hidden states are replaced with continuous vectors, and emission and transition probabilities with neural networks. Conventional RNNs with sigmoid units suffer from gradient diffusion or explosion, making training very difficult (Bengio et al., 1994; Pascanu et al.,

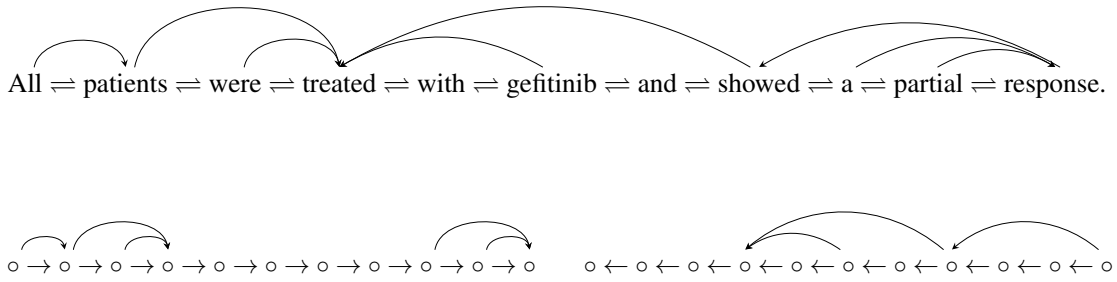


Figure 5.2: The graph LSTM used in this chapter. The document graph (top) is partitioned into two directed acyclic graphs (bottom); the graph LSTM is constructed by a forward pass (Left to Right) followed by a backward pass (Right to Left). Note that information goes from dependency child to parent.

2013). Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) combats these problems by using a series of gates (input, forget and output) to avoid amplifying or suppressing gradients during backprop. Consequently, LSTMs are much more effective in capturing long-distance dependencies, and have been applied to a variety of NLP tasks. However, most approaches are based on linear chains and only explicitly model the linear context, which ignores a variety of linguistic analyses, such as syntactic dependencies, coreference, and discourse relations.

In this section, we propose a general framework for cross-sentence, n-ary relation extraction by generalizing LSTMs to graphs. While there is some prior work on learning tree LSTMs (Tai et al., 2015; Miwa and Bansal, 2016), to the best of our knowledge, graph LSTMs have not been applied to any NLP task yet. Figure 5.1 shows the architecture of

our approach. The input layer is the word embedding of input text. Next is the graph LSTM which learns a contextual representation for each word. We take the hidden layer h produced by graph LSTM as this contextual representation. We assume the entity types and boundaries are given, for the entities in question, their contextual representations[3] are concatenated and become the only input to the relation classifiers. The layers are trained jointly with backpropagation. Our framework is agnostic to the choice of composing the contextual entity representation and the choice of classifiers. We use mean pooling for the entity representations from multi-word entities, and logistic regression classifier in this pilot study. Jointly designing the strategy of composing the entity representations and the choice of classifiers with graph LSTM would be interesting future work.

At the core of the graph LSTM is a *document graph* that captures various dependencies among the input words. By choosing what dependencies to include in the document graph, our approach naturally subsumes linear-chain or tree LSTMs.

Compared to conventional LSTMs, the graph formulation presents new challenges. Due to potential loops in the graph, a straightforward way to conduct backpropagation might require many iterations to reach a fixed point. Moreover, in the presence of a potentially large number of edge types (adjacent-word, syntactic dependency, etc.), parametrization becomes a key problem.

In the remainder of this section, we first introduce the document graph and show how to

[3]The representation of a multi-word entity is the average of representations of its words.

conduct backpropagation in graph LSTM. We then discuss two strategies for parametrizing the recurrent units. Finally, we show how to conduct multi-task learning with our approach.

5.3.1 Document Graph

To model a variety of dependencies from linguistic analysis at our disposal, we follow Quirk and Poon (2016) and introduce a *document graph* to capture intra- and inter-sentential dependencies. A document graph is a graph with nodes representing words and edges representing linear context (adjacent words), syntactic dependencies, coreference (Lee et al., 2013), and discourse relations (Xue et al., 2015). Figure 1.1 shows the document graph for our running example that suggests tumors with *L858E* mutation in *EGFR* gene responds to the drug *gefitinib*.

The document graph is the base for constructing the graph LSTM. By excluding all dependencies other than adjacent words, we get back the linear-chain LSTM. Similarly, other prior approaches can be replicated in our framework by limiting the graph to the shortest dependency path or the parse tree.

5.3.2 Backpropagation in Graph LSTM

Conventional LSTMs are essentially very deep feed-forward neural networks. For example, a left-to-right linear LSTM has a hidden vector for each word, which is generated by a neural network (recurrent unit) that takes as input the embedding of the given word and the hidden vector of the previous word. In discriminative learning, these hidden vectors then serve as input for the end classifiers, from which the gradients are backpropagated through the whole network.

Generalizing such a strategy to graphs with cycles typically requires unrolling recurrence for a number of steps, as in Graph Neural Network (Scarselli et al., 2009) or Gated Graph Sequence Neural Network (Li et al., 2016). Essentially, a copy of the graph is created for each step that serves as input for the next. The result is a feed-forward neural network through time, and backpropagation is conducted accordingly.

In principle, we could adopt the same strategy in our graph LSTM. Effectively, gradients are backpropagated in a similar fashion as in loopy belief propagation (LBP). However, this makes learning much more expensive as each update step requires multiple iterations of backpropagation. Moreover, the loopy backpropagation could suffer from similar problems in LBP, such as oscillation or failure to converge.

We observe that dependencies such as coreference and discourse relations are generally sparse, so the backbone of a document graph consists of the linear chain and the syntac-

tic dependency tree. As in belief propagation, such structures can be leveraged to make backpropagation more efficient by replacing the synchronous update, as in the unrolling strategy, with asynchronous update, as in linear-chain LSTM. This opens up opportunities for a variety of strategies in ordering backpropagation updates.

In this chapter, we use a simple strategy that performed quite well in preliminary experiments, and leave further exploration in future work. Specifically, we partition the document graph into two directed acyclic graphs (DAGs). One DAG contains the left-to-right linear chain, as well as other forward-pointing dependencies. The other DAG covers the right-to-left linear chain and the backward-pointing dependencies. Figure 5.2 illustrates this strategy. Effectively, we partition the original graph into the forward pass (left-to-right), followed by the backward pass (right-to-left), and construct the LSTM accordingly. When the document graph only contains the linear chain, the graph LSTM degenerates into the bi-direction LSTM (BiLSTM).

5.3.3 Topological Order

Although we only explored decomposing a document graph into a forward DAG and a backward DAG in this chapter, there are many different ways to decompose the document graphs. As long as one can define a topological order for all the nodes in the graph, any cyclic graph without self loop can be decomposed into two DAGs. Specifically, all the

arcs pointing from a predecessor to a successor in the topological order can be arranged in one DAG, while all other arcs being arranged in another DAG. In our case, the topological order is straightforwardly defined according to the natural sequence order in the writing system. Another way that might be plausible is to order the words according to the syntactic tree, either from root to leaves or the reverse. We did not obtain this order because it performed worse in our preliminary experiments.

For the applications in other domains, if there are domain knowledge that help define better topological orders, one can also decompose the graph according to those orders.

5.3.4 The Basic Recurrent Propagation Unit

A standard LSTM unit consists of an input vector (word embedding), a memory cell and an output vector (contextual representation), as well as several gates. The *input gate* and *output gate* control the information flowing into and out of the cell, where the *forget gate* can optionally remove information from the recurrent connection to a precedent unit.

In linear-chain LSTMs, each unit contains only one forget gate, as it has only one direct precedent (i.e., the adjacent-word edge pointing to the previous word). In graph LSTM, however, a unit may have several precedents, including connections to the same word via different edges. We thus introduce a forget gate for each precedent, similar to the approach taken by Tai et al. (2015) for tree LSTM.

Unlike in linear-chain LSTM, we can potentially have many edge types besides from adjacent-word. This opens up many possibilities for parametrization over the edge types and we explore two of them.

Full Parametrization Our first proposal simply introduces a different set of parameters for each edge type, with computation specified below.

$$i_t = \sigma(W_i x_t + \frac{1}{J}\sum\nolimits_{j \in P(t)} U_i^{m(j)} h_j + b_i) \tag{5.1}$$

$$f_{tj} = \sigma(W_f x_t + U_f^{m(j)} h_j + b_f) \tag{5.2}$$

$$o_t = \sigma(W_o x_t + \frac{1}{J}\sum\nolimits_{j \in P(t)} U_o^{m(j)} h_j + b_o) \tag{5.3}$$

$$\tilde{c_t} = \tanh(W_c x_t + \frac{1}{J}\sum\nolimits_{j \in P(t)} U_c^{m(j)} h_j + b_c) \tag{5.4}$$

$$c_t = i_t \odot \tilde{c_t} + \frac{1}{J}\sum\nolimits_{j \in P(t)} f_{tj} \odot c_j \tag{5.5}$$

In each of the gating functions (Equations 5.1, 5.2, 5.3), x_t is the input word vector, W's are the weight matrices on x_t and b's are the bias term vectors, just as in traditional LSTMs. The major differences lie in the recurrence term (the middle term): for the *input* and *output* gates, they rely on all the predecessors to decide their values, and each predecessor j is associated with a typed weight matrix $U^{m(j)}$, where $m(j)$ denotes a type mask. The computation of each forget gate only depends on the predecessor with which the gate is associated.

Edge-Type Embedding Full parameterization is straightforward, but it requires a lot

of parameters when there are many edge types. For example, there are dozens of syntactic edge types, each corresponding to a Stanford dependency label. To reduce the number of parameters and leverage potential correlation among the edge types, we learned a low-dimensional embedding of the edge types, and conducted an outer product of the previous hidden vector and the edge-type embedding to generate a "typed hidden representation", which is a matrix. The new computation is as follows:

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + \frac{1}{J}\sum\nolimits_{j \in P(t)} U_i \times_T (h_j \otimes e_j) + b_i) \\
f_{tj} &= \sigma(W_f x_t + U_f \times_T (h_j \otimes e_j) + b_f) \\
o_t &= \sigma(W_o x_t + \frac{1}{J}\sum\nolimits_{j \in P(t)} U_o \times_T (h_j \otimes e_j) + b_o) \\
\tilde{c_t} &= \tanh(W_c x_t + \frac{1}{J}\sum\nolimits_{j \in P(t)} U_c \times_T (h_j \otimes e_j) + b_c) \\
c_t &= i_t \odot \tilde{c_t} + \frac{1}{J}\sum\nolimits_{j \in P(t)} f_{tj} \odot c_j
\end{aligned}
$$

U's are now $h \times h \times d$ tensors (h is the dimension of the hidden vector and d is the dimension for edge-type embedding), and $(h_j \otimes e_j)$ produces a $h \times d$ matrix. $\times_T$ denotes a tensor dot product defined as $T \times_T A = \sum_d (T_{:,:,d} \cdot A_{:,d})$, which produces an h-dimensional vector. The edge-type embedding is jointly trained with the rest.

5.3.5 Comparison with Prior LSTM Approaches

The main advantages of a graph formulation are its generality and flexibility. As seen in Section 3.1, linear-chain LSTMs are a special case when the document graph is the linear chain of adjacent words. Similarly, Tree LSTMs (Tai et al., 2015) are a special case when the document graph is the parse tree.

