Leveraging Dependency Trees for Structured Prediction

Submitted by

Zhanming JIE

Thesis Advisor

Prof. Wei LU

Information Systems Technology and Design

A thesis submitted to the Singapore University of Technology and Design in fulfillment of the requirement for the degree of Doctor of Philosophy

2020

PhD Thesis Examination Committee

TEC Chair:	Prof. Ngai-Man Cheung
Main Advisor:	Prof. Wei Lu
Internal TEC member 1:	Prof. Shaowei Lin
Internal TEC member 2:	Prof. Dario Poletti
External TEC member 1:	Prof. Jing Jiang (Singapore Management University)

"When you embrace stress, you can transform fear into courage, isolation into connection, and suffering into meaning."

Kelly McGonigal

Abstract

Information Systems Technology and Design

Doctor of Philosophy

Leveraging Dependency Trees for Structured Prediction

by Zhanming JIE

Syntactic dependency structures leverage shallow semantic information, which could be helpful for downstream tasks in natural language processing (NLP). In this thesis, we present some of our recent works on improving named entity recognition (NER) and semantic parsing by making use of the rich structured information in the dependency trees. We statistically show the strong correlation between the dependency trees and named entities. Specifically, named entities often form subtrees under the dependency tree structures, and the dependency relations (e.g. *nsubj*) are strong indicators for the existence of entities. Motivated by the above observations, we proposed the dependency-guided structured models based on the conditional random fields (CRF) to capture the underlying correlations for the NER task. Our large-scale experiments on four languages demonstrate the effectiveness of the proposed model, especially for the Catalan and Spanish languages. Further analysis reveals that the improvements mainly result from the dependency relations and long-distance interactions provided by dependency trees.

Our following research work attacks a more challenging and semantic-level task, semantic parsing without the given dependency trees. The semantic parsing task is to map the natural language sentences into (tree-structured) logical forms. We introduce a dependency-based hybrid tree representation to capture the interactions between the semantic units and the natural language words. However, as the dependency trees are not observed, we regard the dependency trees as the latent structures in our proposed latent-variable model. Our dynamic-programming inference procedure is similar to the Eisner's algorithm for dependency parsing. The proposed model achieves state-of-the-art performance on 7 out of 8 languages, especially for some datasets with flexible word order.

Keyword: Dependency Trees, Structured Prediction, Named Entity Recognition, Semantic Parsing, Dynamic Programming

Publications

- **Zhanming Jie** and Wei Lu. 2019. Dependency-Guided LSTM-CRF for Named Entity Recognition. In *Proceedings of EMNLP.*
- **Zhanming Jie**, Pengjun Xie, Wei Lu, Ruixue Ding and Linlin Li. 2019. Better Modeling of Incomplete Annotations for Named Entity Recognition. In *Proceedings of NAACL.*
- **Zhanming Jie** and Wei Lu. 2018. Dependency-based Hybrid Tree for Semantic Parsing. In *Proceedings of EMNLP.*
- Chu Guo, **Zhanming Jie**, Wei Lu and Dario Poletti. 2018. Matrix Product Operators for Sequence to Sequence Learning. In *Physical Review E.*
- **Zhanming Jie**, Aldrian Obaja Muis, and Wei Lu. 2017. Efficient Dependency-Guided Named Entity Recognition. In *Proceedings of the AAAI.*
- **Zhanming Jie**, Ming Cheung, and James She. A Cloud-Assisted Framework for Bag-of-Features Tagging in Social Networks. 2015. In *IEEE Fourth Symposium on Network Cloud Computing and Applications.*
- Alvin Junus, Ming Cheung, James She, and **Zhanming Jie**. 2015. Community-Aware Prediction of Virality Timing Using Big Data of Social Cascades. In *Proceedings of IEEE BigDataService.*
- Ming Cheung, James She, and **Zhanming Jie**. 2015. Connection Discovery Using Big Data of User-Shared Images in Social Media. In *IEEE Transactions on Multimedia.*
- **Zhanming Jie** and Wei Lu. 2014. Multilingual Semantic Parsing: Parsing Multiple Languages into Semantic Representations. In *Proceedings of COLING.*

Acknowledgements

First and foremost, I'm very grateful to have Prof. Wei Lu as my PhD advisor. Without a doubt, I have a lot of for him. What I have learned from him during these years will benefit me for the rest of my life. He is the person who is sincerely serious about research, which really impresses me. I first met Prof. Wei Lu in 2014 when I did my internship with him and I knew nothing about NLP. He is extremely efficient and focused during working. He taught me what correct attitude is to research, how to do good research in terms of reading, writing, my presentation skill, and more importantly, how to think independently as a researcher. All of the things that he endowed me and I have learned from him can not be simply stated in a few sentences. But I really appreciate that I can spend these four years with him.

I would like to thank my mother and brother for their support during my PhD study. They do not give me pressure and support most of the decisions I made, though my mom is sometimes a bit immature and she is the "funniest" one in the family. When I get into trouble like mental problems, I often seek help from my brother. Talking to him makes me feel better and he is the one usually talks a lot though what he said is not really helpful. I really enjoy the happiness and joy in this family.

Throughout my PhD study, there are a lot of friends who have provided their help to me. Aldrian, who was previously a research assistant in our group, gave me tremendous guidance in my first year. He is a really smart and also very friendly teacher to me. Raymond and Li Hao had a lot of research discussions with me during my PhD study. Besides research, we chat with each other on our PhD life. I really appreciate the conversations with them as it sometimes helps me get away from the pressure of "tough" research. Gary and I joined the PhD program together back to Jan 2016. Yuqi and Gary are my best friends since I join SUTD. We took classes together, played together and talk about everything. I really appreciate the time that we spent in our PhD study and wish them a promising fortune in the future. Of course, there are so many friends and team members that I haven't mentioned but I'm really grateful for their appearance in my life. PhD life sometimes is tough. Because of them, it makes my life more fun and more wonderful.

Finally, I would like to thank the one inside my heart (*i.e.*, myself) for insisting on the PhD study. Four years ago, you had already gone through 1-year PhD life and I know that was a really tough and stressed time for you. Thus, you quit that PhD and Prof. Wei Lu gave you a second chance to demonstrate yourself. But I know you were still very stressed and somehow you might have lost your faith in doing the PhD. You were even thinking that you do not have any talents in doing research. Though you published some papers, you also failed so many research topics. After several failures, you doubted yourself, questioned yourself and denied yourself that you can do a good job. Later on, you realized that it is not about just research and publishing papers. You realized that you love doing research, especially good research. No matter if you can publish in the top conference or not, you still love doing research. As your advisor requires, I hope you are already an independent researcher now. I hope you can keep doing great research in the future and do not forget your original motivation for doing a PhD.

Contents

List of Figures

List of Tables

List of Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
NER	**N**amed **E**entity **R**ecognition
HMM	**H**idden **M**arkov **M**odel
MEMM	**M**aximum **E**ntropy **M**arkov **M**odel
CRF	**C**onditional **R**andom **F**ield
LSTM	**L**ong **S**ort-**T**erm **M**emory
DGM	**D**ependency-**G**uided **M**odel
SQL	**S**tructured **Q**uery **L**anguage
AMR	**A**bstract **M**eaning **R**epresentation
DGLSTM	**D**ependency **G**uided LSTM
NL	**N**atural **L**anguage
MR	**M**eaning **R**epresentation
MST	**M**aximum **S**panning **T**ree
GCN	**G**raph **C**onvolutional **N**etwork

To my mom and brother, for their endless support.

Chapter 1

Introduction

1.1 Dependency Trees

Dependency trees describe the bilexical relations between pairs of words in a sentence. Such relations are drawn from a fixed inventory of grammatical relations (Jurafsky, 2000). For example, Figure 1.1 shows an example sentence (taken from OntoNotes 5.0 (Weischedel et al., 2013)) with dependency annotations to describe the bilexical relationship between pairs of words.

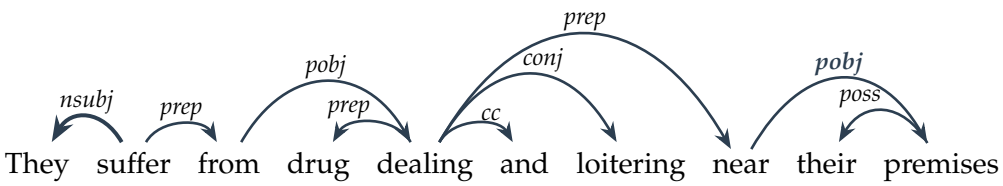


FIGURE 1.1: Example sentences annotated with dependency structures.

The word "*They*" is the subject of the verb "*suffer*", and "*from*" is the preposition of this verb. Overall, we can see that each word has exactly one parent as in the dependency trees except for the root (i.e., "*suffer*" in this case). On the other hand, this tree is ***projective*** in the sense that there is no crossing edge in the dependency trees. According to Jurafsky (2000), an arc from a head to a dependent is said to be projective if there is a path from the head to every word that lies between the head and the dependent in the sentence. A dependency tree is then said to be projective if all the arcs that make it up are projective. Most of the dependency trees in English are projective while non-projective dependencies are also allowed, particularly for some languages with flexible word orders, such as Czech.

As different languages have different forms of dependencies as well as different types of dependency relations, *Universal Dependencies*[1] (Nivre et al., 2016) are proposed to consistently unify the grammar across different languages. The speedy progress of universal dependencies allows us to work on the downstream tasks by making use of such dependency trees.

[1]https://universaldependencies.org/

1.2 Motivation

Motivated by the fact that the dependency trees convey semantic-level information, we focus on the structured prediction tasks of named entity recognition (Tjong Kim Sang and De Meulder, 2003) and semantic parsing which requires a semantic-level understanding of natural languages. Specifically, we study the underlying connections between the dependency trees and the output structures (*e.g.*, named entity sequences and structured meaning representation) based on our observation from existing datasets. We raise several research questions in this thesis for us to exploit the connections for the downstream tasks.

1.2.1 Dependency-Guided Named Entity Recognition

Though the dependencies describe the bilexical relations between pairs of words, we should be aware that some words that form a named entity should be treated as a *complete* unit in the dependency trees. For example, the GPE entity "*Hong Kong*" is a complete unit and it should form a subtree in the following example. The same applies to the EVENT entity "*seminar on the actual practice of tax reform*".

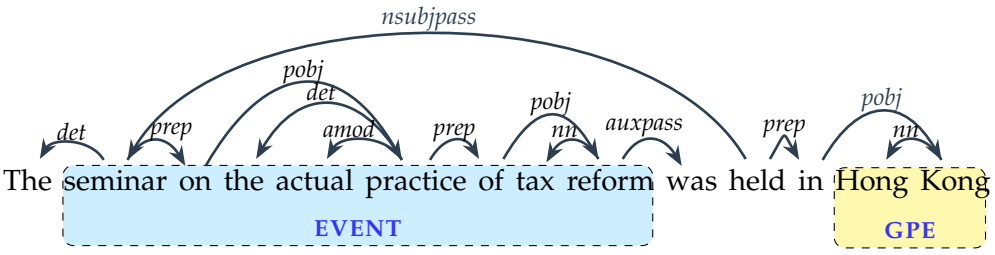


FIGURE 1.2: Example sentences annotated with named entities and dependencies in the OntoNotes 5.0 dataset.

By observing the named entity datasets with dependency annotations, we found that most of the entities often form dependency subtrees in a sentence (Jie, Muis, and Lu, 2017). On the other hand, certain entities have strong correlations with the dependency relations (Jie and Lu, 2019). For example, in the OntoNotes dataset (Weischedel et al., 2013), 61% of NORP (i.e., Nationalities or religious or political groups) entities attach to the dependency relations *amod* (i.e., adjectival modifier).

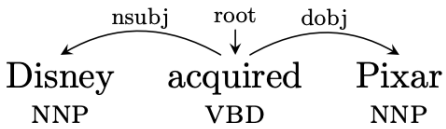
The major research challenge is how we can make use of these two types of relationships to improve our named entity recognition systems. We ask the following three research questions:

- *We know entities form subtrees in the dependencies. Can we impose certain constraints in the NER models to filter out some candidates that cannot form subtrees?*
- *Besides, how to incorporate the complete structured information provided by the dependency trees for NER?*
- *What if we do not have the complete annotations for NER? How does the dependency help in such a scenario?*

We aim to design efficient models to solve these two research questions and capture the complete structured information in our model architecture (Jie, Muis, and Lu, 2017; Jie and Lu, 2019). The third question is an actual incomplete annotation scenario (Jie et al., 2019) and the relationship between the dependency and named entities may take a role in such a scenario.

1.2.2 Dependencies in Semantic Parsing

Semantic parsing is a fundamental task within the field of natural language processing (NLP). Semantic parsing aims to transform the natural language sentences into machine-interpretable meaning representations automatically. On the other hand, the previous work (Reddy et al., 2016) shows that dependency trees leverage shallow semantic information.



(a) The dependency tree for *Disney acquired Pixar*.

$$(\text{nsubj}\ (\text{dobj}\ \text{acquired}\ \text{Pixar})\ \text{Disney})$$

(b) The s-expression for the dependency tree.

$$\lambda x.\, \exists yz.\, \text{acquired}(x_e) \wedge \text{Disney}(y_a) \wedge \text{Pixar}(z_a) \wedge \text{arg}_1(x_e, y_a) \wedge \text{arg}_2(x_e, z_a)$$

(c) The composed lambda-calculus expression.

FIGURE 1.3: Example of transforming dependency into logical forms (taken from Reddy et al. (2016)).