In graph LSTMs, the encoding of linguistic knowledge is factored from the backpropagation strategy (Section 3.2), making it much more flexible, including introducing cycles. For example, Miwa and Bansal (2016) conducted joint entity and binary relation extraction by stacking a LSTM for relation extraction on top of another LSTM for entity recognition. In graph LSTMs, the two can be combined seamlessly using a document graph comprising both the word-adjacency chain and the dependency path between the two entities.

The document graph can also incorporate other linguistic information. For example, coreference and discourse parsing are intuitively relevant for cross-sentence relation extraction. Although existing systems have not yet been shown to improve cross-sentence relation extraction (Quirk and Poon, 2017), it remains an important future direction to explore incorporating such analyses, especially after adapting them to the biomedical domains (Bell et al., 2016).

5.3.6 Multi-task Learning with Sub-relations

Multi-task learning has been shown to be beneficial in training neural networks (Collobert and Weston, 2008; Peng and Dredze, 2017). By learning a layer of contextual entity representations, our framework makes it straightforward to conduct multi-task learning. All it takes is to add classifiers for the sub-relations. The classifiers for the n-ary relation and its sub-relations share the same graph LSTM and word embedding and can potentially help each other by pooling their supervision signals.

In our experiments for the molecular tumor board domain, we explore this paradigm by jointly learning both the ternary relation (drug-gene-mutation) and the binary sub-relation (drug-mutation). Experiment results show that this provides significant gains in both tasks.

5.4 Implementation Details

We implemented our methods using the Theano library (Theano Development Team, 2016). We used logistic regression for our relation classifiers. Hyper parameters were set based on preliminary experiments on a small development dataset. Training was done using mini-batched stochastic gradient descent (SGD) with batch size 8. We used a learning rate of 0.02 and trained for at most 30 epochs, with early stopping based on development data (Caruana et al., 2001; Graves et al., 2013). The dimension for the hidden vectors in

LSTM units was set to 150, and the dimension for the edge-type embedding was set to 3. The word embeddings were initialized with the publicly available 100-dimensional GloVe word vectors trained on 6 billion words from Wikipedia and web text[4] (Pennington et al., 2014). Other model parameters were initialized with random samples drawn uniformly from the range $[-1, 1]$.

In multi-task training, we alternated among all tasks, each time passing through all data for one task[5], and updating the parameters accordingly. This was repeated for 30 epochs.

5.5 Domain: Molecular Tumor Boards

Our main experiments focus on extracting ternary interactions over drugs, genes and mutations, which is important for molecular tumor boards. A drug-gene-mutation interaction is broadly construed as an association between the drug efficacy and the mutation in the given gene. There is no annotated dataset for this problem. However, due to the importance of such knowledge, oncologists have been painstakingly curating known relations from reading papers. Such a manual approach cannot keep up with the rapid growth of the research literature, and the coverage is generally sparse and not up to date. However, the curated knowledge can be used for distant supervision.

[4] `http://nlp.stanford.edu/projects/glove/`

[5] However, drug-gene pairs have much more data, so we sub-sampled the instances down to the same size as the main n-ary relation task.

5.5.1 Datasets

We obtained biomedical literature from PubMed Central[6], consisting of approximately *one million* full-text articles as of 2015. Note that only a fraction of papers contain knowledge about drug-gene-mutation interactions. Extracting such knowledge from the vast body of biomedical papers is exactly the challenge. As we will see in later subsections, distant supervision enables us to generate a sizable training set from a small number of manually curated facts, and the learned model was able to extract orders of magnitude more facts. In future work, we will explore incorporating more known facts for distant supervision and extracting from more full-text articles.

We conducted tokenization, part-of-speech tagging, and syntactic parsing using SPLAT (Quirk et al., 2012), and obtained Stanford dependencies (de Marneffe et al., 2006) using Stanford CoreNLP (Manning et al., 2014). We used the entity taggers from Literome (Poon et al., 2014) to identify drug, gene and mutation mentions.

We used the Gene Drug Knowledge Database (GDKD) (Dienstmann et al., 2015) and the Clinical Interpretations of Variants In Cancer (CIVIC) knowledge base[7] for distant supervision. They contain 517 and 1001 unique (drug, gene, mutation) tuples, respectively, for which there are known drug-response relations. The knowledge bases distinguish fine-grained interaction types, which we do not use in this chapter.

[6] `http://www.ncbi.nlm.nih.gov/pmc/`
[7] `http://civic.genome.wustl.edu`

5.5.2 Distant Supervision

After identifying drug, gene and mutation mentions in the text, co-occurring triples with known interactions were chosen as positive examples. However, unlike the single-sentence setting in standard distant supervision, care must be taken in selecting the candidates. Since the triples can reside in different sentences, an unrestricted selection of text spans would risk introducing many obviously wrong examples. We thus followed Quirk and Poon (2017) in restricting the candidates to those occurring in a *minimal span*, i.e., we retain a candidate only if is no other co-occurrence of the same entities in an overlapping text span with a smaller number of consecutive sentences. Furthermore, we avoid picking unlikely candidates where the triples are far apart in the document. Specifically, we considered entity triples within K consecutive sentences, ignoring paragraph boundaries. $K = 1$ corresponds to the baseline of extraction within single sentences. We explored $K \leq 3$, which captured a large fraction of candidates without introducing many unlikely ones.

Only 59 distinct drug-gene-mutation triples from the knowledge bases were matched in the text. Even from such a small set of unique triples, we obtained 3,462 ternary relation instances that can serve as positive examples. For multi-task learning, we also considered drug-gene and drug-mutation sub-relations, which yielded 137,469 drug-gene and 3,192 drug-mutation relation instances as positive examples.

Model	Single-Sent.	Cross-Sent.
Feature-Based	74.7	77.7
CNN	77.5	78.1
BiLSTM	75.3	80.1
Graph LSTM - EMBED	76.5	80.6
Graph LSTM - FULL	**77.9**	**80.7**

Table 5.1: Average test accuracy in five-fold cross-validation for drug-gene-mutation ternary interactions. Feature-Based used the best performing model in (Quirk and Poon, 2017) with features derived from shortest paths between all entity pairs.

We generate negative examples by randomly sampling co-occurring entity triples without known interactions, subject to the same restrictions above. We sampled the same number as positive examples to obtain a balanced dataset[8].

5.5.3 Automatic Evaluation

To compare the various models in our proposed framework, we conducted five-fold cross-validation, treating the positive and negative examples from distant supervision as gold annotation. To avoid train-test contamination, all examples from a document were

[8]We released the code and the dataset at `http://hanover.azurewebsites.net`. We also give the best hyper-parameter settings in the script we provided for running out code.

Model	**Single-Sent.**	**Cross-Sent.**
Feature-Based	73.9	75.2
CNN	73.0	74.9
BiLSTM	73.9	76.0
BiLSTM-Shortest-Path	70.2	71.7
Tree LSTM	**75.9**	75.9
Graph LSTM-EMBED	74.3	76.5
Graph LSTM-FULL	75.6	**76.7**

Table 5.2: Average test accuracy in five-fold cross-validation for drug-mutation binary relations, with an extra baseline using a BiLSTM on the shortest dependency path (Xu et al., 2015b; Miwa and Bansal, 2016).

assigned to the same fold. Since our datasets are balanced by construction, we simply report average test accuracy on held-out folds. Obviously, the results could be noisy (e.g., entity triples not known to have an interaction might actually have one), but this evaluation is automatic and can quickly evaluate the impact of various design choices.

We evaluated two variants of graph LSTMs: "Graph LSTM-FULL" with full parametrization and "Graph LSTM-EMBED" with edge-type embedding. We compared graph LSTMs with three strong baseline systems: a well-engineered feature-based classifier (Quirk and

Poon, 2017), a convolutional neural network (CNN) (Zeng et al., 2014; Santos et al., 2015; Wang et al., 2016), and a bi-directional LSTM (BiLSTM). Following Wang et al. (2016), we used input attention for the CNN and a input window size of 5. Quirk and Poon (2017) only extracted binary relations. We extended it to ternary relations by deriving features for each entity pair (with added annotation to signify the two entity types), and pooling the features from all pairs.

For binary relation extraction, prior syntax-aware approaches are directly applicable. So we also compared with a state-of-the-art tree LSTM system (Miwa and Bansal, 2016) and a BiLSTM on the shortest dependency path between the two entities (BiLSTM-Shortest-Path) (Xu et al., 2015b).

Table 5.1 shows the results for cross-sentence, ternary relation extraction. All neural-network based models outperformed the feature-based classifier, illustrating their advantage in handling sparse linguistic patterns without requiring intense feature engineering. All LSTMs significantly outperformed CNN in the cross-sentence setting, verifying the importance in capturing long-distance dependencies.

The two variants of graph LSTMs perform on par with each other, though Graph LSTM-FULL has a small advantage, suggesting that further exploration of parametrization schemes could be beneficial. In particular, the edge-type embedding might improve by pretraining on unlabeled text with syntactic parses.

Both graph variants significantly outperformed BiLSTMs ($p < 0.05$ by McNemar's chi-square test), though the difference is small. This result is intriguing. In Quirk and Poon (2017), the best system incorporated syntactic dependencies and outperformed the linear-chain variant (Base) by a large margin. So why didn't graph LSTMs make an equally substantial gain by modeling syntactic dependencies?

One reason is that linear-chain LSTMs can already captured some of the long-distance dependencies available in syntactic parses. BiLSTMs substantially outperformed the feature-based classifier, even without explicit modeling of syntactic dependencies. The gain cannot be entirely attributed to word embedding as LSTMs also outperformed CNNs.

Another reason is that syntactic parsing is less accurate in the biomedical domain. Parse errors confuse the graph LSM learner, limiting the potential for gain. In Section 6, we show supporting evidence in a domain when gold parses are available.

We also reported accuracy on instances within single sentences, which exhibited a broadly similar set of trends. Note that single-sentence and cross-sentence accuracies are not directly comparable, as the test sets are different (one subsumes the other).

We conducted the same experiments on the binary sub-relation between drug-mutation pairs. Table 5.2 shows the results, which are similar to the ternary case: Graph LSTM-FULL consistently performed the best for both single sentence and cross-sentence instances. BiLSTMs on the shortest path substantially underperformed BiLSTMs or graph

	Drug-Gene-Mut.	**Drug-Mut.**
BiLSTM	80.1	76.0
+Multi-task	**82.4**	78.1
Graph LSTM	80.7	76.7
+Multi-task	82.0	**78.5**

Table 5.3: Multi-task learning improved accuracy for both BiLSTMs and Graph LSTMs.

LSTMs, losing between 4-5 absolute points in accuracy, which could be attributed to the lower parsing quality in the biomedical domain. Interestingly, the state-of-the-art tree LSTMs (Miwa and Bansal, 2016) also underperformed graph LSTMs, even though they encoded essentially the same linguistic structures (word adjacency and syntactic dependency). We attributed the gain to the fact that Miwa and Bansal (2016) used separate LSTMs for the linear chain and the dependency tree, whereas graph LSTMs learned a single representation for both.

To evaluate whether joint learning with sub-relations can help, we conducted multi-task learning using Graph LSTM-FULL to jointly train extractors for both the ternary interaction and the drug-mutation, drug-gene sub-relations. Table 5.3 shows the results. Multi-task learning resulted in a significant gain for both the ternary interaction and the drug-mutation interaction. Interestingly, the advantage of graph LSTMs over BiLSTMs

is reduced with multi-task learning, suggesting that with more supervision signal, even linear-chain LSTMs can learn to capture long-range dependencies that are were made evident by parse features in graph LSTMs. Note that there are many more instances for drug-gene interaction than others, so we only sampled a subset of comparable size. Therefore, we do not evaluate the performance gain for drug-gene interaction, as in practice, one would simply learn from all available data, and the sub-sampled results are not competitive.

We included coreference and discourse relations in our document graph. However, we didn't observe any significant gains, similar to the observation in Quirk and Poon (2017). We leave further exploration to future work.

5.5.4 PubMed-Scale Extraction (Absolute Recall)

Our ultimate goal is to extract all knowledge from available text. However, it is hard to measure the recall on such domains since the true total facts number is not available. We thus retrained our model using the best system from automatic evaluation (i.e., Graph LSTM-FULL) on all available data, and used the trained model to extract relations from all PubMed Central articles. We report the number of distinct relations we extracted as a measure of *absolute* recall.

Table 5.4 shows the number of candidates and extracted interactions. With as little

as 59 unique drug-gene-mutation triples from the two databases[9], we learned to extract orders of magnitude more unique interactions. The results also highlight the benefit of cross-sentence extraction, which yields 3 to 5 times more relations than single-sentence extraction. Note that the manually curated database GDKD and CIVIC contained less than 1500 known relations between drug, gene, and mutation, and they took the community significant efforts to build.

Table 5.5 conducts a similar comparison on unique number of drugs, genes, and mutations. Again, machine reading covers far more unique entities, especially with cross-sentence extraction.

5.5.5 Manual Evaluation

Our automatic evaluations are useful for comparing competing approaches, but may not reflect the true classifier precision as the labels are noisy. Therefore, we randomly sampled extracted relation instances and asked three researchers knowledgeable in precision medicine to evaluate their correctness. For each instance, the annotators were presented with the provenance: sentences with the drug, gene, and mutation highlighted. The annotators determined in each case whether this instance implied that the given entities

[9]There are more in the databases, but these are the only ones for which we found matching instances in the text. In future work, we will explore various ways to increase the number, e.g., by matching underspecified drug classes to specific drugs.

	Single-Sent.	**Cross-Sent.**
Candidates	10,873	57,033
$p \geq 0.5$	1,408	4,279
$p \geq 0.9$	530	1,461
GDKD + CIVIC	59	

Table 5.4: Numbers of unique drug-gene-mutation interactions extracted from PubMed Central articles, compared to that from manually curated KBs used in distant supervision. p signifies output probability.

were related. Note that evaluation does not attempt to identify whether the relationships are true or replicated in follow-up papers; rather, it focuses on whether the relationships are entailed by the text.

We focused our evaluation efforts on the cross-sentence ternary-relation setting. We considered three probability thresholds: 0.9 for a high-precision but potentially low-recall setting, 0.5, and a random sample of all candidates. In each case, 150 instances were selected for a total of 450 annotations. A subset of 150 instances were reviewed by two annotators, and the inter-annotator agreement was 88%.

Table 5.6 shows that the classifier indeed filters out a large portion of potential candidates, with estimated instance accuracy of 64% at the threshold of 0.5, and 75% at 0.9.

	Drug	**Gene**	**Mut.**
GDKD + CIVIC	16	12	41
Single-Sent. ($p \geq 0.9$)	68	228	221
Single-Sent. ($p \geq 0.5$)	93	597	476
Cross-Sent. ($p \geq 0.9$)	103	512	445
Cross-Sent. ($p \geq 0.5$)	144	1344	1042

Table 5.5: Numbers of unique drugs, genes and mutations in extraction from PubMed Central articles, in comparison with that in the manually curated Gene Drug Knowledge Database (GDKD) and Clinical Interpretations of Variants In Cancer (CIVIC) used for distant supervision. p signifies output probability.

Interestingly, LSTMs are effective at screening out many entity mention errors, presumably because they include broad contextual features.

5.6 Domain: Genetic Pathways

We also conducted experiments on extracting genetic pathway interactions using the GENIA Event Extraction dataset (Kim et al., 2009). This dataset contains gold syntactic parses for the sentences, which offered a unique opportunity to investigate the impact of

	Precision	Entity Error	Relation Error
Random	17%	36%	47%
$p \geq 0.5$	64%	7%	29%
$p \geq 0.9$	75%	1%	24%

Table 5.6: Sample precision of drug-gene-mutation interactions extracted from PubMed Central articles. p signifies output probability.

syntactic analysis on graph LSTMs. It also allowed us to test our framework in supervised learning.

The original shared task evaluated on complex, nested events for nine event types, many of which are unary relations (Kim et al., 2009). Following Poon et al. (2015), we focused on gene regulation and reduced it to binary-relation classification for head-to-head comparison. We followed their experimental protocol by sub-sampling negative examples to be about three times of positive examples.

Since the dataset is not entirely balanced, we reported precision, recall, and F1. We used our best performing graph LSTM from the previous experiments. By default, automatic parses were used in the document graphs, whereas in Graph LSTM (GOLD), gold parses were used instead. Table 5.7 shows the results. Once again, despite the lack of

Model	Precision	Recall	F1
Poon et al. (2015)	37.5	29.9	33.2
BiLSTM	37.6	29.4	33.0
Graph LSTM	41.4	30.0	34.8
Graph LSTM (GOLD)	**43.3**	**30.5**	**35.8**

Table 5.7: GENIA test results on the binary relation of gene regulation. Graph LSTM (GOLD) used gold syntactic parses in the document graph.

intense feature engineering, linear-chain LSTMs performed on par with the feature-based classifier (Poon et al., 2015). Graph LSTMs exhibited a more commanding advantage over linear-chain LSTMs in this domain, substantially outperforming the latter ($p < 0.01$ by McNemar's chi-square test). Most interestingly, graph LSTMs using gold parses significantly outperformed that using automatic parses, suggesting that encoding high-quality analysis is particularly beneficial.

5.7 Related Work

There are many prior work on relation extraction in NLP. However, most work on relation extraction has been applied to binary relations of entities in a single sentence. We first review relevant work on the single-sentence binary relation extraction task, and then

mention related work on n-ary and cross-sentence relation extraction.

5.7.1 Binary relation extraction

The traditional feature-based methods rely on carefully designed features to learn good models, and often integrate diverse sources of evidence such as word sequences and syntax context (Kambhatla, 2004; Zhou et al., 2005; Boschee et al., 2005; Suchanek et al., 2006; Chan and Roth, 2010; Nguyen and Grishman, 2014). The kernel-based methods design various subsequence or tree kernels (Mooney and Bunescu, 2005; Bunescu and Mooney, 2005; Qian et al., 2008) to capture structured information. Recently, models based on neural networks have advanced the state of the art by automatically learning powerful feature representations (Xu et al., 2015a; Zhang et al., 2015; Santos et al., 2015; Xu et al., 2015b, 2016). Some explored combine neural features with traditional features (Zeng et al., 2014; Yu et al., 2014, 2015; Nguyen and Grishman, 2015a; Gormley et al., 2015) or the kernel methods (Nguyen et al., 2015) to improve the performance.

Most neural architectures can be summarized as Figure 5.1, where there is a core representation learner (blue) that takes word embeddings as input and produces contextual entity representations. Such representations are then taken by relation classifiers to produce the final predictions. Effectively representing sequences of words, both convolutional (Zeng et al., 2014; Wang et al., 2016; Santos et al., 2015) and RNN-based architec-

tures (Zhang et al., 2015; Socher et al., 2012; Cai et al., 2016) have been successful. Most of these have focused on modeling either the surface word sequences or the hierarchical syntactic structure. Miwa and Bansal (2016) proposed an architecture that benefits from both types of information, using a surface sequence layer, followed by a dependency-tree sequence layer.

5.7.2 N-ary relation extraction

Early work on extracting relations between more than two arguments has been done in MUC-7, with a focus on fact/event extraction from news articles (Chinchor, 1998). Semantic role labeling in the Propbank (Palmer et al., 2005) or FrameNet (Baker et al., 1998) style are also instances of n-ary relation extraction, with extraction of events expressed in a single sentence. McDonald et al. (2005) extract n-ary relations in a bio-medical domain, by first factoring the n-ary relation into pair-wise relations between all entity pairs, and then constructing maximal cliques of related entities. Recently, neural models have been applied to semantic role labeling (FitzGerald et al., 2015; Roth and Lapata, 2016). These works learned neural representations by effectively decomposing the n-ary relation into binary relations between the predicate and each argument, by embedding the dependency path between each pair, or by combining features of the two using a feed-forward network. Although some re-ranking or joint inference models have been employed, the

representations of the individual arguments do not influence each other. In contrast, we propose a neural architecture that jointly represents n entity mentions, taking into account long-distance dependencies and inter-sentential information.

5.7.3 Cross-sentence relation extraction

Several relation extraction tasks have benefited from cross-sentence extraction, including MUC fact and event extraction (Swampillai and Stevenson, 2011), record extraction from web pages (Wick et al., 2006), extraction of facts for biomedical domains (Yoshikawa et al., 2011), and extensions of semantic role labeling to cover implicit inter-sentential arguments (Gerber and Chai, 2010). These prior works have either relied on explicit coreference annotation, or on the assumption that the whole document refers to a single coherent event, to simplify the problem and reduce the need for powerful representations of multi-sentential contexts of entity mentions. Recently, cross-sentence relation extraction models have been learned with distant supervision, and used integrated contextual evidence of diverse types without reliance on these assumptions (Quirk and Poon, 2016), but that work focused on binary relations only and explicitly engineered sparse indicator features.

5.7.4 Relation extraction using distant supervision

Distant supervision has been applied to extraction of binary (Mintz et al., 2009; Poon et al., 2015) and n-ary (Reschke et al., 2014; Li et al., 2015) relations, traditionally using hand-engineered features. Neural architectures have recently been applied to distantly supervised extraction of binary relations (Zeng et al., 2015). Our work is the first to propose a neural architecture for n-ary relation extraction, where the representation of a tuple of entities is not decomposable into independent representations of the individual entities or entity pairs, and which integrates diverse information from multi-sentential context. To utilize training data more effectively, we show how multi-task learning for component binary sub-relations can improve performance. Our learned representation combines information sources within a single sentence in a more integrated and generalizable fashion than prior approaches, and can also improve performance on single-sentence binary relation extraction.

5.8 Conclusion

We explore a general framework for cross-sentence n-ary relation extraction based on graph LSTMs. The graph formulation subsumes linear-chain and tree LSTMs and makes it easy to incorporate rich linguistic analysis when learning representations. Experiments

on a typical low resource IE tasks: biomedical domains relation extraction showed that encoding rich linguistic knowledge in learned representations provides consistent gain.

While there is much room to improve in both recall and precision, our results indicate that machine reading can already be useful in precision medicine. In particular, automatically extracted facts (Section 5.4) can serve as candidates for manual curation. Instead of scanning millions of articles to curate from scratch, human curators would just quickly vet thousands of extractions. The errors identified by curators offer direct supervision to the machine reading system for continuous improvement. Therefore, the most important thing is to attain high recall and reasonable precision, which our current models are already capable of.

There are many future directions that worth exploring:

- Improved discourse modeling in the low-resource domain to enhance the cross-sentence extraction;
- Joint learning of entity mention detection / entity linking and relation extraction;
- Applications of the cross-sentence N-ary relation extraction to other domains, which will also be low-resource IE scenarios.

Chapter 6

Joint Entity and Relation Extraction

In the previous chapter, we consider a standard setting for relation extraction, where we assume entity boundaries and entity types are given. However, when the annotations are scarce, such a setting is unrealistic for two main reasons: 1) manually-labeled entities boundaries and types are costly to obtain; 2) an automatic entity detector often cannot be directly applied. For example, the Stanford NER tagger, a popular named entity recognizer, cannot detect named entities in the biomedical domain as it is not trained to identity technical terms such as proteins, drugs, genes, and mutation names.