The underlying observation is helpful for extracting semantic meaning representation from the dependency trees. Reddy et al. (2016) and Reddy et al. (2017) proposed a supervised learning approach to transform the universal dependencies into lambda-calculus expression. Similarly, Wang, Xue, and Pradhan (2015a) proposed a transition-based algorithm to convert the dependency trees into abstract meaning representations (AMR). Universal dependencies are obtained by training the dependency parsers using the Universal dependency treebanks. They demonstrated the new state-of-the-art semantic parsing results on certain datasets. However, their semantic parsing performance largely depends on the quality of the dependency parsers. Without enough training data, it is not guaranteed that we can have a high-quality dependency parser. Thus, we raise the following research question:

- *Can we extract the dependency tree without the dependency annotations in the semantic parsing task?*

Our goal here is to design a structured model that regards the dependency tree as a latent variable in the semantic parsing task.

1.3 Thesis Outline

The thesis is organized as follows:

- Chapter 2 presents the background of named entity recognition and semantic parsing. We introduce the task definitions, the current approaches and the challenges in these structured prediction tasks.
- Chapter 3 presents the structured relationships between the dependency trees and semantically meaningful named entities. We propose our dependency-guided model based on the semi-Markov conditional random fields (semi-CRF) and conduct experiments on several datasets. The experimental results demonstrate that our model is much more efficient than the semi-CRF and achieves comparable or even better performance than the semi-CRF model.
- Chapter 4 further shows deeper relationships between the dependency trees and named entities, including the long-distance interactions and correlations between the dependency relations and the named entity types. In order to fully exploiting these relationships, we propose a dependency-guided LSTM CRF model to capture these properties. Our large-scale experiments on the datasets with four languages demonstrate the proposed model is effective across different languages and the dependencies are consistently helpful.
- Chapter 5 presents a practical scenario of incomplete annotations in NER. Specifically, we do not assume O labels are available in incomplete annotations. We propose the iterative training solution and discuss the inefficiency of this approach. However, such an iterative approach is not efficient during training. In order to address the inefficiency, we further present the dependency-based solution to this problem as our future work.
- Chapter 6 further studies the semantics leveraged from the dependency trees. We build the connections between the dependency trees and the meaning representations in semantics. We then present a dependency-based hybrid tree representation to capture the interaction between the natural language and the tree-structured meaning representation. We discuss the expressiveness of such a hybrid tree representation. Such hybrid trees are latent and can be learned by a structured latent-variable model. We also present a potential dependency hybrid tree solution to recently popular meaning representation, SQL.
- Chapter 7 presents the potential research directions besides the contribution in this thesis, such as joint dependency parsing and named entity recognition. We also argue that the linear-chain model might not always be the best fit for any languages and propose some future directions in better design of a universal named entity recognition system in terms of different languages. We also point out some future research directions for universal semantic parsing using one framework for different types of meaning representations.
- Chapter 8 concludes the strong connections between the dependency trees and semantic representation in natural language as well as how dependencies benefit

the language understanding. We summarize the overall research contribution in this thesis and what we would like to do in the future.

1.4 Contributions

Our overall contributions throughout the PhD study can be summarized as follows:

- Statistically, we found the strong structured relationship between the dependency trees and named entities. Such a relationship leverages the named entities often form subtrees in the dependency trees and attach to certain specific dependency relations.
- Through the observation, we are able to make use of this property to build efficient dependency-guided models for named entity recognition (NER). Motivated by the structured relationship, we can reduce the search space for an NER model (Jie, Muis, and Lu, 2017) to have efficient inference speed. We developed an efficient dependency-guided model (DGM) based on the semi-Markov CRF. Such a DGM model enjoys the linear-time average complexity while having comparable performance as the semi-Markov CRF which has L-times larger complexity.
- On the other hand, such a relationship allows us to design a better dependency-guided neural architecture (Jie and Lu, 2019) rather than simply bidirectional long short-term memory (BiLSTM) networks (Hochreiter and Schmidhuber, 1997). We demonstrate that our dependency-guided neural architecture is able to improve the NER performance by capturing the above relationship between the named entities and dependencies on the datasets with several languages.
- We introduce a practical scenario of named entity recognition with incomplete annotations (Jie et al., 2019) and propose an iterative training solution to this problem. Our scenario does not assume O labels are always available in practice and the proposed approach iteratively recovers the O labels. Though this solution achieves state-of-the-art performance in such a scenario, the training time is extremely expensive. To find an efficient solution, we further discuss a dependency-based approach by applying the relationships presented in the above sections as our future work.
- We further explore the deeper relationship between the dependency trees and semantic representations. Though the previous work (Reddy et al., 2016) has been working on transforming the dependency trees (with shallow semantics) into meaning representation, we focus on extracting the meaningful dependency trees without explicit dependency tree annotations (Jie and Lu, 2018). We present a novel *dependency-based hybrid tree* representation that captures both words and semantics in a joint manner. As a result, such a dependency-based hybrid tree largely benefits the task of semantic parsing, especially for languages with flexible word order.

Chapter 2

Background: Named Entity Recognition and Semantic Parsing

This chapter introduces the background knowledge of two structured prediction tasks: named entity recognition (Yadav and Bethard, 2018; Li et al., 2018) and semantic parsing (Kamath and Das, 2018). Besides, we present the common approaches to some of the research challenges in these two fields.

2.1 Named Entity Recognition

Named entity recognition is one of the most fundamental tasks in NLP. The task aims to identify the named entities in unstructured text into predefined categories (Grishman and Sundheim, 1996) such as person names, organization, locations, quantities, etc. One standard approach to NER is to regard the problem as a sequence labeling problem, where each word is assigned a tag, indicating whether the word belongs to part of any named entity or appears outside of all entities. More specifically, Ratinov and Roth (2009) designed different tagging schemes to fully encode different properties of the entity tags.

NER is an important step for many downstream tasks or applications such as relation extraction, question answering, semantic parsing, etc. The goal of relation extraction is to predict a relation between a pair of entities in a sentence or document. A number of research Miwa and Bansal (2016) and Xu et al. (2016) has been using the dependency trees to explicitly encode relation representations for pairs of entities. Dong and Lapata (2016) and Dong and Lapata (2018) proposed end-to-end neural models with pre-processed named entities for the semantic parsing tasks. Besides, as named entities are semantically meaningful,

2.1.1 Resources

To date, there are some benchmark datasets (especially for English) and well-known tools for NER. We summarize the commonly-used resources in the research community. The CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) datasets contains the English portion taken from the Reuters news stories[1] between August 1996 and August 1997, and the German data taken from the ECI Multilingual Text Corpus with

[1] http://www.reuters.com/researchandstandards/

articles written at the end of August 1992. The datasets are pre-processed with automatic tokenization, part-of-speech tagging and chunking with existing models. Entities are annotated by several annotators. Similar to the CoNLL-2003 datasets, the CoNLL-2002 (Tjong Kim Sang, 2002) contains the portion of Spanish and Dutch languages. These datasets have been benchmark datasets to evaluate recent feature-based (Ling and Weld, 2012; Finkel, Grenager, and Manning, 2005) and neural models (Collobert, Kavukcuoglu, and Farabet, 2011; Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016; Peters et al., 2018a; Devlin et al., 2019) so far.

Pradhan et al. (2012) and Pradhan et al. (2013) further proposed a much larger dataset, OntoNotes 5.0 for the CoNLL-2012 shared task[2]. The task is originally crafted for coreference resolution in multiple languages. As it provides the named entity annotations, we can use it to evaluate the current NER models. The evaluation is more reliable as the dataset contain a much larger volume of data and more entity types to evaluate a model's effectiveness and robustness. Recent research works (Li et al., 2017; Ghaddar and Langlais, 2018; Jie and Lu, 2019; Liu, Yao, and Lin, 2019) also evaluate their models on this dataset.

Among the existing NER systems, Stanford NER (Finkel, Grenager, and Manning, 2005)[3] and Spacy[4] are the most popular tools in NLP. Another one that has a pretty nice demonstration is the AllenNLP package[5]. They implemented the state-of-the-art NER models with the ELMo contextualized representations (Peters et al., 2018a).

2.1.2 Approaches

The approaches for sequence labeling, or specifically NER can be largely categorized into feature-based and neural-based models.

Feature-based Approaches Previous approaches used sequence labeling models such as hidden Markov models (HMMs) (Zhou and Su, 2002), maximum entropy Markov models (MEMMs) (McCallum, Freitag, and Pereira, 2000), as well as linear-chain (Finkel, Grenager, and Manning, 2005) and semi-Markov conditional random fields (CRFs/semi-CRFs) (Sarawagi and Cohen, 2004). These models are feature-based approaches that require us to design binary features. For example, the word and/or part-of-speech tag can be a good feature indicator for certain entities. Also, the word shape and whether a word is capital contribute to the existence of the named entities. Designing better features in a model results in better performance in NER (Lin and Wu, 2009; Passos, Kumar, and McCallum, 2014). Though we know some of the useful feature templates, it is not realistic for us to propose all useful features and these features are often sparse.

Neural-based Approaches While most research efforts exploited standard word-level features (Ratinov and Roth, 2009), more sophisticated features such as word shape and part-of-speech tags can also be used (Finkel, Grenager, and Manning, 2005). Instead of

[2]http://conll.cemantix.org/2012/data.html
[3]https://nlp.stanford.edu/software/CRF-NER.shtml
[4]https://spacy.io/
[5]https://allennlp.org/

hand-crafted features, recent literature has been focusing on designing different neural architectures to obtain the feature representations by making use of the word embeddings (Mikolov et al., 2013) and contextualized representation (Peters et al., 2018a; Devlin et al., 2019; Akbik, Blythe, and Vollgraf, 2018). Both the word embeddings and the contextualized representaiton (Smith, 2019) are trained on a large-scale corpus to obtain meaningful representations. The prevalent neural architecture for NER is using bidirectional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997) to encode the sentence as it is able to give us context representation. Chiu and Nichols (2016), Lample et al. (2016), and Ma and Hovy (2016) applied the BiLSTM architecture to obtain significantly better performance compared to the feature-based approaches. Afterward, contextualized representations such as ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2019) trained on the larger corpus further improved the performance on NER. Such improvements show that the contextualized representations contain even more useful information for the NER task.

2.1.3 Challenges

However, though these models have achieved state-of-the-art performance on standard English datasets, especially in the news domain with sufficient training data. There exist many challenges for NER in other languages, other domains and other scenarios when training data is incomplete or insufficient. Different languages have different properties in terms of grammar (Chomsky, 1956), word order (Greenberg, 1963), compositional structures, etc. A number of ongoing research has been focusing on the incomplete annotation problem in NER. Yang et al. (2018) showed the effectiveness of such a model on Chinese NER with incomplete annotations due to the fact that they required a certain number of fully annotated data to perform joint training. Greenberg et al. (2018) applied this model on a biomedical NER task and achieved promising performance with incomplete annotations. However, in their assumption for the incomplete annotations, the O labels are still considered, which we believe is not realistic. Carlson, Gaffney, and Vasile (2009) modified the structured perceptron algorithm and defined features only on the tokens with annotated labels in partially labeled sequences. Fernandes and Brefeld (2011) and Lou, Hamprecht, and HCI (2012) proposed to use a large-margin learning framework similar to structured support vector machines with latent variables (Yu and Joachims, 2009). Jie et al. (2019) presented a practical scenario for NER with incomplete annotations where we should not assume the O label annotations are always available. They solved the problems with an iteratively self-training approach. Mayhew et al. (2019) further presented an even more challenging setting and proposed a constrained binary learning method to learn the weights of a token being an O label. Besides incomplete annotations, we also suffer from the problem of insufficient training data in the field of NER. The common technique for this problem is to use transfer learning (Pan and Yang, 2009; Weiss, Khoshgoftaar, and Wang, 2016) to transfer the model parameters trained on the source data with fruitful annotations to target data where training data is insufficient (Yang, Salakhutdinov, and Cohen, 2017; Lin and Lu, 2018). Such a technique also applies to the cross-domain scenario. For example, we can train an NER model on the news domain and fine-tune the model on the social media domain (e.g., Twitter dataset).

2.2 Semantic Parsing

Semantic parsing is a fundamental task within the field of natural language processing (NLP). Consider a natural language (NL) sentence and its corresponding meaning representation (MR) as illustrated in Figure 1.3. Semantic parsing aims to transform the natural language sentences into machine-interpretable meaning representations automatically. The task has been popular for decades and keeps receiving significant attention from the NLP community. Various systems (Zelle and Mooney, 1996; Kate, Wong, and Mooney, 2005; Zettlemoyer and Collins, 2005; Liang, Jordan, and Klein, 2011) were proposed over the years to deal with different types of semantic representations. Such models include structure-based models (Wong and Mooney, 2006; Lu et al., 2008; Kwiatkowski et al., 2010; Jones, Johnson, and Goldwater, 2012) and neural network based models (Dong and Lapata, 2016; Cheng et al., 2017).

2.2.1 Meaning Representations

The literature on semantic parsing has focused on various types of semantic formalisms. The λ-calculus expressions (Zettlemoyer and Collins, 2005) have been popular and widely used in semantic parsing tasks over recent years (Dong and Lapata, 2016; Gardner and Krishnamurthy, 2017; Reddy et al., 2016; Reddy et al., 2017; Susanto and Lu, 2017a; Cheng et al., 2017). Dependency-based compositional semantics (DCS)[6] was introduced by Liang, Jordan, and Klein (2011), whose extension, λ-DCS, was later proposed by Liang (2013). Various models (Berant et al., 2013; Wang, Berant, and Liang, 2015; Jia and Liang, 2016) on semantic parsing with the λ-DCS formalism were proposed. Graph-based formalism is later proposed to represent rich and structured semantic of natural languages. For example, Banarescu et al. (2013) proposed the abstract meaning representation (AMR) to map the natural language words into the sense in Propbank (Kingsbury and Palmer, 2002). Structured query language (SQL) (Finegan-Dollak et al., 2018) has been popular in recent years. Lots of research efforts have been focusing on building various SQL datasets (Yu et al., 2018a; Zhong, Xiong, and Socher, 2017) and models (Lin et al., 2019) to accelerate the research progress in this field. In this work, we focus on the tree-structured semantic formalism which has been examined by various research efforts (Wong and Mooney, 2006; Kate and Mooney, 2006; Lu et al., 2008; Kwiatkowski et al., 2010; Jones, Johnson, and Goldwater, 2012; Lu, 2014; Zou and Lu, 2018).