In this chapter, we explore joint entity and relation extraction in an end-to-end setting. Given a piece of raw text, the system detects the entities and relations between them *jointly*. Such a setting has been widely studied (Roth and Yih, 2007; Li and Ji, 2014; Miwa

and Sasaki, 2014; Miwa and Bansal, 2016). However, previous works often assume sufficient annotations are available. In this chapter, we investigate this task in a low resource setting. We study different strategies of making joint prediction and making use of shared representations. We also propose a novel graphical model with neural factor that making simultaneously decisions and learning shared representations for entity and relation extraction. We seek the answers to the following questions:

- Is it beneficial to share the representation learners for the entity component and the relation component in joint entity and relation extraction?
- What is the best learning and inference strategy for end-to-end entity and relation extraction models? We compare two different modeling ideas – 1) incremental inference and 2) simultaneous inference.

6.1 Joint Entity and Relation Extraction

Making joint entity and relation extraction task has been modeled as a structured prediction problem. Previous works have shown that a model training both tasks jointly (Roth and Yih, 2007; Li and Ji, 2014; Miwa and Sasaki, 2014) works better than a pipeline scheme, which trains models for entity detection and relation extraction separately. The pipeline model, which detects the entity boundary and type first and then predicts the re-

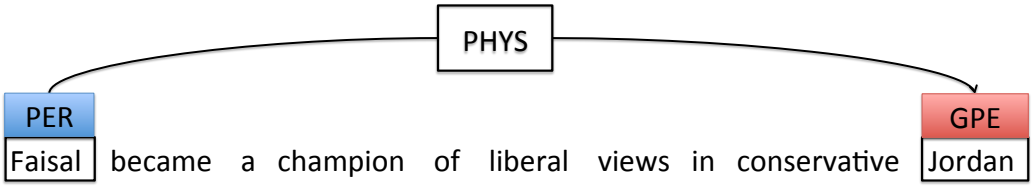


Figure 6.1: An example for joint entity and relation extraction. The physically located (PHY) relation between entities Faisal and Jordan helps identifying Jordan as a geo-political entity.

lation types, is prone to error propagation, while the joint training enables the information flow between the modules in the pipeline (Yu and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014; Miwa and Bansal, 2016; Li et al., 2017; Zheng et al., 2017).

Figure 6.1 shows a concrete example that to learn the *PHY (physically located)* relation between the entities *Faisal* and *Jordan*, it is important to have the information about the entity types (person and geo-political location, respectively); and conversely, the knowledge about the *PHY* relation between the entities (with strong indication from the context word "in") helps decide the entity types.

The above example demonstrates that an end-to-end relation extractor can be benefited by modeling the interdependencies between entity types and relation types. Various approaches have been proposed to take advantage of these interdependency (Roth and Yih, 2007; Li and Ji, 2014; Miwa and Sasaki, 2014; Miwa and Bansal, 2016; Katiyar and Cardie, 2017; Zheng et al., 2017). We categorize these models based on two perspec-

tives – 1) how the model makes joint predictions (incremental inference or simultaneous inference) 2) if the model learns shared representations of features for each module.

For the first perspective, **incremental inference** approach makes *sequential decisions* to incrementally build the joint prediction based on sub-decisions Li and Ji (2014); Miwa and Sasaki (2014). Formally, they have two modules: $p(\mathbf{y}^e|\mathbf{x})$ for entity and $p(\mathbf{y}^r|\hat{\mathbf{y}}^e, \mathbf{x})$ for relation, where $\hat{\mathbf{y}}^e = \arg\max_{\mathbf{y}^e \in \mathcal{Y}^e} p(\mathbf{y}^e|\mathbf{x})$ represents the best decision from the entity module. The decision process resembles pipeline models; however these sub-modules are trained jointly. The errors in relation extraction will provide information to the mention detection module during training, and thus improve both modules. In contrast, **simultaneous inference** approach Yu and Lam (2010) uses a graph to describe the inter-dependencies between entities and relations, and resolve their types *simultaneously*. This enables better interactions between the entity and relation modules and makes it easier to design global constraints and features to reduce the risk the incremental inference model faced that the entity module made bad decisions early on.

For the second perspective, since entity and relation are closely related to each other, it can be beneficial to share the representations (features) for each module and build a joint model through a multi-task learning framework. In this approach, the two tasks (partially) share the representation learners to produce representations for words and phrases as features, and pass them to each task specific model. Recent work on deep learning for joint

	Feature Based	**DNNs Based**
Incremental Inference	✓	✓
Simultaneous Inference	✓	This chapter

Table 6.1: An Overview of Work on End-to-end Entity and Relation Extraction.

entity and relation extraction (Miwa and Bansal, 2016; Katiyar and Cardie, 2017; Zheng et al., 2017) coincidentally all employed some parameter sharing strategy in their design of the architectures to share representations between entity and relation modules.

In this chapter, we emphasize the distinction of the *inference* and *representation sharing* aspects of joint models, and propose to combine the advantages of both. We propose a graphical model with neural-nets factors, which combines the benefits of graphical models for incorporating domain knowledge and make simultaneous decisions, and neural networks for automatically learning and flexibly sharing representations between the entity and relation modules. The inference problem is formulated as an integer linear programming (ILP), which can easily incorporate domain knowledge and global features as constraints to make *simultaneous inference*. We call our model *DNNs-ILP*. This joint framework better utilizes the interactions between the entity and relation modules with the ILP formulation, and also flexibly shares the representations for entities and relations with deep neural networks, thus yield robust models in the low resource setting. We also

conduct controlled experiments on the two aspects of the joint model.

This chapter makes the following contributions.

- A novel *DNNs-ILP* formulation that combines the benefit of joint inference and representation sharing for joint entity and relation extraction.
- Controlled experiments on the effects of incremental v.s. joint inference and representation sharing v.s. non-sharing for the proposed joint model.

6.2 Related Work

Work on joint entity and relation extraction can be roughly categorized in Table 6.1. The headers "Feature Base" and "DNNs Based" indicate whether the models are based on hand-engineer features, or automatically learned representations by deep neural networks; and "Incremental Inference" and "Simultaneous Inference" indicate whether the predictions are *sequentially* or *simultaneously* made. We are the first work that combines DNNs models with joint inference in a principled way.

The work that explored different inference strategies are mostly feature-based. Many work explored joint inference with separately pre-trained models [1]. The joint inference strategies including re-ranking (Ji and Grishman, 2005), integer linear programming (ILP)

[1]The models for entity detection and relation extraction are separately trained, and the joint inference is performed during test time

(Roth and Yih, 2007), card-pyramid parsing (Kate and Mooney, 2010), and graphical models (Singh et al., 2013). Li and Ji (2014); Miwa and Sasaki (2014) proposed joint training incremental inference models using structured perceptron and table completing strategies respectively. The decision process resembles pipeline models, but they are joint model because the two modules are trained jointly. The errors in relation extraction will provide information to the mention detection module during training, and thus improve both modules. Yu and Lam (2010) proposed a simultaneous inference graphical model where the decisions (inference) for the entities and relations are made *together*. This enables better interactions between the entity and relation modules and makes it easier to design global constraints and features to reduce the risk the **incremental inference** model faced that the entity module made bad decisions early on.

More recently, Miwa and Bansal (2016); Katiyar and Cardie (2017) proposed neural network based end-to-end models. These models almost exclusively adopted the incremental inference strategy and focused on the exploration of different neural architectures and parameter sharing strategies to build the joint models.

We propose to combine the benefit of simultaneous inference *and* parameter sharing for end-to-end entity and relation extraction. The closest previous work to ours is (Zheng et al., 2017), which is also a DNNs based simultaneous inference and joint learning model. However, their formulation of a sequence tagging schema only allowed an entity to involve

in one relation, because their tags were a conjoint of entity tags and relation tags, and each token was only assigned one tag. On contrary, our formulation of assigning each token an entity label, and each pair of tokens a relation tag, which addresses the problem of one entity for multiple relations. The inference can be efficiently conducted with ILP formulations.

6.3 Model

We formulate the joint entity and relation extraction as a structured prediction problem. Each token will be assign an entity tag (we use BILOU tagging schema as introduced in Chapter 2.2.2.2), and each pair of token will be assign a relation tag. The relation tag and the entity tags for the corresponding tokens are correlated. The whole system can be viewed as a high-order CRF model with neural-nets factors, where the factors between inputs and output tags are learned by deep neural networks, and the dependencies between the tags does not have first-order Markov assumption.

6.3.1 Model Overview

Figure 6.2 demonstrates the factor graph representation for our *DNNs-ILP* model for the sentence "U.S. intelligence officials now believe ... report." There are three types of

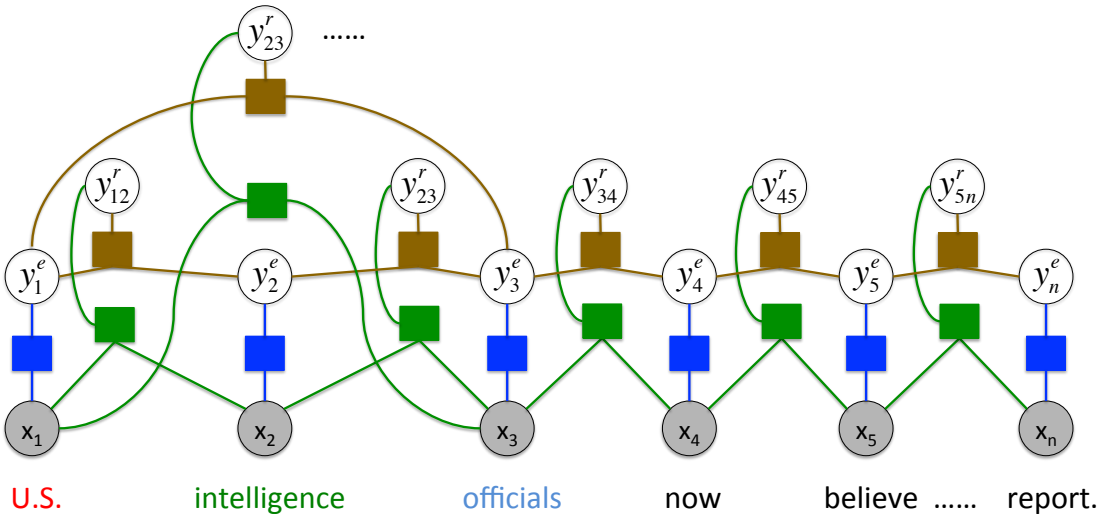


Figure 6.2: The factor graph representation for our structured tagging model. Some of the relation factors are omitted for simplicity. The joint probability that this factor graph represented is shown in Equation 6.1.

factors: the first type correlates the entity tags y^e with the inputs $\mathbf{x}$ (the blue factors in the figure), we call it the "*entity factor*"; the second type correlates the relation tags y^r with the inputs $\mathbf{x}$ (the green factors in the figure), we call it the "*relation factor*"; the third type links entity tags with relation tags (the brown factors in the figure), we call it the "*entity relation factor*". Each entity tag associates with two factor, one *entity factor* and one *entity relation factor*. Each relation tag also associates with two factors, one *relation factor* and one *entity relation factor*. Some relation tags and the associated factors are omitted in the figure for simplicity.

The model optimize the conditional joint probability:

$$p(\mathbf{y}^r, \mathbf{y}^e|\mathbf{x}) = \frac{1}{Z(\mathbf{x})}\big(\prod \phi^e(\mathbf{x}, \mathbf{y}^e)\big)\big(\prod \phi^r(\mathbf{x}, \mathbf{y}^r)\big)\big(\prod \phi^{re}(\mathbf{y}^e, \mathbf{y}^r)\big), \tag{6.1}$$

Where $Z(\mathbf{x}) = \sum_{y^e}\sum_{y^r}\big(\prod \phi^e(\mathbf{x}, \mathbf{y}^e)\big)\big(\prod \phi^r(\mathbf{x}, \mathbf{y}^r)\big)\big(\prod \phi^{re}(\mathbf{y}^e, \mathbf{y}^r)\big)$ is the partition function. $\phi^e, \phi^r, \phi^{re}$ represent the potential functions associate with the three types of factors respectively. The *entity factor* and the *relation factor* are computed by deep neural networks, and they are jointly trained end-to-end with the final objective (the conditional joint probability in equation 6.1). The *entity relation factor* is computed by maximum likelihood estimation (MLE) of the distribution in the training data [2]. They are pre-computed and do not jointly optimize with the final objective function.

Apparently, in this *DNNs-ILP* model, the connections between the variables form cyclic graphs, therefore, exact marginal inference for computing the partition function Z in equation 6.1 is intractable. To overcome the computation challenges, one choice is to employ approximate inference such as loopy belief propagation (loopyBP) to compute Z, however, the convergence is hard to control. Therefore, we propose to minimize the following structured support vector machine (SSVM) loss instead of maximizing the conditional probability in equation 6.1:

$$\min_{w} ||w||^2 + C\sum_{n=1}^{l}\max_{\mathbf{y}^*_\mathbf{n}\in\mathbb{Y}}(0, \Delta(y^*_n, y_n) + S(\mathbf{x}_n, \mathbf{y}^*_n) - S(\mathbf{x}_n, \mathbf{y}_n)), \tag{6.2}$$

[2] In other words, $P(y^{e_i}, y^{e_j}, y^{r_m}) = \frac{c(y^{e_t}, y^{e_k}, y^{r_p})}{\sum_t\sum_k\sum_p c(y^{e_t}, y^{e_k}, y^{r_p})}$, where $c(.)$ denotes the counts in the training data.

where w denotes the model parameters, n indicates the instance number, $\Delta(y^*, y)$ represents a distance measure between the gold output y and the predicted output y^* (in our case $y = y^e, y^r$), and $S(.)$ denotes a scoring function. The intuition behind this loss function requires the score of gold output structure y is greater than the score of the best output structure under the current model y^* with a margin $\Delta(y^*, y)$, or else there will be some loss. The goal of the training is to minimize the loss. One advantage of this objective function is that it only needs maximum a posteriori estimation (MAP) inference to find the best output structure y_n^*, which can be efficiently computed by integer linear programming.

6.3.2 Integer Linear Programming Inference

We propose to formulate the joint inference problem as an integer linear programming that solves:

$$\arg\max_{y^* \in \mathcal{Y}} S(\mathbf{x}, \mathbf{y}) = \sum_{i,j} \sum_{t,p} (\phi_i^{e_t} y_i^{e_t} + \phi_{ij}^{r_p} y_{ij}^{r_p} + \phi_{ij}^{re_p} y_{ij}^{r_p}), \tag{6.3}$$

where ϕ denotes the potential functions as introduced before; $y_{e_t}^i$ denotes a $0, 1$-valued variable for token x_i that indicates whether it is an entity of type e_t. Similarly, $y_{ij}^{r_p}$ represents a $0, 1$-valued variable for a pair of tokens (x_i, x_j), indicates whether there is an

relation of type r_p between them. We add constraints:

$$\sum_{e^t} y_i^{e^t} \leq 1 \text{ for } \forall i \in N, \tag{6.4}$$

$$\sum_{r_p} y_{ij}^{r_p} \leq 1 \text{ for } \forall i, j \in N, \tag{6.5}$$

$$\sum_{r_p \in P \setminus null} y_{ij}^{r_p} \leq \sum_{e^t \in T \setminus null} y_i^{e^t} + \sum_{e^t \in T \setminus null} y_j^{e^t} \text{ for } \forall i, j \in N, \tag{6.6}$$

to ensure each token can only be assigned one entity type (equation 6.4), and each pair of tokens can only be assigned one relation type (equation 6.5). We also introduce the constraint in equation 6.6 to ensure the consistency of the entity assignments and the relation assignments. Specifically, if there is a non-null relation assigned to tokens (x_i, x_j), then x_i and x_j must also be assigned non-null entity labels (they cannot be non-entities). We use this inference method during training to help compute $S(\mathbf{x}_n, \mathbf{y}_n^*)$ in equation 6.2, and also during test time to make predictions.

6.3.3 Neural Networks Factors

As we mentioned in Section 6.3.1, The *DNNs-ILP* model can be viewed as a high-order CRF model with neural network factors. The neural-nets factors compute the potential functions for the *entity factor* and the *relation factor*. For simplicity, we used BiLSTM as

the basic representation learner for both factors, and got:

$$\mathbf{h}^e = h_1^e, h_2^e, ..., h_n^e = BiLSTM^{(e)}(\mathbf{x}), \tag{6.7}$$

$$\mathbf{h}^r = h_1^r, h_2^r, ..., h_n^r = BiLSTM^{(r)}(\mathbf{x}). \tag{6.8}$$

where h^e and h^r denote the hidden representations for entity factor and relation factor respectively. We further define the potential functions:

$$\phi^e = \tanh\left(W_{(e)}^T \mathbf{h}^e + \mathbf{b}^{(e)}\right), \tag{6.9}$$

$$\phi^r = \tanh\left(W_{(r)}^T [\mathbf{h}^r; \mathbf{h}^r] + \mathbf{b}^{(r)}\right), \tag{6.10}$$

where $[;]$ denotes element-wise concatenation [3], $\tanh(.)$ denotes element-wise hyperbolic tangent transformation. $W_{(e)} \in \mathbb{R}^{h \times T}, \mathbf{b}^{(e)} \in \mathbb{R}^T, W_{(r)} \in \mathbb{R}^{2h \times P}, \mathbf{b}^{(r)} \in \mathbb{R}^P$ are model parameters.

6.3.4 Sharing Representations

Since both the entity potential and the relation potential are computed with BiLSTMs, it opens up the opportunities to learn shared representations for entity mention detection and relation extraction by sharing their BiLSTMs encoder. We explore two ways to conduct the representation sharing.

[3] Concatenate $\mathbf{h}_i^r$ with $\mathbf{h}_j^r$ for $i, j \in N$

Hard Parameter Sharing simply sharing all the parameters of $BiLSTM^{(e)}$ and $BiLSTM^{(h)}$, namely let:

$$BiLSTM^{(r)} = BiLSTM^{(e)}.$$

This strategy make $\mathbf{h}^e$ in equation 6.7 equals to $\mathbf{h}^r$ in equation 6.8, which helps producing robust feature representations for both entity mention detection and relation extraction. The differences in equation 6.9 and equation 6.10 will help produce different potential functions for entities and relations.

Soft Parameter Tying is similar to the hard parameter sharing strategy, except that we *regularize* $BiLSTM^{(e)}$ and $BiLSTM^{(h)}$ to be similar instead of forcing them to be the same. Namely we add the following objective:

$$\min \frac{1}{2}||BiLSTM^{(e)} - BiLSTM^{(h)}||_2$$

to the objective function in equation 6.2. This imposes softer constraint on the learned feature representations for entities and relations, but still encourages them to share similar feature representations.

6.3.5 Pruning

To enhance the computational efficiency, we also apply pruning to reduce the number of tokens that we need to assign labels. The pruning generates "candidates" that are

likely to be entities, thus the *DNNs-ILP* model only needs to assign an entity tag for each candidate, and a relation tag for each pair of candidates.

The pruning can be down in several ways: one can either pruning using a pre-trained high recall low precision entity mention detector, or pruning on-the-fly according to the entity scores provided by the entity potential ϕ^e, or identifying noun phrases as candidates. We used the second choice for simplicity.

6.4 Parameter Estimation

The proposed *DNNs-ILP* model can be trained end-to-end by minimizing the structured-SVM loss in equation 6.2. Model training is a straightforward application of gradient based back-propagation. We use the stochastic gradient descent (SGD) algorithm and decay the learning rate when results on development data do not improve after 10 consecutive epochs. We train for up to 60 epochs and use early stopping (Caruana et al., 2001; Graves et al., 2013) as measured on development data. We select the best model for each dataset based on hyper-parameter tuning. We use dropout on the embeddings and the BiLSTM output vectors. We use 100-dimensional pre-trained embeddings provided by GloVe (Pennington et al., 2014). All other model parameters are initialized uniformly at random in the range of $[-1, 1]$.

We tune the hyper-parameters including the initial learning rate in $\{0.005, 0.01, 0.02\}$;

the dropout rate for the input embedding and the hidden vectors in $\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$; and the LSTM hidden vector dimensions in $\{100, 150, 300\}$. We tune these hyper-parameter using beam search on development data.

6.5 Experiment

We conduct the experiments on two benchmark datasets: ACE05 and ACE04. We compare our *DNNs-ILP* joint model with several different strategies for end-to-end entity and relation extraction.

6.5.1 Datasets and Evaluation Metrics

ACE05 defines 7 coarse-grained entity types, namely Person (PER), Organization (ORG), Geographical Entities (GPE), Location (LOC), Facility (FAC), Weapon (WEA) and Vehicle (VEH). There are 6 coarse-grained relation types, namely Physical (PHYS), Person-Social (PER-SOC), Organization-Affiliation (ORG-AFF), Agent-Artifact (ART), GPE-Affiliation (GPE-AFF). We evaluate on these coarse-grained types use the same data splits and task settings as (Li and Ji, 2014).

ACE04 defines the same 7 coarse-grained entity types as ACE05 (Doddington et al., 2004), but defines 7 coarse-grained relation types incuding a Discourse (DISC) type. We

follow the cross-validation setting of (Chan and Roth, 2010) and (Li and Ji, 2014).

Evaluation reports the primary micro F1-scores as well as micro precision and recall on both entity and relation extraction. We follow (Li and Ji, 2014) to treat an entity as correct when its type and the region of its head are correct. We treat a relation as correct when its type and argument entities' head words are correct. We treat all non-negative relations on wrong entities as false positives. We removed duplicated entities and relations, and resolved nested entities follow (Miwa and Bansal, 2016).

6.5.2 Comparisons on Inference Strategies

Datasets	Methods	Entity			Relation			Entity + Relation		
		Prec	Recall	F1	Prec	Recall	F1	Prec	Recall	F1
ACE05	**Pipeline**	72.8	72.6	72.7	40.2	26.8	32.2	31.3	24.3	27.4
	Incremental Inference	78.6	73.0	75.7	48.3	34.4	40.2	45.1	31.9	37.4
	Simultaneous Inference	77.2	76.1	76.6	55.3	33.5	41.7	52.9	31.2	39.3
ACE04	**Pipeline**	70.5	69.8	70.1	26.3	24.4	25.3	26.3	19.3	22.2
	Incremental Inference	73.2	71.4	74.3	39.2	25.2	30.7	37.6	22.3	28.0
	Simultaneous Inference	75.9	74.9	75.4	40.0	27.9	32.9	37.3	25.5	30.3

Table 6.2: The main results on end-to-end entity and relation extraction. Comparing the pipeline, incremental inference and joint inference strategies.