2.2.2 Approaches

We summarize recent approaches on lambda-calculus expressions, tree-structured semantics, AMR and SQL meaning representations in this section.

Lambda-calculus expressions Similar to NER, early approaches (Cai and Yates, 2013; Poon, 2013; Berant and Liang, 2014; Krishnamurthy, 2016; Krishnamurthy, Dasigi, and Gardner, 2017) are mostly feature-based or rule-based approaches. For example, Krishnamurthy and Mitchell (2015) proposed a rule-based semantic parser containing three

[6] Unlike ours, their work captures dependencies between semantic units but not natural language words.

steps: CCG syntactic parsing, entity linking and semantic analysis which assigns a logical form to each word. Lu and Ng (2011) and Lu (2014) proposed the hybrid tree model for tree-structured logical forms. Lately, Dong and Lapata (2016) first proposed to use the sequence-to-sequence neural model with attention for lambda-calculus expressions. Reddy et al. (2016) and Reddy et al. (2017) proposed a linguistically motivated procedure to transform syntactic dependencies into logical forms.

Functional Query Logical Forms Wong and Mooney (2006) proposed the WASP semantic parser that regards the task as a phrase-based machine translation problem. Lu et al. (2008) proposed a generative process to generate natural language words and semantic units in a joint model. The resulting representation is called *hybrid tree* where both natural language words and semantics are encoded into a joint representation. The UBL-s (Kwiatkowski et al., 2010) parser applied the CCG grammar (Steedman, 1996) to model the joint representation of both semantic units and contiguous word sequences which do not overlap with one another. Jones, Johnson, and Goldwater (2012) applied a generative process with Bayesian tree transducer and their model also simultaneously generates the meaning representations and natural language words. Lu (2014) and Lu (2015) proposed a discriminative version of the hybrid tree model of (Lu et al., 2008) where richer features can be captured. Dong and Lapata (2016) proposed a sequence-to-tree model using recurrent neural networks where the decoder can branch out to produce tree structures. Susanto and Lu (2017b) augmented the discriminative hybrid tree model with a multilayer perceptron and achieved state-of-the-art performance.

Abstract Meaning Representations Existing work focus on the datasets released by LDC[7]. Flanigan et al. (2014) proposed the first graph-based system[8] for automatic AMR parsing. The core of their algorithm is to use a maximum spanning tree (MST) algorithm to construct the graph representation. Artzi, Lee, and Zettlemoyer (2015) proposed a CCG parsing algorithm for AMR parsing. Lyu and Titov (2018) introduced a neural parser which treats alignments as latent variables within a joint probabilistic model. Wang, Xue, and Pradhan (2015a) proposed a two-stage framework that uses the transition-based algorithm to transforms the dependency tree into an AMR graph. Afterward, there is a lot of research efforts in designing different transition-based systems in terms of transition actions and different neural encoders (Wang, Xue, and Pradhan, 2015b; Wang et al., 2016; Misra and Artzi, 2016; Damonte, Cohen, and Satta, 2017; Vilares and Gómez-Rodríguez, 2018). With the thrive of sequence-to-sequence model for semantic parsing (Dong and Lapata, 2016), Konstas et al. (2017) presented a novel training procedure that can perform AMR generation and achieve competitive results. As a rich semantic meaning representation, it can benefit some downstream applications such as machine reading comprehension (Sachan and Xing, 2016) and text generation (Song et al., 2016).

[7] https://catalog.ldc.upenn.edu/LDC2020T02
[8] http://github.com/jflanigan/jamr

SQL The task of translating natural-language-questions-to-SQL (NLQ2SQL) queries is commonly tackled using semantic parsing approaches, which aim at parsing natural language to a structured logic form. The NLQ2SQL task has mostly been formulated as a slot-filling problem that leverages the unique structure of SQL query. Zhong, Xiong, and Socher (2017) proposed the Seq2SQL method to prune the output space of the target query by leveraging the SQL structures. Xu, Liu, and Song (2017) proposed the SQLNet to tackled the "order-matter" problem in condition generation by using a sketch-based approach. Dong and Lapata (2018)proposed a two-stage method COARSE2FINE to first generate a sketch of the meaning of the given question and then fill in missing details based on both the natural language input and the sketch. Yu et al. (2018b) further improved SQLNet to TypeSQL by capturing the rare entities and numbers in natural language questions and utilizing the type information.

A number of researchers also began to address the NLQ2SQL task by directly generating the targeted SQL queries using sequence-to-sequence (Seq2Seq) based methods (Dong and Lapata, 2016; Sutskever, Vinyals, and Le, 2014; Wang, Brockschmidt, and Singh, 2018), which first encode the natural language questions as the vector representations and then decode the encoded vectors into the corresponding SQL queries. Several related approaches (Wang, Brockschmidt, and Singh, 2018) are developed to predict the SQL queries by copying a token from the source question using a copy mechanism. The NLQ2SQL task has also been solved in a unified framework for ten different natural language processing tasks (McCann et al., 2018). In order to effectively solve the NLQ2SQL task, the table schema (Wang, Brockschmidt, and Singh, 2018; Lukovnikov et al., 2018) or executed query answers against the database (Pasupat and Liang, 2015; Yih et al., 2015; Yin et al., 2016) are able to provide indirectly information to guide the SQL query generation tasks.

Chapter 3

Dependency-Guided Named Entity Recognition

3.1 Motivation

Existing research efforts have exploited dependency structured information by designing dependency-related local features that can be used in the NER models (Sasano and Kurohashi, 2008; Ling and Weld, 2012; Cucchiarelli and Velardi, 2001). Figure 3.1 shows two example phrases annotated with both dependency and named entity information. The local features are usually the head word and its part-of-speech tag at current position. For example, "*Shlomo*" with entity tag B-PER in the first sentence has two local dependency features, head word "*Ami*" and head tag "NNP". However, such a simple treatment of dependency structures largely ignores the global structured information conveyed by the dependency trees, which can be potentially useful in building NER models.

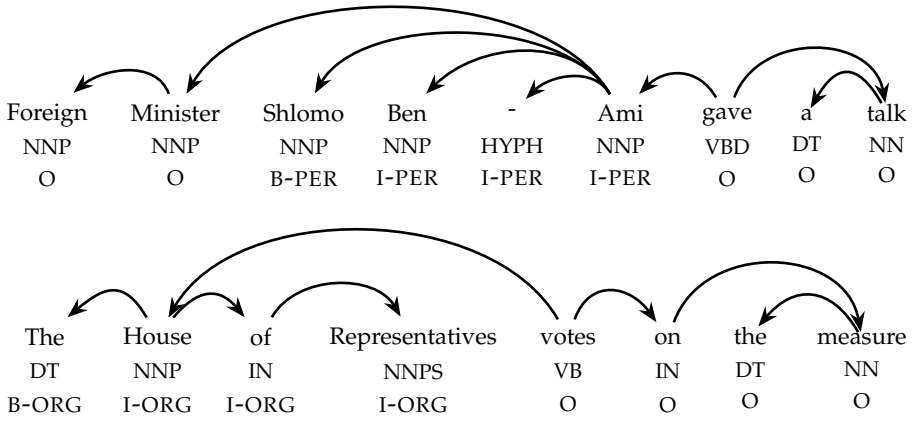


FIGURE 3.1: Two sentences annotated with both dependency and named entity information. The edges on top of words represent the dependencies and the labels with IOB encoding are the entity types.

One key observation we can make in Figure 3.1 is that named entities are often covered by a single or multiple consecutive dependency arcs. In the first example, the named entity "*Shlomo Ben - Ami*" of type PER (*person*) is completely covered by the single dependency arc from "*Ami*" to "*Shlomo*". Similarly, the named entity "*The House of Representatives*" of type ORG (*organization*) in the second example is covered by

multiple arcs which are adjacent to each other. Such information can potentially be the global features we can obtain from the dependency trees. This leads to the following questions: 1) can such global structured information conveyed by dependency trees be exploited for improved NER, and 2) if so, how to build new NER models where such information can be explicitly incorporated?

With these two questions, we perform some investigations on how to better utilize the structured information conveyed by dependency trees for building novel models for improved named entity recognition. The model assumes the availability of dependency trees before performing NER, which can be obtained from a dependency parser or given as part of the input. Unlike existing approaches that only exploit dependency structures for encoding local features, the model is able to explicitly take into account the global structured information conveyed by dependency trees when performing learning and inference. We call our proposed NER model the *dependency-guided* model (DGM), and build it based on the conventional semi-Markov conditional random fields (semi-CRFs) (Sarawagi and Cohen, 2004), a classic model used for information extraction.

3.2 Related Work

Named entity recognition has a long history in the field of natural language processing. One standard approach to NER is to regard the problem as a sequence labeling problem, where each word is assigned a tag, indicating whether the word belongs to part of any named entity or appears outside of all entities. Previous approaches used sequence labeling models such as hidden Markov models (HMMs) (Zhou and Su, 2002), maximum entropy Markov models (MEMMs) (McCallum, Freitag, and Pereira, 2000), as well as linear-chain (Finkel, Grenager, and Manning, 2005) and semi-Markov conditional random fields (CRFs/semi-CRFs) (Sarawagi and Cohen, 2004). Muis and Lu (2016b) proposed a weak semi-CRFs model which has a lower complexity than the conventional semi-CRFs model while still having a higher complexity than the linear-chain CRFs model. Our model is proved to have the same time complexity as linear-chain CRFs model in the average case. The quality of the CRFs model typically depends on the features that are used. While most research efforts exploited standard word-level features (Ratinov and Roth, 2009), more sophisticated features can also be used. Ling and Weld (2012) showed that using syntactic-level features from dependency structures in a CRFs-based model can lead to improved NER performance. Such dependency structures were also used in the work by Liu, Huang, and Zhu (2010), where the authors utilized such structures for building a skip-chain variant of the original CRFs model. This shows that some simple structured information conveyed by dependency trees can be exploited for improved NER. In their skip-chain CRFs model, they simply added certain dependency arcs as additional dependencies in the graphical model, resulting in loopy structures. However, such a model did not explicitly explore the relation between entities and global structured information of the dependency trees. The authors also showed that such a model does not outperform a simpler approach that adds additional dependencies between similar words only on top of the original CRFs model. In this work, we also focus on utilizing dependency structures for improving

NER. Unlike previous approaches, we focus on exploiting the global structured information conveyed by dependency trees to improve the NER process. Comparing with the semi-CRFs model, our model is not only able to perform competitively in terms of performance, but also more *efficient* in terms of running time.

There are also some existing works that focus on improving the efficiency of NER and other information extraction models. For example, Okanohara et al. (2006) used a separate naive Bayes classifier to filter some entities during training and inference in their semi-CRFs based model. While the filtering process was used to reduce the computational cost of the semi-CRFs model, the model still needs to enumerate all the possible chunks. Yang and Cardie (2012) extended the original semi-CRFs for extracting opinion expressions and used the constituency parse tree information to avoid constructing implausible segments. Lu and Roth (2015) proposed an efficient and scalable model using hypergraph which can handle overlapping entities. Muis and Lu (2016a) extended the hypergraph representation to recognize both contiguous and discontiguous entities.

3.3 Background

Before we introduce our models, we would like to have a brief discussion on the relevant background. Specifically, in this section, we review the two classic models that are commonly used for named entity recognition, namely the linear-chain conditional random fields and the semi-Markov conditional random fields models.

3.3.1 Linear-chain CRFs

Conditional random fields, or CRFs (Lafferty, McCallum, and Pereira, 2001) is a popular model for structured prediction, which has been widely used in various natural language processing problems, including named entity recognition (McCallum and Li, 2003) and semantic role labeling (Cohn and Blunsom, 2005).