Figure 6.2 shows the results comparing the pipeline, incremental inference and simultaneous inference strategies under our *DNNs-ILP* framework on ACE04 and ACE05 datasets. Both the incremental and the simultaneous inference models jointly trained a shared BiLSTM representation learner to produce features for both the entity and the relation modules. The pipeline model separately trained an entity mention detector and a relation extractor, and make predictions incrementally. The incremental inference is achieved by first realize the *one-best* prediction of the entity mention detection, and predict relations based on the entity mention outputs.

We can see that the behaviors on ACE04 and ACE05 are consistent: both joint inference strategies significantly improved over the pipeline strategy, as they better modeled the interactions between entities and relations.

The simultaneous inference strategy further improves over the incremental inference as the inference better utilize the global constraints in equation 6.6 to make more robust predictions and thus helps training.

6.5.3 Comparisons on Parameter Sharing

Figure 6.3 shows the results comparing the effect of parameter sharing for the representation learners on entity mention detection and relation extraction modules. For this ablation study, we only experimented on ACE05 dataset. We used the hard parameter

Methods	Entity			Relation			Entity + Relation		
	Prec	**Recall**	**F1**	**Prec**	**Recall**	**F1**	**Prec**	**Recall**	**F1**
Simultaneous Inference	77.2	76.1	76.6	55.3	33.5	41.7	52.9	31.2	39.3
- Sharing Parameters	76.9	75.9	76.4	48.7	27.1	34.8	46.3	25.2	32.6

Table 6.3: Ablation study of the proposed DNNs-ILP model on ACE05.

tying strategy.

The results shown that sharing parameters of the representation learner indeed helped joint entity and relation extraction especially on the relation extraction aspect. This is partially due to the fact that there are much more entity mentions than relation mentions in the ACE05 dataset. By transferring the knowledge from the representation learner for entity mention detection, the relation extractor enjoyed 7% improvements on F1 score.

6.5.4 Comparison with State-of-the-art Models

Figure 6.4 compares our model with the state-of-the-art results (Katiyar and Cardie, 2017) on ACE04 and ACE05 datasets. We can see that our performances lag behind the state-of-the-art, majorly because the each individual module is not as good. The reason could be that we have not yet incorporate many useful information that are widely used in previous relation extraction systems on ACE datasets. The features including part-of-speech tag embeddings, WordNet type embeddings, dependency arc type embeddings, etc.

Datasets	Methods	Entity			Relation			Entity + Relation		
		Prec	Recall	F1	Prec	Recall	F1	Prec	Recall	F1
ACE05	**Katiyar & Cardie**	84.0	81.3	82.6	60.5	55.3	57.8	57.8	52.9	55.3
	Our Model	77.2	76.1	76.6	55.3	33.5	41.7	52.9	31.2	39.3
ACE04	**Katiyar & Cardie**	81.2	78.1	79.6	50.2	48.8	49.3	46.4	45.3	45.7
	Our Model	75.9	74.9	75.4	40.0	27.9	32.9	37.3	25.5	30.3

Table 6.4: The comparison of our system with the state-of-the-art results on ACE04 and ACE05 data.

We are working on adding all those features to improve the individual modules, so that we can have better joint models.

6.6 Discussion

In this chapter, we proposed a novel graphical model with neural factors for joint entity and relation extraction. The proposed model combined the advantages from two perspectives of joint models: joint inference that makes holistic and robust decision, and representation sharing that learns robust representations in a multi-task fashion. We investigated the effect of joint inference and representation sharing for joint entity and relation extraction by conducting ablation studies under the proposed framework, and found that both joint inference and representation sharing are helpful for the joint models.

Our results were not competitive to the state-of-the-art results, as our individual models were not as good. We plan to work on improving the individual modules by adding feature embeddings in the future.

Chapter 7

Conclusion

This dissertation explored information extraction in the low-resource setting, and researched representation learning with joint models for low-resource IE. We initiated two tasks that belong to low-resource IE settings: named entity recognition in Chinese social media and cross-sentence n-ary relation extraction in Biomedical literature. We have released datasets and softwares to facilitate research in related areas.

I have explored two directions of learning robust representations to combat the problem of limited supervision. The first direction explored learning representations from heterogeneous sources and transferring the knowledge. The second direction explored learning representations with explicit structural modeling to incorporate prior domain knowledge. Both directions were proven efficient in handling low-resource IE tasks.

I conclude by summarizing the contributions of each chapter and discussing directions for future work.

7.1 Summary

Chapter 3 introduced the task of NER on Chinese social media, exploring two multi-task learning strategies to leverage supervision signals from unlabeled texts and annotations of Chinese word segmentation to help conduct NER on Chinese social media with a small annotated dataset. Experiments showed that both multi-task learning strategies yielded large improvements over classic supervised NER models. When two tasks are very related, sharing high-level representations (e.g., contextual representations produced by RNNs) is more efficient than only sharing low-level representations (e.g., word embeddings). We also found that NER on Chinese social media remains a challenging task, with results lagging behind both formal Chinese text and English social media. We released the dataset and code to facilitate research in related areas.

Chapter 4 introduced a multi-task domain adaptation framework incorporating the characteristics of both multi-task learning and domain adaptation to learn robust representations from heterogeneous sources. The framework leveraged and transferred knowledge from *both* related tasks *and* related domains to help low-resource information extraction tasks. We applied the proposed framework to two sequence tagging tasks: a) Chinese

word segmentation and b) named entity recognition for Chinese texts for two domains: a) news report and b) social media, achieving the state-of-the-art results in both tasks in the (low-resource) social media domain.

Chapter 5 introduced a new setting of cross-sentence n-ary relation extraction, proposing a graph-based long short-term memory network (graph LSTMs) architecture to learn representations for relation extraction. The graph formulation provided a unified way to explore different LSTM approaches and incorporate various intra- and inter-sentential structures, including sequential, syntactic, and discourse relations. The relation classifier relied solely on "entity representations" learned by graph LSTMs, thus was easy for scaling to arbitrary relation arity n, as well as multi-task learning with related relations. Experiments demonstrated the effectiveness of both the cross-sentence n-ary relation extraction formulation and the graph LSTMs based RE architecture. Cross-sentence extraction produced more knowledge, and multi-task learning significantly improved extraction accuracy. We also conducted thorough analysis comparing various LSTM approaches, which led to interesting insight into how linguistic analysis impacts performance. We released the dataset composed by distant supervision for the new setting of cross-sentence n-ary relation extraction.

Chapter 6 proposed a novel graphical model with neural factors for joint entity and relation extraction the combined two advantages of joint models: a). joint inference, which

makes holistic and robust decisions, and b). representation sharing that learns robust representations in a multi-task fashion. We conducted a thorough ablation study comparing the effectiveness of joint inference and representation sharing in low-resource IE.

7.2 Future Work

My research has demonstrated that large-scale information extraction with minimal resources and annotations can be the future. With limited supervision, learning representations with structured joint models present a principled way to combat the problem of inadequate supervision signals.

The next phase of my research agenda is to pursue both new models and other information extraction applications with scarce annotations.

7.2.1 Combining Structured Modeling with Neural Representation Learning

Structured models encode prior domain knowledge and thus present a good way to involve domain experts, while representation learning yield abstract representations from data that are generalizable to many tasks. Combining these two strikes a balance between incorporating domain knowledge and data-driven learning, both of which are important for

information extraction. However, it is challenging to effectively incorporate structures that are helpful and tractable for inference and learning. I have explored combining graphical models with neural networks and have designed novel neural architectures for IE tasks in low-resource conditions. In the future, I will closely collaborate with domain experts to incorporate important structures into neural models and conduct efficient joint learning and inference. My existing body of research on joint inference for graphical models of underlying morphs and phonology (Cotterell et al., 2014, 2015; Peng et al., 2015a), and topic modeling (Peng et al., 2014) provide a solid background for future exploration.

7.2.2 Interpretation of Continuous Representations

With recent advances in deep representation learning, most NLP applications enjoy significant performance gains from deep neural network training and continuous representations. However, the neural architectures for NLP are usually quite complicated; extremely deep neural networks with non-linear transformations on each layer present challenges in the interpretation of the models. However, many specialized domains, including biomedicine, economics, and law, need explanation of the models' behavior for humans to draw insights from the model or confirm the results. I want to develop new learning methods that can explain the behavior of deep neural network models and can interpret the learned low-dimensional representations.

7.2.3 Multi-lingual Representation Learning

I have studied learning representations across tasks and domains to improve the performance of information extraction tasks with inadequate annotations; learning representations across languages, then, would be a natural and significant direction to explore. I have investigated learning multi-lingual representations using topic models (Peng et al., 2014), and I plan to build on this work in the future.

7.2.4 Applications of Information Extraction to New Domains

Information Extraction is widely applicable to many domains, including social media messages, biomedical publications, emails, and business reports. Most of these domains have little annotated data for training. I have explored news, social media, and biomedical domains, and I plan to explore new domains in collaboration with domain experts in order to build better IE systems.

Bibliography

Aguilar, J., Beller, C., McNamee, P., Van Durme, B., Strassel, S., Song, Z., and Ellis, J. (2014). A comparison of the events and relations across ACE, ERE, TAC-KBP, and FrameNet annotation standards. In *ACL Workshop: EVENTS*.

Ananiadou, S., Pyysalo, S., Tsujii, J., and Kell, D. B. (2010). Event extraction for systems biology by text mining the literature. *Trends in biotechnology*, 28(7):381–390.

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.

Augenstein, I. and Søgaard, A. (2017). Multi-task learning of keyphrase boundary classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vancouver, Canada. Association for Computational Linguistics.

BIBLIOGRAPHY

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Bell, D., Hahn-Powell, G., Valenzuela-Escarcega, M. A., and Surdeanu, M. (2016). An investigation of coreference phenomena in the biomedical domain. In *Proceedings of the Tenth Edition of the Language Resources and Evaluation Conference*.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*.

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L. (2006). Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

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

Benton, A., Mitchell, M., and Hovy, D. (2017). Multi-task learning for mental health using

social media text. In *Proceedings of the 15th conference on European chapter of the Association for Computational Linguistics (EACL)*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Blitzer, J., Dredze, M., Pereira, F., et al. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bodenreider, O. (2004). The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270.

Bodnari, A., Szolovits, P., and Uzuner, Ö. (2012). Mcores: a system for noun phrase coreference resolution for clinical records. *Journal of the American Medical Informatics Association*, 19(5):906–912.

Bolukbasi, T., Chang, K.-W., Zou, J., Saligrama, V., and Kalai, A. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Proceedings of NIPS*.

BIBLIOGRAPHY

Boschee, E., Weischedel, R., and Zamanian, A. (2005). Automatic information extraction. In *Proceedings of the International Conference on Intelligence Analysis*, volume 71. Citeseer.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Bui, Q.-C., Katrenko, S., and Sloot, P. M. (2011). A hybrid approach to extract protein–protein interactions. *Bioinformatics*, 27(2):259–265.

Bundschus, M., Dejori, M., Stetter, M., Tresp, V., and Kriegel, H.-P. (2008). Extraction of semantic biomedical relations from text using conditional random fields. *BMC bioinformatics*, 9(1):207.

Bunescu, R. C. and Mooney, R. J. (2005). A shortest path dependency kernel for relation extraction. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 724–731. Association for Computational Linguistics.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

BIBLIOGRAPHY

Cai, R., Zhang, X., and Wang, H. (2016). Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Callison-Burch, C. and Dredze, M. (2010). Creating speech and language data with Amazon's Mechanical Turk. In *NAACL Workshop on Creating Speech and Language Data With Mechanical Turk*.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

Caruana, R., Lawrence, v., and Giles, L. (2001). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Proceedings of NIPS*.

Caruana, R. A. (1993). Multitask connectionist learning. In *In Proceedings of the 1993 Connectionist Models Summer School*. Citeseer.