We focus our discussions on the linear-chain CRFs in this section. The probability of predicting a possible output sequence $\mathbf{y}$ (*e.g.*, a named entity label sequence in our case) given an input $\mathbf{x}$ (*e.g.*, a sentence) is defined as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(\mathbf{w^T f}(\mathbf{x}, \mathbf{y}))}{Z(\mathbf{x})} \tag{3.1}$$

where $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is the feature vector defined over $(\mathbf{x}, \mathbf{y})$ tuple, $\mathbf{w}$ is the weight vector consisting of parameters used for the model, and $Z(\mathbf{x})$ is the partition function used for normalization, which is defined as follows:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp(\mathbf{w}^T \mathbf{f}(\mathbf{x}, \mathbf{y})) \tag{3.2}$$

In a (first-order) linear-chain CRF, the partition function for a input of length n can also be written as follows:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp \sum_{(y', y, i) \in \mathcal{E}(\mathbf{x}, \mathbf{y})} \mathbf{w}^T \mathbf{f}(\mathbf{x}, y', y, i) \tag{3.3}$$

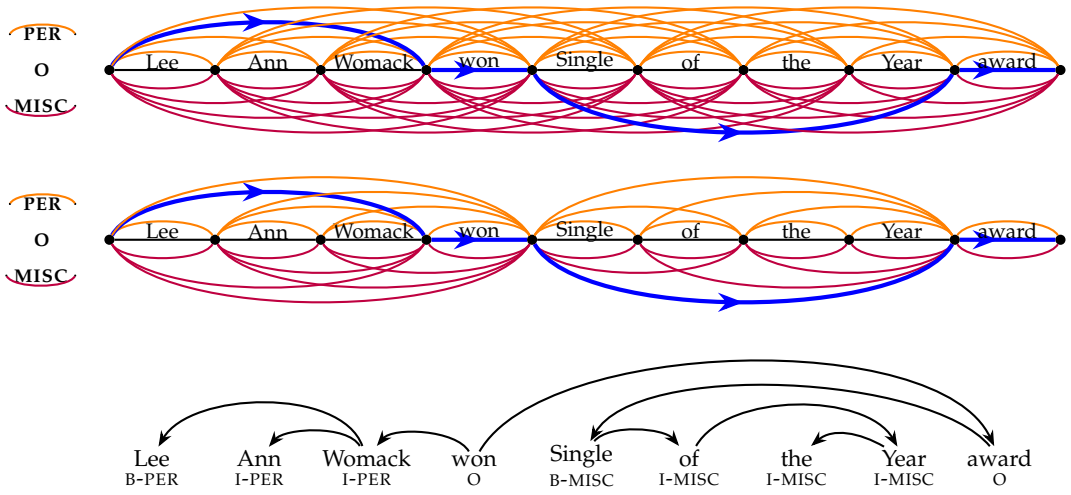


FIGURE 3.2: Illustrations of possible combinations of entities for the conventional semi-CRFs model (top) and our DGM model (middle), as well as the example sentence with its dependency structure (bottom).

where $\mathcal{E}(\mathbf{x}, \mathbf{y})$ is the set of edges which defines the input $\mathbf{x}$ labeled with the label sequence $\mathbf{y}$ and $\mathbf{f}(\mathbf{x}, y', y, i)$ is a local feature vector defined at the i-th position of the input sequence. T is the set of all output labels. The above function can be computed efficiently using dynamic programming. It can be seen that the time complexity of a linear-chain CRFs model is $\mathcal{O}(n|T|^2)$.

We aim to minimize the negative joint log-likelihood with L_2 regularization for our dataset:

$$\mathcal{L}(\mathbf{w}) = \sum_i \log \sum_{\mathbf{y}'} \exp(\mathbf{w}^T \mathbf{f}(\mathbf{x}_i, \mathbf{y}')) - \sum_i \mathbf{w}^T \mathbf{f}(\mathbf{x}_i, \mathbf{y}_i) + \lambda \mathbf{w}^T \mathbf{w} \tag{3.4}$$

where $(\mathbf{x}_i, \mathbf{y}_i)$ is the i-th training instance and λ is the L_2 regularization coefficient.

The objective function is convex and we can make use of the L-BFGS (Byrd et al., 1995) algorithm to optimize it. The partial derivative of $\mathcal{L}$ with respect to each parameter w_k is:

$$\frac{\partial \mathcal{L}}{\partial w_k} = \sum_i \Big(\mathbf{E}_{p(\mathbf{y}'|\mathbf{x}_i)} \left[f_k(\mathbf{x}_i, \mathbf{y}') \right] - f_k(\mathbf{x}_i, \mathbf{y}_i) \Big) + 2\lambda w_k$$

3.3.2 Semi-Markov CRFs

The semi-Markov conditional random fields, or semi-CRFs (Sarawagi and Cohen, 2004) is an extension of the standard linear-chain CRFs. Different from linear-chain CRFs, which makes use of simple first-order Markovian assumptions, the semi-CRFs assumes that the transitions between words inside a span (*e.g.*, an entity consisting of multiple words) can be non-Markovian. Such an assumption allows more flexible non-Markovian features to be defined. The resulting model was shown to be more effective than its linear-chain counterpart in information extraction tasks.

The partition function in this setting becomes:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp \sum_{(y', y, i-l, i) \in \mathcal{E}(\mathbf{x}, \mathbf{y})} \mathbf{w}^T \mathbf{f}(\mathbf{x}, y', y, i-l, i) \tag{3.5}$$

where $\mathbf{f}(\mathbf{x}, y', y, i-l, i)$ represents the local feature vector at position i with a span of size l. In this case, the time complexity becomes $\mathcal{O}(nL|T|^2)$. The value L is the maximal length of the spans the model can handle.

Consider the sentence "*Lee Ann Womack won Single of the Year award*". The upper portion of Figure 3.2 shows all the possible structures that are considered by the partition function with $L = 4$. These structures essentially form a compact lattice representation showing all the possible combinations of entities (with length restriction) that can ever appear in the given sentence. Each orange and red edge is used to represent an entity of type PER and MISC respectively. The black lines are used to indicate words that are not part of any entity. The directed path highlighted in bold blue corresponds to the correct entity information associated with the phrase, where "*Lee Ann Womack*" is an entity of type PER, and "*Single of the Year*" is another entity of type MISC (*miscellaneous*).

3.4 Our Model

The primary assumption we would like to make is that the dependency trees of sentences are available when performing the named entity recognition task.

3.4.1 NER with Dependency Features

One approach to exploit information from dependency structures is to design local features based on such dependency structures and make use of conventional models such as semi-CRFs for performing NER. Previous research efforts have shown that such features extracted from dependency structures can be helpful when performing the entity recognition task (Sasano and Kurohashi, 2008; Ling and Weld, 2012; Cucchiarelli and Velardi, 2001).

In Figure 3.2, we have already used a graphical illustration to show the possible combinations of entities that can ever appear in the given sentence. Each edge in the figure corresponds to one entity (or a single word that is outside of any entity – labeled with O). Features can be defined over such edges, where each feature is assigned a weight. Such features, together with their weights, can be used to score each possible path in the figure. When dependency trees are provided, one can define features involving some local dependency structures. For example, for the word "*Womack*", one possible feature that can be defined over edges covering this word can be of the form "*Womack* $\leftarrow$*won*", indicating there is an incoming arc from the word "*won*" to the current word "*Womack*". However, we note that such features largely ignore the global structured information as presented by the dependency trees. Specifically, certain useful facts such as the word "*Lee*" is a child of the word "*Womack*" and at the same time a grandchild of the word "*won*" is not captured due to such a simple treatment of dependency structures.

3.4.2 Dependency-Guided NER

The main question we would like to ask is: how should we make good use of the structured information associated with the dependency trees to perform named entity recognition? Since we are essentially interested in NER only, would there be some more global structured information in the dependency trees that can be used to guide the NER process?

From the earlier two examples shown in Figure 3.1 as well as the example shown in Figure 3.2 we can observe that the named entities tend to be covered by a single or multiple adjacent dependency arcs. This is not a surprising finding as for most named entities which convey certain semantically meaningful information, it is expected that there exist certain internal structures – *i.e.*, dependencies amongst different words within the entities. Words inside each named entity typically do not have dependencies with words outside the entities, except for certain words such as head words which typically have incoming arcs from outside words.

This finding motivates us to exploit such global structured information as presented by dependency trees for performing NER. Following the above observation, we first introduce the following notion:

Definition 1 A *valid span* either consists of a single word, or is a word sequence that is covered by a chain of (undirected) arcs where no arc is covered by another.

For example, in Figure 3.2, the word sequence "*Lee Ann Womack*" is a valid span since there exists a single arc from "*Womack*" to "*Lee*". The sequence "*Ann Womack won*" is also a valid span due to a chain of undirected arcs – one between "*Ann*" and "*Womack*" and the other between "*Womack*" and "*won*". Similarly, the single word "*Womack*" is also a valid span given the above definition.

Based on the above definition, we can build a novel dependency-guided NER model based on the conventional semi-CRFs by restricting the space of all possible combinations of entities to those that strictly contain only valid spans. This leads to the following new partition function:

$$Z(\mathbf{x}) = \sum_{(i,j)\in\mathcal{S}_L(\mathbf{x})} \sum_{y'\in T} \sum_{y\in T} \exp(\mathbf{w}^T \mathbf{f}(\mathbf{x}, y', y, i, j)) \tag{3.6}$$

where $\mathcal{S}(\mathbf{x})$ refers to the set of valid spans for a given input sentence $\mathbf{x}$, and $\mathcal{S}_L(\mathbf{x})$ refers to its subset that contains only those valid spans whose lengths are no longer than L.

We call the resulting model *dependency-guided model* (DGM). Figure 3.2 (middle) presents the graphical illustration of all possible paths (combination of entities) with $L = 4$ that our model considers. For example, since the word sequence "*Ann Womack won Single*" is not a valid span, the corresponding edges covering this word sequence is removed from the original lattice representation in Figure 3.2 (top). The edge covering the word sequence "*of the Year*" remains there as it is covered by the arc from "*of*" to "*Year*", thus a valid span. As we can see, the amount of paths that we consider in the new model is substantially reduced.

(A) Best-case Scenario (B) Worst-case Scenario

FIGURE 3.3: The best-case and worst-case scenarios of DGM.

3.4.3 Time Complexity

We can also analyze the time complexity required for this new model. We show example best-case and worst-case scenarios in Figure 3.3. For the former case there are $\mathcal{O}(n)$ valid spans. Thus the time complexity in the best case is $\mathcal{O}(n|T|^2)$, the same as the linear-chain CRFs model. For the latter, there are $\mathcal{O}(nL)$ valid spans, leading to the time complexity $\mathcal{O}(nL|T|^2)$ in the worst case, the same as the semi-Markov CRFs model.

Furthermore, we have the following theorem for the average-case time complexity:

Theorem 1 *The average-case time complexity of* DGM *is* $\mathcal{O}(n|T|^2)$.

Proof We refer the readers to our full paper (Jie, Muis, and Lu, 2017) with appendix[1]. We credit Aldrian with his substantial effort for the complexity derivation.

Now, this is a remarkable result, showing that the complexity of our DGM model in its average case is still linear in the sentence length n even if we set L to its maximal value n. This is also true empirically, which we show in the supplementary material (S.3.2). So, while our model retains the ability to capture non-Markovian features like semi-Markov CRFs, it is more scalable and can be used to extract longer entities.

Besides the standard DGM model defined above, we also consider a variant, where we restrict the chain (of arcs) to be of length 1 (*i.e.*, single arc) only. We call the resulting model DGM-S, where S stands for *single arc*. Such a simplified model will lead to an even simpler lattice representation, resulting in even less running time. However it may also lead to lower performance as such a variant might not be able to capture certain named entities. In one of the later sections, we will conduct experiments on such a model and compare its performance with other models.

3.4.4 Number of Edges

The number of edges in a graphical model is proportional to the time complexity of training the model, and so in the interest of calculating the time complexity of the models, in this section we show theoretically that on average case, the number of edges in our DGM model is linear in terms of n, the number of words in a sentence. Then for all models we show empirically the relationship between the number of edges and the sentence length n.

[1] https://arxiv.org/pdf/1810.08436.pdf

Empirical Count

In this subsection, we calculate empirically the relationship between the number of edges present per sentence in each model and n, the sentence length, to provide an evidence for the theoretical analysis presented in previous subsection in a dataset where most of the trees are projective. We compute this by averaging the number of edges per sentence in all 7 subsections of the dataset. Figure 3.4 shows the result. We can see that the number of edges in semi-CRFs model is much more than that of DGM and DGM-S. All three models have a linear complexity in terms of n. However, note that the semi-CRFs model comes with a scaling factor L, while our models do not. For the purpose of the calculation of number of edges in the semi-CRFs model, we used $L = 8$, the same as the one we used in the main paper.

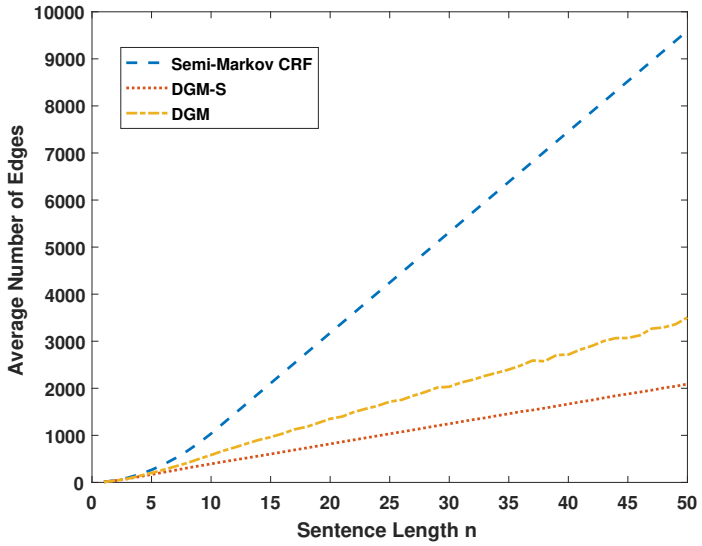


FIGURE 3.4: The average number of edges over all sentences in all datasets with respect to sentence length n. For semi-CRF, we set $L = 8$. Also note that in the dataset we have $|T| = 5$ (PER, ORG, GPE, MISC, and special label O denoting non-entities)

.

We also calculated the average number of edges involved per token by averaging the average number of edges per token, over all sentences. More formally, let the number of sentences in a dataset be N, the number of edges in i-th sentence be E_i, and the length of i-th sentence be n_i. Then the average number of edges involved per token $\bar{E}$ is:

$$\bar{E} = \frac{\sum_{i=1}^{N} \frac{E_i}{n_i}}{N}.$$

Table 3.1 shows the average number of edges involved per token for different models. Since the number of edges in our models is linear in terms of n, the average number of edges per token will be constant, plus some small variance accounting for the boundary

cases in the data. This also explains the fact that both DGM-S and DGM models perform much faster compared to the semi-CRF model.

	ABC	CNN	MNB	NBC	P2.5	PRI	VOA	Avg.
DGM-S	39.1	39.8	37.9	38.9	40.9	39.8	40.7	39.5
DGM	57.3	62.9	53.7	56.2	71.3	59.8	65.5	61.0
semi-CRFs	117.8	133.9	110.3	119.8	172.4	126.4	144.3	132.1

TABLE 3.1: The average number of possible edges involved in each token when we construct the model.

3.5 Experimental Setup

For experiments, we followed (Finkel and Manning, 2009) and used the Broadcast News section of the OntoNotes dataset. Instead of using its earlier 2.0 release, we used the final release – release 5.0 of the dataset, which is available for download[2]. The dataset includes the following 6 subsections: ABC, CNN, MNB, NBC, PRI and VOA. Moreover, the current OntoNotes 5.0 release also includes some English P2.5 data, which consists of corpus translated from Chinese and Arabic.[3] Following (Finkel and Manning, 2009), we split the first 75% of the data for training and performed evaluations on the remaining 25%. We set $L = 8$, which can cover all entities in our dataset, and developed the L_2 coefficient using cross-validation (see supplementary material S.1 for details). Following previous works on dependency parsing (Chen and Manning, 2014), we preprocessed the dataset using the Stanford CoreNLP[4] to convert the constituency trees to basic Stanford dependency (De Marneffe, MacCartney, and Manning, 2006) trees. In our NER experiments, in addition to using the given dependency structures converted from the constituency trees, we also trained a popular transition-based parser, MaltParser[5] (Nivre, Hall, and Nilsson, 2006), using the training set and then used this parser to predict the dependency trees on the test set to be used in our model.

We also looked at the SemEval-2010 Task 1 OntoNotes English corpus[6], which contains sentences with both dependency and named entity information. Although this dataset is a subset of the OntoNotes dataset, it comes with manually annotated dependency structures.

[2]https://catalog.ldc.upenn.edu/LDC2013T19/

[3]The OntoNotes 5.0 dataset also contains the earlier release from OntoNotes 1.0 to OntoNotes 4.0. However, we found the number of sentences of the Broadcast News in the current OntoNotes 5.0 dataset does not match the number reported in (Finkel and Manning, 2009; Finkel and Manning, 2010), which was based on OntoNotes 2.0. Furthermore, they removed some instances which are inconsistent with their model. We thus cannot conduct experiments to compare against the results reported in their work.

[4]http://stanfordnlp.github.io/CoreNLP/

[5]http://maltparser.org/

[6]https://catalog.ldc.upenn.edu/LDC2011T01/

3.5.1 Features

In order to make comparisons, we implemented a linear-chain CRFs model as well as a semi-Markov CRFs model to serve as baselines. The features we used are basic features which are commonly used in linear-chain CRFs and semi-CRFs. For the linear-chain CRFs, we consider the current word/POS tag, the previous word/POS tag, the current word shape, the previous word shape, prefix/suffix of length up to 3, as well as transition features. For word shape features, we followed (Finkel et al., 2005) to create them. For the semi-CRFs model, we have the following features for each segment: the word/POS tag/word shape before and after the segment, indexed words/POS tags/word shapes in current segment, surface form of the segment, segment length, segment prefix/suffix of length up to 3, the start/end word/tags of current segment, and the transition features.

Inspired by Ling and Weld (2012), we applied the following dependency features for all models: (1) current word in current position/segment and its head word (and its dependency label); (2) current POS tag in current position/segment and its head tag (and dependency label). More details of features can be found in the supplementary material (S.4).

3.5.2 Feature Representations

This section gives an example on the feature representation defined in Features section in the main paper. For illustration purpose, we take the first sentence of Figure 1 in the main paper as an example. The code released has the detailed feature implementation as well.

Say that we are currently at the position of word "*Ami*". The features in the linear-chain CRF model is represented in table 3.2. In semi-CRFs model, the features are defined over segment. We take the segment "*Shlomo Ben - Ami*" as an example to describe the features in Table 3.3.

3.5.3 Data Statistics

There are in total 18 well-defined named entity types in the OntoNotes 5.0 dataset. Since majority of the entities are from the following three types: PER (*person*), ORG (*organization*) and GPE (*geo-political entities*), following (Finkel and Manning, 2009), we merge all the other entity types into one general type, MISC (*miscellaneous*). Table 3.4 shows the statistics of total number of sentences and entities.

To show the relationships between the named entities and dependency structures, we present the number and percentage of entities that can be handled by our DGM-S model and DGM model respectively. The entities that can be handled by DGM-S should be covered by a single arc as indicated in our model section. As for DGM, the entity spans should be *valid* as in definition 1. Overall, we can see that 93.3% and 92.5% of the entities can be handled by the DGM-S model in training set and test set, respectively. These two numbers for DGM are much higher – 99.7% and 99.6%. With the predicted dependency structures in test set, 91.7% of the entities can be handled by the DGM-S model, while for DGM it is 97.4%.

Features	Examples
current word	*Ami*
current POS	NNP
previous word	-
previous POS	HYPH
current word shape	Xxx
previous word shape	-
prefix up to length 3	*A*, *Am*, *Ami*
suffix up to length 3	*i*, *mi*, *Ami*
transition	I-PER + I-PER
Dependency features	**Examples**
current word + head	*Ami* + *gave*
current word + head + label	*Ami* + *gave* + nsubj
current POS + head POS	NNP + VBD
current POS + head POS + label	NNP + VBD + nsubj

TABLE 3.2: The features for the example sentence in the linear-chain CRFs model.

Features	Examples
word before segment	*Minister*
POS before segment	NNP
word shape before segment	Xxxx
word after segment	*gave*
POS after segment	VBD
word shape after segment	xxxx
prefix up to length 3	*S*, *Sh*, *Shl*
suffix up to length 3	*i*, *mi*, *Ami*
start word	start:+*Shlomo*
end word	end:+*Ami*
start POS	start POS:+NNP
end POS	end POS:+NNP
segment length	4
transition	O + PER
indexed word	1:*Shlomo*, 2:*Ben*, 3:-, 4:*Ami*
indexed POS	1:NNP, 2:NNP, 3:HYPH, 4:NNP
indexed shape	1:Xxxx, 2:Xxx, 3:-, 4:Xxx
the whole segment	*Shlomo Ben - Ami*
dependency	same as dependency features in Table 3.2

TABLE 3.3: The features for the example sentence in the semi-CRFs model.

These numbers confirm that it is indeed the case that most named entities do form

	# Sent.	# Entities		
		ALL	DGM-S	DGM
Train	9,996	18,855	17,584 (93.3%)	18,803 (99.7%)
Test	3,339	5,742	5,309 (92.5%)	5,720 (99.6%)

TABLE 3.4: Dataset statistics.

valid spans, even when using predicted dependency trees, and that such global structured information of dependency trees can be exploited for NER.

3.5.4 Detailed Data Statistics and Parameter Tuning

Table 3.5 shows the detailed statistics of all subsections. Overall, over 99.6% entities are representable in DGM model and around 91% to 94% entities are representable DGM-S model. We tuned the L_2 regularization parameter using 10-fold cross-validation for all

	Train				Test			
	# Sent.	# Entity			# Sent.	# Entity		
		ALL	DGM-S	DGM		ALL	DGM-S	DGM
ABC	873	1,365	1,281 (93.9%)	1,360 (99.6%)	292	444	415 (93.5%)	444 (100.0%)
CNN	4,318	6,627	6,113 (92.2%)	6,609 (99.7%)	1,440	1,620	1,474 (91.0%)	1,613 (99.6%)
MNB	477	690	653 (94.6%)	689 (99.9%)	160	177	162 (91.5%)	176 (99.4%)
NBC	480	922	868 (94.1%)	918 (99.6%)	161	343	312 (91.0%)	340 (99.1%)
P2.5	890	1,995	1,827 (91.6%)	1,988 (99.7%)	298	672	616 (91.7%)	669 (99.6%)
PRI	1,536	3,096	2,916 (94.2%)	3,090 (99.8%)	513	992	927 (93.5%)	990 (99.8%)
VOA	1,422	4,160	3,926 (94.4%)	4,149 (99.7%)	475	1,494	1,403 (93.9%)	1,488 (99.6%)
Total	9,996	18,855	17,584 (93.3%)	18,803 (99.7%)	3,339	5,742	5,309 (92.5%)	5,720 (99.6%)

TABLE 3.5: Dataset Statistics. The number of sentences and entities in the Broadcast News corpus of OntoNotes 5.0 dataset.

the models. Specifically for each model, we performed cross validation on the largest subsection, CNN, and obtained the best parameter with highest average F-score. We then used this parameter for other subsections as well. The values of L_2 regularization parameter we evaluated is $[0.0001, 0.001, 0.01, 0.1, 1]$. Specifically, we have four models and each of them is associated with two variants with or without dependency features. We obtained the best regularization parameter 0.1 for all the models except the DGM-S without dependency features, whose best regularization parameter is 1.

3.6 Results and Discussions

3.6.1 NER Performance

Following previous work (Finkel and Manning, 2009), we report standard F-score in this section (detailed results with precision and recall can be found in the supplementary material S.5). Table 3.6 shows the results of all models when dependency features are exploited. Specifically, it shows results when the given and predicted dependency trees are considered, respectively. Overall, the semi-Markov CRFs, DGM-S and DGM

models perform better than the linear-chain CRFs model. Our DGM model obtains an overall F-score at 80.5% and outperforms the semi-CRFs model significantly ($p < 0.001$ with bootstrap resampling (Koehn, 2004)). For individual subsection, our DGM also performs the best. On 2 out of the 7 subsections, our model's improvements over the baseline semi-CRFs model are significant ($p < 0.001$ with bootstrap resampling). For some other subsections like ABC, CNN, MNB, NBC and PRI, DGM has higher F-score than the semi-CRFs model but the improvements are not statistically significant. The results show that comparing with semi-CRFs, our DGM model, which has a lower average-case time complexity, still maintains a competitive performance. Such results confirm that the global structured information conveyed by dependency trees are useful and can be exploited by our DGM model.

Dependency	Model	ABC	CNN	MNB	NBC	P2.5	PRI	VOA	Overall
Given	Linear-chain CRFs	70.2	75.9	**75.7**	65.9	70.8	83.2	84.6	77.8
	Semi-Markov CRFs	**71.9**	**78.2**	**74.7**	**69.4**	73.5	**85.1**	85.4	79.6
	DGM-S	71.4	77.0	73.4	68.4	72.8	**85.1**	85.2	79.0
	DGM	**72.3**	**78.6**	**76.3**	**69.7**	**75.5**	**85.5**	**86.8**	**80.5**
Predicted	Linear-chain CRFs	68.4	75.4	74.4	66.3	70.8	83.3	83.7	77.3
	Semi-Markov CRFs	**71.6**	**78.0**	73.5	**71.5**	**73.7**	**84.6**	**85.3**	**79.5**
	DGM-S	70.6	76.4	73.4	68.7	71.3	**83.9**	84.4	78.2
	DGM	**71.9**	**77.6**	**75.4**	**71.4**	**73.9**	**84.2**	**85.1**	**79.4**

TABLE 3.6: NER results for all models, when given and predicted dependency trees are used and dependency features are used. Best values and the values which are not significantly different in 95% confidence interval are put in bold.

The performance of DGM-S is worse than that of semi-CRFs and DGM in general since there are still many named entities that can not be handled by such a simplified model (see Table 3.5). This model has the same time complexity as the linear-chain CRFs, but performs better than linear-chain CRFs, largely due to the fact that certain structured information of dependency trees are exploited in DGM-S. We note that such a simplified model obtains similar results as semi-CRFs for the two larger subsections, PRI and VOA. This is largely due to the fact that a larger percentage of the entities in these two subsections can be handled by DGM-S. It is noted that the performance of both semi-CRFs and DGM degrades when the predicted dependency trees are used instead of the given. The drop in F-score for DGM is more significant as compared to the semi-CRFs. Nevertheless, their results remain comparable. Such experiments show the importance of considering high quality dependency structures for performing guided NER in our model.

To understand the usefulness of the global structured information of dependency trees better, we conducted further experiments by excluding dependency features from our models. Such results are shown in Table 3.7. Our DGM model consistently performs competitively with the semi-CRFs model. The only exception occurs when the ABC subsection is considered and the predicted dependency trees are used (p=0.067). In general, we can see that when dependency features are excluded, the overall F-score for all models drop substantially. However, we note that for semi-CRFs, the drop in F-score is 1.1% for given dependency trees, and is 1.0% for predicted trees, whereas

(A) Given dependency tree (B) Predicted dependency tree

FIGURE 3.5: The effect of different dependency parses on the output of DGM. These are taken out from a part of a sentence. The NER result in (a) is correct, while (b) is not.

for DGM, the drops with given and predicted trees are both 0.6%. Overall, such results largely show that our proposed model is able to effectively make use of the global structured information conveyed by dependency trees for NER.

Dependency	Model	ABC	CNN	MNB	NBC	P2.5	PRI	VOA	Overall
Given	Linear-chain CRFs	66.5	74.1	74.9	65.4	70.8	82.9	82.3	76.3
	Semi-Markov CRFs	**72.3**	76.6	**75.0**	**69.3**	73.7	84.1	83.3	78.5
	DGM-S	69.4	76.1	73.4	68.0	72.5	85.2	85.1	78.6
	DGM	**72.7**	**77.2**	**75.8**	**68.5**	**76.8**	**86.2**	**85.5**	**79.9**
Predicted	Linear-chain CRFs	66.5	74.1	74.9	65.4	70.8	82.9	82.3	76.3
	Semi-Markov CRFs	**72.3**	**76.6**	**75.0**	**69.3**	**73.7**	84.1	83.3	**78.5**
	DGM-S	69.1	75.6	73.8	67.2	72.0	84.5	**84.2**	78.0
	DGM	71.3	**76.2**	**75.9**	**68.8**	**74.6**	**85.1**	**84.3**	**78.8**

TABLE 3.7: NER results for all models, when given and predicted dependency trees are used but dependency features are not used. Best values and the values which are not significantly different in 95% confidence interval are put in bold.