Cdesouza de Souza, J. G., Zamani, H., Negri, M., Turchi, M., and Daniele, F. (2015). Multitask learning for adaptive quality estimation of automatically transcribed utterances. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado. Association for Computational Linguistics.

Chan, Y. S. and Roth, D. (2010). Exploiting background knowledge for relation extrac-

tion. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 152–160. Association for Computational Linguistics.

Chen, M., Xu, Z., Sha, F., and Weinberger, K. Q. (2012). Marginalized denoising autoencoders for domain adaptation. In *Proceedings of ICML*.

Chen, X., Qiu, X., Zhu, C., Liu, P., and Huang, X. (2015a). Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chen, X., Xu, L., Liu, Z., Sun, M., and Luan, H. (2015b). Joint learning of character and word embeddings. In *International Joint Conference on Artificial Intelligence (IJCAI'15)*.

Cheng, H., Fang, H., and Ostendorf, M. (2015). Open-domain name error detection using a multi-task rnn. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal. Association for Computational Linguistics.

Cherry, C. and Guo, H. (2015). The unreasonable effectiveness of word representations for twitter named entity recognition. In *Proceedings of the North America Chapter of Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics.

BIBLIOGRAPHY

Chinchor, N. A. (1998). Overview of MUC-7/MET-2. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.

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. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Collins, M. and Singer, Y. (1999). Unsupervised models for named entity classification. In *Proceedings of the 1999 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 100–110. Citeseer.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning (ICML)*, pages 160–167. ACM.

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

BIBLIOGRAPHY

Cotterell, R., Peng, N., and Eisner, J. (2014). Stochastic contextual edit distance and probabilistic fsts. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 625–630.

Cotterell, R., Peng, N., and Eisner, J. (2015). Modeling word forms using latent underlying morphs and phonology. *Transactions of the Association for Computational Linguistics*, 3:433–447.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. (2000). Learning to construct knowledge bases from the world wide web. *Artificial intelligence*, 118(1-2):69–113.

Craven, M. and Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86.

Craven, M., McCallum, A., PiPasquo, D., Mitchell, T., and Freitag, D. (1998). Learning to extract symbolic knowledge from the world wide web. Technical report, DTIC Document.

Cross, J. and Huang, L. (2016). Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

BIBLIOGRAPHY

Cummins, R., Zhang, M., and Briscoe, T. (2016). Constrained multi-task learning for automated essay scoring. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany. Association for Computational Linguistics.

Daumé III, H. (2009a). Bayesian multitask learning with latent hierarchies. In *Proceedings of UAI*.

Daumé III, H. (2009b). Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

de Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454, Genoa, Italy. ELRA.

Dienstmann, R., Jang, I. S., Bot, B., Friend, S., and Guinney, J. (2015). Database of genomic biomarkers for cancer drugs and clinical targetability in solid tumors. *Cancer Discovery*, 5:118–123.

Doddington, G. R., Mitchell, A., Przybocki, M. A., Ramshaw, L. A., Strassel, S., and Weischedel, R. M. (2004). The automatic content extraction (ACE) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

Dredze, M. and Crammer, K. (2008). Online methods for multi-domain learning and adaptation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dredze, M., Kulesza, A., and Crammer, K. (2010a). Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*.

Dredze, M., McNamee, P., Rao, D., Gerber, A., and Finin, T. (2010b). Entity disambiguation for knowledge base population. In *Conference on Computational Linguistics (Coling)*.

Duan, H., Sui, Z., Tian, Y., and Li, W. (2012). The cips-sighan clp 2012 chineseword segmentation onmicroblog corpora bakeoff. In *Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 35–40, Tianjin, China. Association for Computational Linguistics.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*.

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

BIBLIOGRAPHY

Emerson, T. (2005). The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*.

Etter, D., Ferraro, F., Cotterell, R., Buzek, O., and Van Durme, B. (2013). Nerit: Named entity recognition for informal text. Technical report, Technical Report 11, Human Language Technology Center of Excellence, Johns Hopkins University, July.

Everett III, H. (1963). Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations research*, 11(3):399–417.

Fang, X., Gao, J., and Sheng, H. (2004). A semi-supervised approach to build annotated corpus for chinese named entity recognition. In Streiter, O. and Lu, Q., editors, *ACL SIGHAN Workshop 2004*, pages 129–133, Barcelona, Spain. Association for Computational Linguistics.

Finin, T., Lawrie, D., McNamee, P., Mayfield, J., Oard, D., Peng, N., Gao, N., Lin, Y.-C., and Dowd, T. (2015). HLTCOE participation in TAC KBP 2015: Cold start and TEDL.

Finin, T., Murnane, W., Karandikar, A., Keller, N., Martineau, J., and Dredze, M. (2010). Annotating named entities in twitter data with crowdsourcing. In *NAACL Workshop on Creating Speech and Language Data With Mechanical Turk*.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd*

Annual Meeting of the Association for Computational Linguistics (ACL), pages 363–370. Association for Computational Linguistics.

Finkel, J. R. and Manning, C. D. (2009). Hierarchical bayesian domain adaptation. In *Proceedings of NAACL*.

FitzGerald, N., Täckström, O., Ganchev, K., and Das, D. (2015). Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 960–970, Lisbon, Portugal. Association for Computational Linguistics.

Fromreide, H., Hovy, D., and Søgaard, A. (2014). Crowdsourcing and annotating NER for Twitter# drift. In *LREC*.

Gelfand, A. E. and Smith, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409.

Gerber, M. and Chai, J. Y. (2010). Beyond nombank: A study of implicit arguments for nominal predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1583–1592, Stroudsburg, PA, USA. Association for Computational Linguistics.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of ICML*.

Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of CVPR*.

Gormley, M. R., Yu, M., and Dredze, M. (2015). Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*.

Grishman, R. and Sundheim, B. (1996). Design of the muc-6 evaluation. In *Proceedings of a workshop on held at Vienna, Virginia: May 6-8, 1996*, pages 413–422. Association for Computational Linguistics.

Hamilton, W. L., Leskovec, J., and Jurafsky, D. (2016). Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of ACL*.

He, H. and Sun, X. (2017a). F-score driven max margin neural network for named entity recognition in chinese social media. In *Proceedings of EACL*.

He, H. and Sun, X. (2017b). A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *Proceedings of AAAI*.

He, Z., Wang, H., and Li, S. (2012a). The task 2 of cips-sighan 2012 named entity recogni-

tion and disambiguation in chinese bakeoff. In *Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 108–114, Tianjin, China. Association for Computational Linguistics.

He, Z., Wang, H., and Li, S. (2012b). The task 2 of cips-sighan 2012 named entity recognition and disambiguation in chinese bakeoff. In *Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 108–114, Tianjin, China. Association for Computational Linguistics.

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

Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., and Weld, D. S. (2011). Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

Ji, H. and Grishman, R. (2005). Improving name tagging by reference resolution and relation detection. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 411–418. Association for Computational Linguistics.

Jin, G. and Chen, X. (2008). The fourth international chinese language processing bakeoff: Chinese word segmentation, named entity recognition and chinese pos tagging. In *Sixth SIGHAN Workshop on Chinese Language Processing*, page 69. Citeseer.

Joshi, M., Cohen, W. W., Dredze, M., and Rosé, C. P. (2012). Multi-domain learning: when do domains matter? In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kambhatla, N. (2004). Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

Kate, R. J. and Mooney, R. J. (2010). Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212. Association for Computational Linguistics.

BIBLIOGRAPHY

Katiyar, A. and Cardie, C. (2017). Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., and Tsujii, J. (2009). Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9. Association for Computational Linguistics.

Kiperwasser, E. and Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*.

Koo, T., Carreras Pérez, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics*, pages 595–603.

Kumar, A., Saha, A., and Daume, H. (2010). Co-regularization based semi-supervised domain adaptation. In *Proceedings of NIPS*.

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

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of NAACL*.

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., and Jurafsky, D. (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Lenstra Jr, H. W. (1983). Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548.

Levow, G.-A. (2006). The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*.

Li, C., Weng, J., He, Q., Yao, Y., Datta, A., Sun, A., and Lee, B.-S. (2012). Twiner: Named entity recognition in targeted twitter stream. In *SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 721–730, New York, NY, USA. ACM.

Li, F., Zhang, M., Fu, G., and Ji, D. (2017). A neural joint model for entity and relation extraction from biomedical text. *BMC bioinformatics*, 18(1):198.

Li, H., Krause, S., Xu, F., Moro, A., Uszkoreit, H., and Navigli, R. (2015). Improvement of

n-ary relation extraction by adding lexical semantics to distant-supervision rule learning. In *Proceedings of the International Conference on Agents and Artificial Intelligence*.

Li, Q. and Ji, H. (2014). Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 402–412.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2016). Gated graph sequence neural networks. In *Proceedings of the 4th International Conference on Learning Representations*.

Ling, W., Xiang, G., Dyer, C., Black, A., and Trancoso, I. (2013). Microblogs as parallel corpora. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Linguistics Data Consortium (2014). DEFT ERE Annotation Guidelines: Entities.

Liu, P., Qiu, X., and Huang, X. (2016a). Deep multi-task learning with shared memory. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Liu, P., Qiu, X., and Huang, X. (2016b). Recurrent neural network for text classification with multi-task learning. In *Proceedings of IJCAI*.

Liu, P., Qiu, X., and Huang, X. (2017). Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Liu, X., Duh, K., Matsumoto, Y., and Iwakura, T. (2014). Learning character representations for chinese word segmentation. In *NIPS 2014 Workshop on Modern Machine Learning and Natural Language Processing*.

Liu, X., Zhang, S., Wei, F., and Zhou, M. (2011). Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 359–367. Association for Computational Linguistics.

Liu, X., Zhou, M., Wei, F., Fu, Z., and Zhou, X. (2012a). Joint inference of named entity recognition and normalization for tweets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL '12, pages 526–535, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liu, X., Zhou, M., Wei, F., Fu, Z., and Zhou, X. (2012b). Joint inference of named entity recognition and normalization for tweets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 526–535, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liu, Y., Li, S., Zhang, X., and Sui, Z. (2016c). Implicit discourse relation classification via multi-task neural networks. In *Proceedings of AAAI*.

Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL*.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009). Domain adaptation: Learning bounds and algorithms. In *Proceedings of COLT*.

Mao, X., Dong, Y., He, S., Bao, S., and Wang, H. (2008). Chinese word segmentation and named entity recognition based on conditional random fields. In *IJCNLP*, pages 90–93.

McCallum, A. and Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the North America Chapter of Association for Computational Linguistics (NAACL)*.

McDonald, R., Pereira, F., Kulick, S., Winters, S., Jin, Y., and White, P. (2005). Simple algorithms for complex relation extraction with applications to biomedical ie. In

Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pages 491–498.

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Miller, S., Guinness, J., and Zamanian, A. (2004). Name tagging with word clusters and discriminative training. In *HLT-NAACL*, volume 4, pages 337–342.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Miwa, M. and Bansal, M. (2016). End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Miwa, M. and Sasaki, Y. (2014). Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869.

Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Mooney, R. J. and Bunescu, R. C. (2005). Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Morgan, A. A., Hirschman, L., Colosimo, M., Yeh, A. S., and Colombe, J. B. (2004). Gene name identification and normalization using a model organism database. *Journal of biomedical informatics*, 37(6):396–410.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Nguyen, T. H. and Grishman, R. (2014). Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nguyen, T. H. and Grishman, R. (2015a). Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*.

Nguyen, T. H. and Grishman, R. (2015b). Relation extraction: Perspective from convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 39–48.