We have also conducted experiments on the widely-used NER dataset, CoNLL-2003, using the Stanford dependency parser[7] to generate the dependency trees. Using the same feature set that we describe in this work, our models do not achieve the state-of-the-art results on this dataset. However, they still perform comparably with the semi-CRFs model, while requiring much lesser running time. Note that since our goal in this work is to investigate the usefulness of incorporating the dependency structure information for NER, we did not attempt to tune the feature set to get the best result on a specific dataset. Also it is worth remarking that we still obtain a relatively good performance for our DGM model although the dependency parser is not trained within the dataset itself and that a correct dependency structure information is crucial for the DGM model.

3.6.2 Error Analysis

We provide a further analysis of how the dependency parsing performance affects NER based on Table 3.7. Because our model uses the dependency structure information directly instead of using them as features, we can analyze the influence of dependency

[7] http://nlp.stanford.edu/software/nndep.shtml

structures on NER more clearly. Specifically, we focus on how the dependency parsing results affect the prediction of our DGM model.

A typical error made by our model taken from the results is shown in Figure 3.5 where the dependency tree in Figure 3.5a is given by the conversion of constituent parse tree and the other one in Figure 3.5b is predicted from the MaltParser. The NER result on the left is correct while the one on the right is incorrect. Based on the predicted dependency structure in Figure 3.5b, there is no way to predict an entity type for "*USS Cole*" since this is not a valid span in DGM model. Furthermore, DGM can actually recognize "*Navy*" as an ORG entity even though the predicted dependency is incorrect. But the model considers "*USS Cole Navy*" as an entity due to the interference of other entity features (*e.g.*, NNP tag and Capitalized pattern) that "*USS Cole*" has. While with the given dependency tree, DGM considers "*USS Cole*" as a valid span and correctly recognizes it as a MISC entity.

3.6.3 Speed Analysis

From Figure 3.2 we can see that the running time required for each model depends on the number of edges that the model considers. We thus empirically calculated the average number of edges per word each model considers based on our training data. We found that the average number of edges involved in each token is 132 for the semi-CRFs model, while this number becomes 40 and 61 for DGM-S and DGM respectively. A lower number of edges indicates less possible structures to consider, and a reduced running time. See more detailed information in the supplementary material (S.3.2).

The results on training time per iteration (inference time) for all 7 subsections are shown in Figure 3.6. From the figure we can see that the linear-chain CRFs model empirically runs the fastest. The simple DGM-S model performs comparably with linear-chain CRFs. The semi-CRFs model requires substantially longer time due to the additional factor L (set to 8 in our experiments) in its time complexity. In contrast, our model DGM requires only 47% of the time needed for semi-CRFs for each training iteration, and requires 37% more time than the DGM-S model.

3.6.4 Results on SemEval-2010 Task 1 Dataset

The dataset in SemEval-2010 Task 1 is a subset of OntoNotes English corpus. The dependency and named entities information are annotated in this dataset. There are a total of 3,648 sentences with 4,953 entities in the training set, 741 sentences with 1,165 entities in the development set, and 1,141 sentence with 1,697 entities in the test set. In this dataset, we found that overall 92.7% of entities are representable in DGM-S and 96.6% of entities are representable in DGM.

Table 3.8 and Table 3.9 show the NER performance of all models on SemEval 2010 Task 1 dataset. In Table 3.8, the semi-CRF model with gold dependency features has a higher F-score while it is not significantly better than our DGM ($p = 0.44$). However, DGM performs worse when the predicted dependency is used as features since the percentages of entities representable in DGM is not as high as 99% in the other dataset, and also the predicted dependency information affects our DGM model structures. Furthermore, if we do not use the dependency features, both DGM and semi-CRFs perform

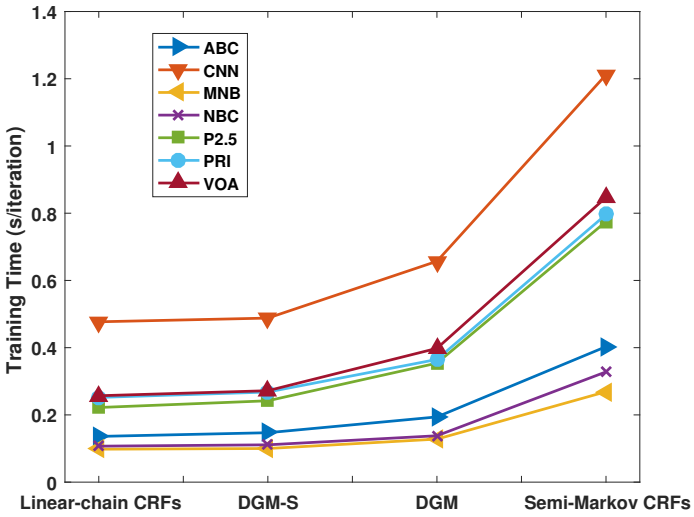


FIGURE 3.6: Training time per iteration of all the models.

	Dependency	Linear-chain CRFs			Semi-Markov CRFs			DGM-S			DGM		
		P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
SemEval 2010	Gold	75.8	72.2	73.9	77.3	73.8	**75.5**	76.1	72.4	74.2	77.0	73.0	**75.0**
	Predicted	75.2	71.1	73.1	77.2	73.2	**75.1**	76.0	72.2	74.1	76.4	71.8	74.1

TABLE 3.8: Named Entity Recognition Results on the SemEval 2010 Task 1 dataset. All the models in this table use the dependency information as features.

	Dependency	Linear-chain CRFs			Semi-Markov CRFs			DGM-S			DGM		
		P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
SemEval 2010	Gold	76.1	71.2	73.6	77.2	72.1	**74.5**	77.1	71.6	74.3	77.7	72.1	**74.8**
	Predicted							77.0	71.2	74.0	77.3	71.5	**74.3**

TABLE 3.9: Named Entity Recognition Results on the SemEval 2010 Task 1 dataset without dependency features. Note that DGM-S and DGM still utilize the dependency information to build the models.

similarly, with p-values of $p = 0.23$ and $p = 0.33$ when using gold and predicted dependency structures for DGM, respectively. The Semi-CRF model achieves 74.5% F-score while DGM (using gold dependency structures) achieves 74.8% F-score with a much faster training time per iteration (inference time). The inference time of semi-CRFs on this dataset is 2.25 times more than the inference time of DGM.

3.6.5 Full Results with Precision, Recall and F-score

This section presents the full results with precision, recall and F-score of all models in the paper. Table 3.10 and Table 3.11 show the results with dependency features while Table 3.12 and Table 3.13 show the results without dependency features. Recall that our

DGM-S and DGM models use the dependency structure information to build the models even if we don't use the dependency features.

	Linear-chain CRFs			Semi-Markov CRFs			DGM-S			DGM		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
ABC	71.5	68.9	70.2	71.7	72.2	**71.9**	71.3	71.5	71.4	72.2	72.4	**72.3**
CNN	76.7	75.1	75.9	78.3	78.2	**78.2**	77.2	76.9	77.0	78.7	78.6	**78.6**
MNB	80.8	71.2	75.7	77.4	72.2	**74.7**	76.5	70.5	73.4	78.8	73.9	**76.3**
NBC	69.0	63.0	65.9	70.7	68.2	**69.4**	70.3	66.7	68.4	70.3	69.1	**69.7**
P2.5	73.2	68.6	70.8	75.0	72.0	73.5	74.7	70.9	72.8	76.7	74.4	**75.5**
PRI	83.9	82.6	83.2	84.7	85.5	**85.1**	84.8	85.4	**85.1**	85.1	85.9	**85.5**
VOA	85.7	83.5	84.6	85.6	85.2	85.4	85.2	85.1	85.2	87.1	86.4	**86.8**
Overall	79.2	76.5	77.8	79.9	79.3	79.6	79.5	78.6	79.0	80.8	80.2	**80.5**

TABLE 3.10: Named Entity Recognition Results on the Broadcast News corpus of OntoNotes 5.0 dataset. All the models in this table are using the gold dependency information. Both DGM-S and DGM models apply the dependency information in two ways: building the model and as well as using them as features.

	Linear-chain CRFs			Semi-Markov CRFs			DGM-S			DGM		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
ABC	70.1	66.7	68.4	71.4	71.7	**71.6**	70.6	70.6	70.6	71.8	72.0	**71.9**
CNN	76.3	74.5	75.4	78.0	78.1	**78.0**	76.7	76.1	76.4	77.6	77.6	**77.6**
MNB	78.6	70.6	74.4	76.2	71.0	73.5	76.5	70.5	73.4	77.7	73.3	**75.4**
NBC	69.6	63.3	66.3	72.5	70.6	**71.5**	70.2	67.3	68.7	72.0	70.9	**71.4**
P2.5	73.4	68.3	70.8	75.2	72.3	**73.7**	73.2	69.6	71.3	74.7	73.2	**73.9**
PRI	83.9	82.7	83.3	84.2	85.0	**84.6**	83.7	84.0	**83.9**	83.7	84.7	**84.2**
VOA	84.8	82.7	83.7	85.4	85.2	**85.3**	84.7	84.0	84.4	85.3	84.8	**85.1**
Overall	78.8	75.9	77.3	79.8	79.3	**79.5**	78.8	77.6	78.2	79.6	79.2	**79.4**

TABLE 3.11: Named Entity Recognition Results on the Broadcast News corpus of OntoNotes 5.0 dataset. All the models in this table are using the predicted dependency information from MaltParser.

	Linear-chain CRFs			Semi-Markov CRFs			DGM-S			DGM		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
ABC	67.8	65.3	66.5	72.2	72.4	**72.3**	69.8	69.0	69.4	72.5	72.9	**72.7**
CNN	75.0	73.3	74.1	76.7	76.4	76.6	76.5	75.7	76.1	77.4	77.0	**77.2**
MNB	77.6	72.3	74.9	76.8	73.3	**75.0**	76.5	70.5	73.4	77.8	73.9	**75.8**
NBC	67.3	63.6	65.4	69.8	68.8	**69.3**	70.1	66.1	68.0	68.5	68.5	**68.5**
P2.5	73.4	68.3	70.8	75.2	72.3	73.7	76.4	69.0	72.5	77.8	75.7	**76.8**
PRI	83.6	82.2	82.9	83.9	84.3	84.1	85.0	85.4	85.2	85.9	86.5	**86.2**
VOA	83.2	81.4	82.3	83.5	83.0	83.3	85.5	84.7	85.1	86.1	84.9	**85.5**
Overall	77.5	75.1	76.3	78.8	78.1	78.5	79.5	77.6	78.6	80.3	79.6	**79.9**

TABLE 3.12: NER results of all models without dependency features. Note that DGM-S and DGM are using the gold dependency structures in their models.

	Linear-chain CRFs			Semi-Markov CRFs			DGM-S			DGM		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
ABC	67.8	65.3	66.5	72.2	72.4	**72.3**	69.4	68.8	69.1	71.2	71.5	71.3
CNN	75.0	73.3	74.1	76.7	76.4	**76.6**	76.1	75.2	75.6	76.4	76.0	**76.2**
MNB	77.6	72.3	74.9	76.8	73.3	**75.0**	77.5	70.5	73.8	78.7	73.3	**75.9**
NBC	67.3	63.6	65.4	69.8	68.8	**69.3**	69.3	65.2	67.2	68.8	68.8	**68.8**
P2.5	73.4	68.3	70.8	75.2	72.3	**73.7**	75.7	68.7	72.0	75.3	73.8	**74.6**
PRI	83.6	82.2	82.9	83.9	84.3	84.1	84.3	84.7	84.5	84.8	85.4	**85.1**
VOA	83.2	81.4	82.3	83.5	83.0	83.3	84.7	83.7	**84.2**	84.9	83.7	**84.3**
Overall	77.5	75.1	76.3	78.8	78.1	**78.5**	78.9	77.0	78.0	79.1	78.5	**78.8**

TABLE 3.13: NER Results of all models without dependency features. Note that DGM-S and DGM are using the predicted dependency structures in their models.

3.7 Conclusion

We proposed a novel efficient dependency-guided model for named entity recognition. Motivated by the fact that named entities are typically covered by dependency arcs, presenting internal structures, we built a model that is able to explicitly exploit global structured information conveyed by dependency trees for NER. We showed that the model theoretically is better than the semi-Markov CRFs model in terms of time complexity. Experiments show that our model performs competitively with the semi-Markov CRFs model, even though it requires substantially less running time. Our further research direction in next chapter investigate the structural relations between dependency trees and named entities.

Chapter 4

Dependency-Guided LSTM-CRF for Named Entity Recognition

Our research approach in previous chapter did not make full use of the dependency tree structure and the dependency relation information. How to make good use of the rich relational information as well as complex long-distance interactions among words as conveyed by the complete dependency structures for improved NER remains a research question to be answered.

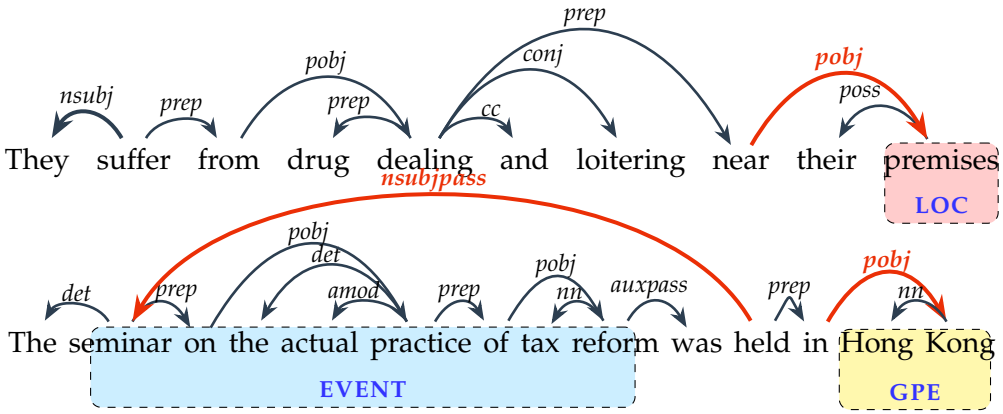


FIGURE 4.1: Example sentences annotated with named entiteis and dependencies in the OntoNotes 5.0 dataset.

The first example in Figure 4.1 illustrates the relationship between a dependency structure and a named entity. Specifically, the word "*premises*", which is a named entity of type LOC (location), is characterized by the incoming arc with label "*pobj*" (prepositional object). This arc reveals a certain level of the semantic role that the word "*premises*" plays in the sentence. Similarly, the two words "*Hong Kong*" in the second example that form an entity of type GPE are also characterized by a similar dependency arc towards them.

The long-distance dependencies capturing non-local structural information can also be very helpful for the NER task (Finkel, Grenager, and Manning, 2005). In the second example of Figure 4.1, the long-distance dependency from "*held*" to "*seminar*" indicates a direct relation "*nsubjpass*" (passive subject) between them, which can be used to

characterize the existence of an entity. However, existing NER models based on linear-chain structures would have difficulties in capturing such long-distance relations (i.e., non-local structures).

One interesting property, as highlighted in the work of Jie, Muis, and Lu (2017), is that most of the entities form subtrees under their corresponding dependency trees. In the example of the EVENT entity in Figure 4.1, the entity itself forms a subtree and the words inside have rich complex dependencies among themselves. Exploiting such dependency edges within the subtrees allows a model to capture non-trivial semantic-level interactions between words within long entities. For example, "*practice*" is the prepositional object (*pobj*) of "*on*" which is a preposition (*prep*) of "*seminar*" in the EVENT entity. Modeling these grandchild dependencies (GD) (Koo and Collins, 2010) requires the model to capture some higher-order long-distance interactions among different words in a sentence.

Inspired by the above characteristics of dependency structures, in this work, we propose a simple yet effective dependency-guided model for NER. Our neural network based model is able to capture both contextual information and rich long-distance interactions between words for the NER task. Through extensive experiments on several datasets on different languages, we demonstrate the effectiveness of our model, which achieves the state-of-the-art performance. To the best of our knowledge, this is the first work that leverages the complete dependency graphs for NER.

4.1 Related Work

NER has been a long-standing task in the field of NLP. While many recent works (Peters et al., 2018a; Akbik, Blythe, and Vollgraf, 2018; Devlin et al., 2019) focus on finding good contextualized word representations for improving NER, our work is mostly related to the literature that focuses on employing dependency trees for improving NER.

Sasano and Kurohashi (2008) exploited the syntactic dependency features for Japanese NER and achieved improved performance with a support vector machine (SVM) (Cortes and Vapnik, 1995) classifier. Similarly, Ling and Weld (2012) included the head word in a dependency edge as features for fine-grained entity recognition. Their approach is a pipeline where they extract the entity mentions with linear-chain conditional random fields (CRF) (Lafferty, McCallum, and Pereira, 2001) and used a classifier to predict the entity type. Liu, Huang, and Zhu (2010) proposed to link the words that are associated with selected typed dependencies (e.g., "*nn*", "*prep*") using a skip-chain CRF (Sutton and McCallum, 2004) model. They showed that some specific relations between the words can be exploited for improved NER. Cucchiarelli and Velardi (2001) applied a dependency parser to obtain the syntactic relations for the purpose of unsupervised NER. The resulting relation information serves as the features for potential existence of named entities. Jie, Muis, and Lu (2017) proposed an efficient dependency-guided model based on the semi-Markov CRF (Sarawagi and Cohen, 2004) for NER. The purpose is to reduce time complexity while maintaining the non-Markovian features. They observed certain relationships between the dependency edges and the named entities. Such relationships are able to define a reduced search space for their model. While these previous approaches do not make full use of the dependency tree structures, we

focus on exploring neural architectures to exploit the complete structural information conveyed by the dependency trees.

4.2 Model

Our dependency-guided model is based on the state-of-the-art BiLSTM-CRF model proposed by Lample et al. (2016). We first briefly present their model as background and next present our dependency-guided model.

4.2.1 Background: BiLSTM-CRF

In the task of named entity recognition, we aim to predict the label sequence $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}$ given the input sentence $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ where n is the number of words. The labels in $\mathbf{y}$ are defined by a label set with the standard IOBES[1] labeling scheme (Ramshaw and Marcus, 1999; Ratinov and Roth, 2009). The CRF (Lafferty, McCallum, and Pereira, 2001) layer defines the probability of the label sequence $\mathbf{y}$ given $\mathbf{x}$:

$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp(score(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp(score(\mathbf{x}, \mathbf{y}'))} \tag{4.1}$$

Following Lample et al. (2016), the score is defined as the sum of transitions and emissions from the bidirectional LSTM (BiLSTM):

$$score(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} F_{\mathbf{x}, y_i} \tag{4.2}$$

where $\mathbf{A}$ is a transition matrix in which $A_{y_i, y_{i+1}}$ is the transition parameter from the label y_i to the label y_{i+1}[2]. $\mathbf{F}_{\mathbf{x}}$ is an emission matrix where $F_{\mathbf{x}, y_i}$ represents the scores of the label y_i at the i-th position. Such scores are provided by the parameterized LSTM (Hochreiter and Schmidhuber, 1997) networks. During training, we minimize the negative log-likelihood to obtain the model parameters including both LSTM and transition parameters.

4.2.2 Dependency-Guided LSTM-CRF

Input Representations The word representation $\mathbf{w}$ in the BiLSTM-CRF (Lample et al., 2016; Ma and Hovy, 2016; Reimers and Gurevych, 2017) model consists of the concatenation of the word embedding as well as the corresponding character-based representation. Inspired by the fact that each word (except the root) in a sentence has exactly one *head* (i.e., *parent*) word in the dependency structure, we can enhance the word representations with such dependency information. Similar to the work by Miwa and Bansal (2016), we concatenate the word representation together with the corresponding head word representation and dependency relation embedding as the input representation. Specifically, given a dependency edge (x_h, x_i, r) with x_h as parent, x_i as child and r as

[1] "S-" indicates the entity with a single word and "E-" indicates the end of an entity.

[2] y_0 and y_{n+1} are `start` and `end` labels.

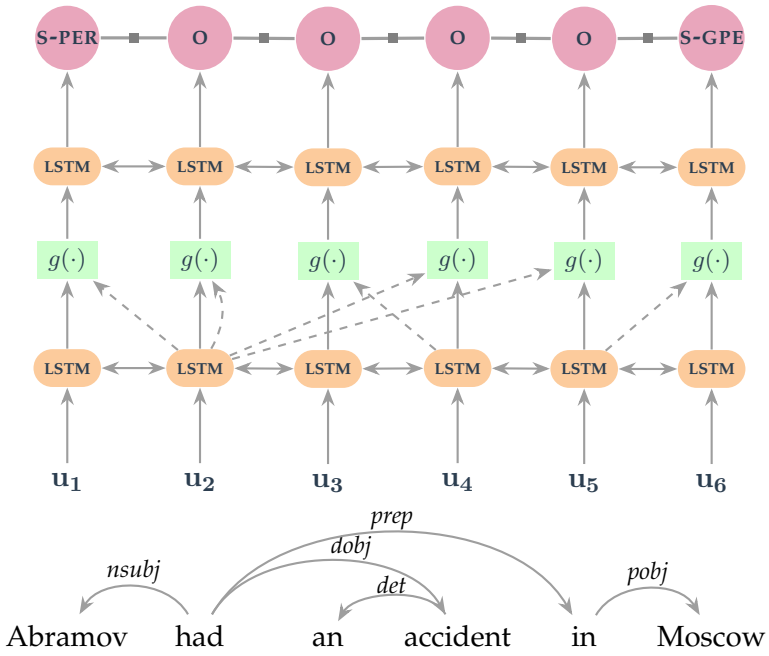


FIGURE 4.2: Dependency-guided LSTM-CRF with 2 LSTM Layers. Dashed connections mimic the dependency edges. "$g(\cdot)$" represents the interaction function.

dependency relation, the representation at position i can be denoted as:

$$\mathbf{u}_i = [\mathbf{w}_i; \mathbf{w}_h; \mathbf{v}_r], \quad x_h = parent(x_i) \tag{4.3}$$

where $\mathbf{w}_i$ and $\mathbf{w}_h$ are the word representations of the word x_i and its parent x_h, respectively. We take the final hidden state of character-level BiLSTM as the character-based representation (Lample et al., 2016). $\mathbf{v}_r$ is the embedding for the dependency relation r. These relation embeddings are randomly initialized and fine-tuned during training. The above representation allows us to capture the direct long-distance interactions at the input layer. For the word that is a root of the dependency tree, we treat its parent as itself[3] and create a root relation embedding. Additionally, contextualized word representations (e.g., ELMo) can also be concatenated into $\mathbf{u}$.

Neural Architecture Given the dependency-encoded input representation $\mathbf{u}$, we apply the LSTM to capture the contextual information and model the interactions between the words and their corresponding parents in the dependency trees. Figure 4.2 shows the proposed dependency-guided LSTM-CRF (**DGLSTM-CRF**) with 2 LSTM layers for the example sentence "*Abramov had an accident in Moscow*" and its dependency structure. The corresponding label sequence is {S-PER, O, O, O, O, S-GPE}. Followed by the first BiLSTM, the hidden states at each position will propagate to the next BiLSTM layer and its child along the dependency trees. For example, the hidden state of the word "*had*", $\mathbf{h}_2^{(1)}$, will propagate to its child "*Abramov*" at the first position. For

[3]We also tried using a root word embedding but the performance is similar.

the word that is root, the hidden state at that specific position will propagate to itself. We use an *interaction function* $g(\mathbf{h}_i, \mathbf{h}_{p_i})$ to capture the interaction between the child and its parent in a dependency. Such an interaction function can be concatenation, addition or a multilayer perceptron (MLP). We further apply another BiLSTM layer on top of the interaction functions to produce the context representation for the final CRF layer.

The architecture shown in Figure 4.2 with a 2-layer BiLSTM can effectively encode the grandchild dependencies because the input representations encode the parent information and the interaction function further propagate the grandparent information. Such propagations allow the model to capture the indirect long-distance interactions from the grandchild dependencies between the words in the sentence as mentioned in Section 6.1. In general, we can stack more interaction functions and BiLSTMs to enable deeper reasoning over the dependency trees. Specifically, the hidden states of the $(l+1)$-th layer $\mathbf{H}^{(l+1)}$ can be calculated from the hidden state of the previous layer $\mathbf{H}^{(l)}$:

$$\begin{aligned}
\mathbf{H}^{(l+1)} &= \text{BiLSTM}\Big(f\big(\mathbf{H}^{(l)}\big)\Big) \\
\mathbf{H}^{(l)} &= \Big[\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \cdots, \mathbf{h}_n^{(l)}\Big] \\
f\big(\mathbf{H}^{(l)}\big) &= \Big[g(\mathbf{h}_1^{(l)}, \mathbf{h}_{p_1}^{(l)}), \cdots, g(\mathbf{h}_n^{(l)}, \mathbf{h}_{p_n}^{(l)})\Big]
\end{aligned}$$

where p_i indicates the parent index of the word x_i. $g(\mathbf{h}_i^{(l)}, \mathbf{h}_{p_i}^{(l)})$ represents the interaction functions between the hidden state at the i-th and p_i-th positions under the dependency edges (x_{p_i}, x_i). The number of layers L can be chosen according to the performance on the development set.

Interaction Function The interaction function between the parent and child representations can be defined in various ways. Table 4.1 shows the list of interaction function considered in our experiments. The first one returns the hidden state itself, which is equivalent to stacking the LSTM layers. The concatenation and addition involve no

Interaction Function	$g(\mathbf{h}_i, \mathbf{h}_{p_i})$
Self connection	$\mathbf{h}_i$
Concatenation	$\mathbf{h}_i \bigoplus \mathbf{h}_{p_i}$
Addition	$\mathbf{h}_i + \mathbf{h}_{p_i}$
MLP	$\text{ReLU}\big(\mathbf{W}_1\mathbf{h}_i + \mathbf{W}_2\mathbf{h}_{p_i}\big)$

TABLE 4.1: List of interaction functions.

parameter, which are straightforward ways to model the interactions. The last one applies an MLP to model the interaction between parent and child representations. With the rectified linear unit (ReLU) as activation function, the $g(\mathbf{h}_i, \mathbf{h}_{p_i})$ function is analogous to a graph convolutional network (GCN) (Kipf and Welling, 2017) formulation. In such a graph, each node has a self connection (i.e., $\mathbf{h}_i$) and a dependency connection with parent (i.e., $\mathbf{h}_{p_i}$). Similar to the work by Marcheggiani and Titov (2017), we adopt different parameters $\mathbf{W}_1$ and $\mathbf{W}_2$ for self and dependency connections.

Dataset	Train		Dev		Test		ST	GD
	# Sent.	# Entity	# Sent.	# Entity	# Sent.	# Entity	(%)	(%)
OntoNotes 5.0 - English	59,924	81,828	8,528	11,066	8,262	11,057	98.5	41.1
OntoNotes 5.0 - Chinese	36,487	62,543	6,083	9,104	4,472	7,494	92.9	49.1
SemEval2010T1 - Catalan	8,709	15,278	1,445	2,431	1,698	2,910	100.0	28.6
SemEval2010T1 - Spanish	9,022	17,297	1,419	2,615	1,705	3,046	100.0	29.8

TABLE 4.2: Dataset statistics. "ST" is the ratio of entities that form subtrees. "GD" is the ratio of entities that have grandchild dependencies within their subtrees.