Nguyen, T. H., Plank, B., and Grishman, R. (2015). Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *ACL (1)*, pages 635–644.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106.

Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of ICML*.

Passos, A., Kumar, V., and McCallum, A. (2014). Lexicon infused phrase embeddings for named entity resolution. *CoRR*, abs/1404.5367.

Pasunuru, R. and Bansal, M. (2017). Multi-task video captioning with video and entailment generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

BIBLIOGRAPHY

Pei, W., Ge, T., and Baobao, C. (2014). Maxmargin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Peng, H., Thomson, S., and Smith, N. A. (2017). Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Peng, N., Cotterell, R., and Eisner, J. (2015a). Dual decomposition inference for graphical models over strings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Peng, N. and Dredze, M. (2015). Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisboa, Portugal.

Peng, N. and Dredze, M. (2016). Improving named entity recognition for chinese social media via learning segmentation representations.

Peng, N. and Dredze, M. (2017). Multi-task domain adaptation for sequence tagging. In *Proceddings of the ACL Workshop on Representation Learning for NLP*.

Peng, N., Ferraro, F., Yu, M., Andrews, N., DeYoung, J., Thomas, M., Gormley, M. R., Wolfe, T., Harman, C., Van Durme, B., et al. (2015b). A concrete chinese NLP pipeline. In *Proceedings of NAACL-HLT*, pages 86–90.

Peng, N., Wang, Y., and Dredze, M. (2014). Learning polylingual topic models from code-switched social media documents. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics ACL.*

Peng, N., Yu, M., and Dredze, M. (2015c). An empirical study of chinese name matching and applications. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 377–383.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pham, N. T., Lazaridou, A., and Baroni, M. (2015). A multitask objective to inject lexical contrast into distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China. Association for Computational Linguistics.

Plank, B. and Moschitti, A. (2013). Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL (1)*, pages 1498–1507.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Poon, H., Quirk, C., DeZiel, C., and Heckerman, D. (2014). Literome: Pubmed-scale genomic knowledge base in the cloud. *Bioinformatics*.

Poon, H., Toutanova, K., and Quirk, C. (2015). Distant supervision for cancer pathway extraction from text. In *Pac. Symp. Biocomput*, pages 120–131.

Qian, L., Zhou, G., Kong, F., Zhu, Q., and Qian, P. (2008). Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 697–704. Association for Computational Linguistics.

Quirk, C., Choudhury, P., Gao, J., Suzuki, H., Toutanova, K., Gamon, M., Yih, W., and Vanderwende, L. (2012). MSR SPLAT, a language analysis toolkit. In *Proceedings of NAACL HLT Demonstration Session*.

Quirk, C. and Poon, H. (2016). Distant supervision for relation extraction beyond the sentence boundary. *arXiv preprint arXiv:1609.04873*.

Quirk, C. and Poon, H. (2017). Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th conference on European chapter of the Association for Computational Linguistics (EACL)*. Association for Computational Linguistics.

Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1375–1384. Association for Computational Linguistics.

Ravikumar, K., Wagholikar, K. B., and Liu, H. (2014). Towards pathway curation through literature mining–a case study using pharmgkb. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, page 352. NIH Public Access.

Rei, M. (2017). Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Reschke, K., Jankowiak, M., Surdeanu, M., Manning, C. D., and Jurafsky, D. (2014). Event extraction using distant supervision. In *Proceedings of LREC*.

Riedel, S., Yao, L., McCallum, A., and Marlin, B. M. (2013). Relation extraction with

matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*, pages 74–84.

Ritter, A., Clark, S., Etzioni, O., et al. (2011). Named entity recognition in tweets: an experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1524–1534. Association for Computational Linguistics.

Roth, D. and Yih, W.-t. (2007). Global inference for entity and relation identification via a linear programming formulation. *Introduction to statistical relational learning*, pages 553–580.

Roth, M. and Lapata, M. (2016). Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Roth, M. and Woodsend, K. (2014). Composition of word representations improves semantic role labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 407–413.

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Santos, C. N. d., Xiang, B., and Zhou, B. (2015). Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Setiawan, H., Huang, Z., Devlin, J., Lamar, T., Zbib, R., Schwartz, R., and Makhoul, J. (2015). Statistical machine translation features with multitask tensor networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China. Association for Computational Linguistics.

Shah, K. and Specia, L. (2016). Large-scale multitask learning for machine translation quality estimation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California. Association for Computational Linguistics.

Singh, S., Riedel, S., Martin, B., Zheng, J., and McCallum, A. (2013). Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 1–6. ACM.

Snow, R., Jurafsky, D., and Ng, A. Y. (2005). Learning syntactic patterns for automatic hy-

pernym discovery. In *Advances in neural information processing systems*, pages 1297–1304.

Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Søgaard, A. and Goldberg, Y. (2016a). Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Berlin, Germany. Association for Computational Linguistics.

Søgaard, A. and Goldberg, Y. (2016b). Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 231–235. Association for Computational Linguistics.

Soon, W. M., Ng, H. T., and Lim, D. C. Y. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

Suchanek, F. M., Ifrim, G., and Weikum, G. (2006). Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 712–717. ACM.

Sun, A., Grishman, R., and Sekine, S. (2011). Semi-supervised relation extraction with large-scale word clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 521–529. Association for Computational Linguistics.

Sun, Y., Lin, L., Yang, N., Ji, Z., and Wang, X. (2014). Radical-enhanced chinese character embedding. In *Neural Information Processing Systems (NIPS)*, pages 279–286. Springer.

Surdeanu, M. and Heng, J. (2014). Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proceedings of the TAC-KBP 2014 Workshop*.

Surdeanu, M., Tibshirani, J., Nallapati, R., and Manning, C. D. (2012). Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Swampillai, K. and Stevenson, M. (2011). Extracting relations within and across sentences. In *Proceedings of RANLP*.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Thompson, B. (2005). Canonical correlation analysis. *Encyclopedia of statistics in behavioral science*.

Tjong Kim Sang, E. F. (2002). Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA. 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., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. (2015). Representing text for joint embedding of text and knowledge bases. In *EMNLP*, volume 15, pages 1499–1509.

Toutanova, K., Lin, X. V., Yih, W.-t., Poon, H., and Quirk, C. (2016). Compositional learning of embeddings for relation paths in knowledge bases and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1434–1444.

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 (ACL)*, pages 384–394. Association for Computational Linguistics.

Turian, J., Ratinov, L., Bengio, Y., and Roth, D. (2009). A preliminary evaluation of word representations for named-entity recognition. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*, pages 1–8.

Vijayaraghavan, P., Vosoughi, S., and Roy, D. (2017). Twitter demographic classification using deep multi-modal multi-task learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Walker, C., Strassel, S., Medero, J., and Maeda, K. (2006). Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Wang, L., Cao, Z., de Melo, G., and Liu, Z. (2016). Relation classification via multi-

level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Waterhouse, S. R., MacKay, D., and Robinson, A. J. (1996). Bayesian methods for mixtures of experts. In *Advances in neural information processing systems*, pages 351–357.

Wick, M., Culotta, A., and McCallum, A. (2006). Learning field compatibilities to extract database records from unstructured text. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 603–611. Association for Computational Linguistics.

Wu, F. and Weld, D. S. (2007). Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50. ACM.

Xu, K., Feng, Y., Huang, S., and Zhao, D. (2015a). Semantic relation classification via convolutional neural networks with simple negative sampling. *arXiv preprint arXiv:1506.07650*.

Xu, Y., Jia, R., Mou, L., Li, G., Chen, Y., Lu, Y., and Jin, Z. (2016). Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651*.

Xu, Y., Mou, L., Li, G., Chen, Y., Peng, H., and Jin, Z. (2015b). Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xue, N., Ng, H. T., Pradhan, S., Prasad, R., Bryant, C., and Rutherford, A. (2015). The CoNLL-2015 Shared Task on Shallow Discourse Parsing. In *Proceedings of CoNLL, Shared Task*.

Yang, Y. and Eisenstein, J. (2015). Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the North America Chapter of Association for Computational Linguistics (NAACL)*.

Yang, Y. and Hospedales, T. M. (2014). A unified perspective on multi-domain and multi-task learning. *arXiv preprint arXiv:1412.7489*.

Yang, Z., Salakhutdinov, R., and Cohen, W. (2016). Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Yao, K., Peng, B., Zweig, G., Yu, D., Li, X., and Gao, F. (2014). Recurrent conditional random field for language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4077–4081. IEEE.

Yedidia, J., Freeman, W., and Weiss, Y. (2003). Understanding belief propagation and

its generalizations. In *Exploring Artificial Intelligence in the New Millennium*, pages 239–236. Morgan Kaufmann Publishers.

Yoshikawa, K., Riedel, S., Hirao, T., Asahara, M., and Matsumoto, Y. (2011). Coreference based event-argument relation extraction on biomedical text. *Journal of Biomedical Semantics*, 2(5):1.

Yu, M. and Dredze, M. (2014). Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 545–550.

Yu, M. and Dredze, M. (2015). Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.

Yu, M., Gormley, M., and Dredze, M. (2014). Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*.

Yu, M., Gormley, M. R., and Dredze, M. (2015). Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Yu, X. and Lam, W. (2010). Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 23rd International*

Conference on Computational Linguistics: Posters, pages 1399–1407. Association for Computational Linguistics.

Zeng, D., Liu, K., Chen, Y., and Zhao, J. (2015). Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*, pages 17–21.

Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J., et al. (2014). Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Zhang, L., Li, L., He, Z., Wang, H., and Sun, N. (2013a). Improving chinese word segmentation on micro-blog using rich punctuations. In *Proceedings of ACL*.

Zhang, L., Wang, H., Sun, X., and Mansur, M. (2013b). Exploring representations from unlabeled data with co-training for chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhang, S., Qin, Y., Wen, J., and Wang, X. (2006). Word segmentation and named entity recognition for sighan bakeoff3. In *Fifth SIGHAN Workshop on Chinese Language Processing*, pages 158–161, Sydney, Australia. Association for Computational Linguistics.

Zhang, S., Zheng, D., Hu, X., and Yang, M. (2015). Bidirectional long short-term memory

networks for relation classification. In *Proceedings of 29th Pacific Asia Conference on Language, Information and Computation*.

Zheng, S., Wang, F., Bao, H., Hao, Y., Zhou, P., and Xu, B. (2017). Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zheng, X., Chen, H., and Xu, T. (2013a). Deep learning for chinese word segmentation and pos tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 647–657.

Zheng, X., Chen, H., and Xu, T. (2013b). Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA. Association for Computational Linguistics.

Zhou, G., Su, J., Zhang, J., and Zhang, M. (2005). Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics.

Zhu, X., Li, M., Gao, J., and Huang, C.-N. (2003). Single character chinese named entity recognition. In *Second SIGHAN Workshop on Chinese Language Processing*, pages 125–132, Sapporo, Japan. Association for Computational Linguistics.

Vita

Nanyun Peng graduated from Peking University, Beijing, China with double Bachelor's degrees in Computational Linguistics and Economics, where she continued to obtain a master's degree in Computer Science. As a Ph.D. student in the Department of Computer Science at Johns Hopkins University, Nanyun affiliated with the Center for Language and Speech Processing and advised by Prof. Mark Dredze. She is broadly interested in Natural Language Processing, Machine Learning, and Information Extraction. Her research focuses on using deep learning for information extraction with scarce human annotations. Nanyun is the recipient of the Johns Hopkins University 2016 Fred Jelinek Fellowship. She has interned at IBM T.J. Watson Research Center, and Microsoft Research Redmond during her Ph.D. study. Beginning in September 2017, Nanyun started her career as a computer scientist at the Information Science Institute at University of Southern California.