4.3 Experiments

Datasets The main experiments are conducted on the large-scale OntoNotes 5.0 (Weischedel et al., 2013) English and Chinese datasets. We chose these datasets because they contain both constituency tree and named entity annotations. There are 18 types of entities defined in the OntoNotes dataset. We convert the constituency trees into the Stanford dependency (De Marneffe and Manning, 2008) trees using the rule-based tool (De Marneffe, MacCartney, and Manning, 2006) by Stanford CoreNLP (Manning et al., 2014). For English, Pradhan et al. (2013) provided the train/dev/test split[4] and the split has been used by several previous works (Chiu and Nichols, 2016; Li et al., 2017; Ghaddar and Langlais, 2018). For Chinese, we use the official splits provided by Pradhan et al. (2012)[5].

Besides, we also conduct experiments on the Catalan and Spanish datasets from the SemEval-2010 Task 1[6] (Recasens et al., 2010)[7]. The SemEval-2010 task was originally designed for the task of coreference resolution in multiple languages. Again, we chose these corpora primarily because they contain both dependency and named entity annotations. Following Finkel and Manning (2009) and Jie, Muis, and Lu (2017), we select the most dominant three entity types and merge the rest into one general a entity type "*misc*". Table 4.2 shows the statistics of the datasets used in main experiments. To further evaluate the effectiveness of the dependency structures, we also conduct additional experiments under a low-resource setting for NER (Cotterell and Duh, 2017).

The last two columns of Table 4.2 show the relationships between the dependency trees and named entities with length larger than 2 for the complete dataset. Specifically, the penultimate column shows the percentage of entities that can form a complete subtree (ST) under their dependency tree structures. Apparently, most of the entities form subtrees, especially for the Catalan and Spanish datasets where 100% entities form subtrees. This observation is consistent with the findings reported in Jie, Muis, and Lu (2017). The last column in Table 4.2 shows the percentage of the grandchild dependencies (Koo and Collins, 2010) (GD) that exist in these subtrees (i.e., entities). Such grandchild dependencies could be useful for detecting certain named entities, especially for

[4] http://cemantix.org/data/ontonotes.html
[5] http://conll.cemantix.org/2012/data.html
[6] http://stel.ub.edu/semeval2010-coref/download
[7] This dataset also has English portion but it is a subset of the OntoNotes English.

long entities. As we will see later in Section 4.4, the performance of long entities can be significantly improved with our dependency-guide model.

The heatmap table in Figure 4.3 shows the correlation between the entity types and the dependency relations in the OntoNotes English dataset. Specifically, each entry denotes the percentage of the entities that have a parent dependency with a specific dependency relation. For example, at the row with GPE entity, 37% of the entity words[8] have a dependency edge whose label is "*pobj*". When looking at column of "*pobj*" and "*nn*", we can see that most of the entities relate to the prepositional object (*pobj*) and noun compound modifier (*nn*) dependencies. Especially for the NORP (i.e., *nationalities or religious or political groups*) and ORDINAL (e.g., "*first*", "*second*") entities, more than 60% of the entity words have the dependency with adjectival modifier (*amod*) relation. Furthermore, every entity type (i.e., row) has a most related dependency relation (with more than 17% occurrences). Such observations present useful information that can be used to categorize named entities with different types.

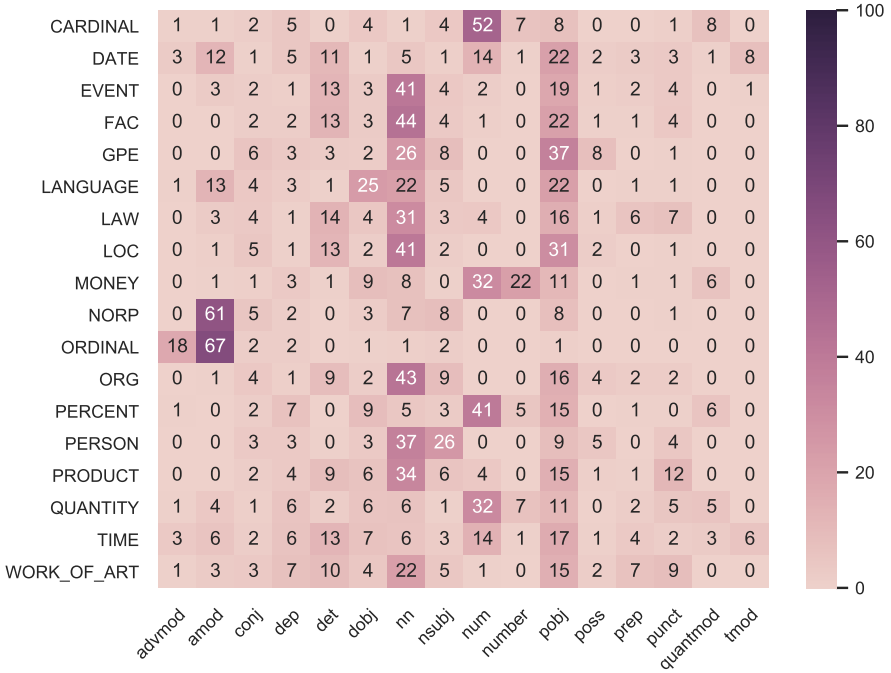


FIGURE 4.3: Percentage of entity words (y axis) with respect to dependency relations (x axis) in the OntoNotes English dataset. Columns with percentage less than 5% are ignored for brevity.

Baselines We implemented the state-of-the-art NER model **BiLSTM-CRF** (Lample et al., 2016) as the first baseline with different number of LSTM layers ($L = \{0, 1, 2, 3\}$). $L = 0$ indicates the model only relies on the input representation. Following Zhang, Qi, and Manning (2018), the complete dependency trees are considered bidirectional and encoded with a contextualized GCN (BiLSTM-GCN). We further add the relation-specific parameters (Marcheggiani and Titov, 2017) and a CRF layer for the NER task.

[8]The words that are annotated with entity labels.

The resulting baseline is **BiLSTM-GCN-CRF**. We use the bootstrapping paired t-test (Berg-Kirkpatrick, Burkett, and Klein, 2012) for significance test when comparing the results of different models.

BiLSTM-GCN-CRF Figure 4.4 shows the neural architecture for the BiLSTM-GCN-CRF model. Following Zhang, Qi, and Manning (2018), the input representation at each position $\mathbf{w}_i$ is the word representation which consists of the pre-trained word embeddings and its character representation. To capture contextual information, we stack a BiLSTM layer before the GCN. Secondly, the GCN captures the dependency tree structure as shown in Figure 4.4. Following Zhang, Qi, and Manning (2018), we treat the dependency trees as undirected and build a symmetric adjacency matrix during the GCN update:

$$\mathbf{h}_i^{(l)} = \text{ReLU}\Big(\sum_{j=1}^{n} A_{ij}\mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)} + \mathbf{b}^{(l)}\Big) \tag{4.4}$$

where $\mathbf{A}$ is the adjacency matrix. $A_{ij} = 1$ indicates there is a dependency edge between the i-th word and the j-th word[9]. $\mathbf{h}_i^{(l)}$ is the hidden state at the i-th position in the l-th layer. We can stack J layers of GCN in the model. In our experiments, we set the number of GCN layers $J = 1$ as we did not observe significant improvements by increasing J. In fact, we might obtain harmful performance for a larger J as deeper GCN layers will diminish the effect of the contextual information, which is important for the task of NER.

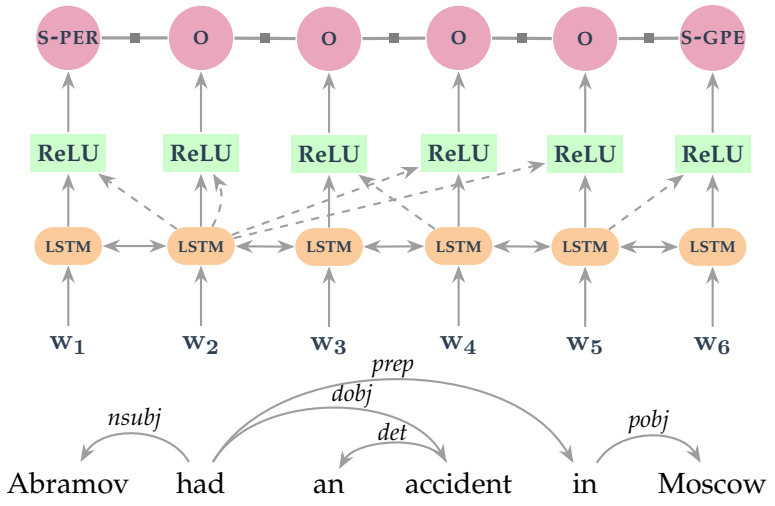


FIGURE 4.4: BiLSTM-GCN-CRF. Dashed connections mimic the dependency edges.

[9] $A_{ij} = A_{ji}$ for symmetric matrix.

However, Equation 4.4 does not include the dependency relation information. We modify the Equation 4.4 and include the dependency relation parameter[10]:

$$\mathbf{h}_i^{(l)} = \sigma\Big(\sum_{j=1}^{n} A_{ij}\big(\mathbf{W}_1^{(l)}\mathbf{h}_j^{(l-1)} + \mathbf{W}_2^{(l)}\mathbf{h}_j^{(l-1)} w_{r_{ij}}\big)\Big)$$

where $w_{r_{ij}}$ is the dependency relation weight that parameterize the dependency relation r between the i-th word and the j-th word. Such formulation uses the relation to weight the adjacent hidden states in the dependencies.

Experimental Setup We choose MLP as the interaction function in our DGLSTM-CRF according to performance on the development set. The hidden size of all models (i.e., LSTM, GCN) is set to 200. We use the Glove (Pennington, Socher, and Manning, 2014) 100-d word embeddings, which was shown to be effective in English NER task (Ma and Hovy, 2016; Peters et al., 2018a). We use the publicly available FastText (Grave et al., 2018) word embeddings for Chinese, Catalan and Spanish. The ELMo (Peters et al., 2018a), deep contextualized word representations[11] are used for all languages in our experiments since Che et al. (2018) provides ELMo for many other languages[12], including Chinese, Catalan and Spanish. We use the average weights over all layers of the ELMo representations and concatenate them with the input representation $\mathbf{u}$. Our models are optimized by mini-batch stochastic gradient descent (SGD) with learning rate 0.01 and batch size 10. The L_2 regularization parameter is 1e-8. The hyperparameters are selected according to the performance on the OntoNotes English development set.

We implemented all the models with PyTorch (Paszke et al., 2017). For both BiLSTM-CRF and DGLSTM-CRF model, we train them on all datasets with 100 epochs and take the model that perform the best on the development set. For BiLSTM-GCN-CRF, we train for 300 epochs with a clipping rate of 3.

4.3.1 Main Results

OntoNotes English Table 4.3 shows the performance comparison between our work and previous work on the OntoNotes English dataset. Without the LSTM layers (i.e., $L = 0$), the proposed model with dependency information significantly improves the NER performance with more than 2 points in F_1 compared to the baseline BiLSTM-CRF ($L = 0$), which demonstrate the effectiveness of dependencies for the NER task. Our best performing BiLSTM-CRF baseline (with Glove) achieves a F_1 score of 87.78 which is better than or on par with previous works (Chiu and Nichols, 2016; Li et al., 2017; Ghaddar and Langlais, 2018) with extra features. This baseline also outperforms the CNN-based models (Strubell et al., 2017; Li et al., 2017). The BiLSTM-GCN-CRF model outperforms the BiLSTM-CRF model but achieves inferior performance compared to the proposed DGLSTM-CRF model. We believe it is challenging to preserve

[10]The bias vector is ignore for brevity.

[11]We also tried BERT (Devlin et al., 2019) in preliminary experiments and obtained similar performance as ELMo. The NER performance using BERT without fine-tuning reported in Peters, Ruder, and Smith (2019) is consistent with the one reported by ELMo (Peters et al., 2018a).

[12]https://github.com/HIT-SCIR/ELMoForManyLangs

the surrounding context information with stacking GCN layers while contextual information is important for NER (Peters et al., 2018b). Overall, the 2-layer DGLSTM-CRF model significantly (with $p < 0.01$) outperforms the best BiLSTM-CRF baseline and the BiLSTM-GCN-CRF model. As we can see from the table, increasing the number of layers (e.g., $L = 3$) does not give us further improvements for both BiLSTM-CRF and DGLSTM-CRF because such third-order information (e.g., the relationship among a word's parent, its grandparent, and great-grandparent) does not play an important role in indicating the presence of named entities.

Model	**Prec.**	**Rec.**	**F_1**
Chiu and Nichols (2016)	86.04	86.53	86.28
Li et al. (2017)	88.00	86.50	87.21
Ghaddar and Langlais (2018)	-	-	87.95
Strubell et al. (2017)	-	-	86.84
BiLSTM-CRF ($L = 0$)	82.03	80.78	81.40
BiLSTM-CRF ($L = 1$)	87.21	86.93	87.07
BiLSTM-CRF ($L = 2$)	87.89	87.68	87.78
BiLSTM-CRF ($L = 3$)	87.81	87.50	87.65
BiLSTM-GCN-CRF	88.30	88.06	88.18
DGLSTM-CRF ($L = 0$)	85.31	82.19	84.09
DGLSTM-CRF ($L = 1$)	**88.78**	87.29	88.03
DGLSTM-CRF ($L = 2$)	88.53	**88.50**	**88.52**
DGLSTM-CRF ($L = 3$)	87.59	88.93	88.25
Contextualized Word Representation			
Akbik, Blythe, and Vollgraf (2018) (Flair)	-	-	89.30
BiLSTM-CRF ($L = 0$) + ELMo	85.44	84.41	84.92
BiLSTM-CRF ($L = 1$) + ELMo	89.14	88.59	88.87
BiLSTM-CRF ($L = 2$) + ELMo	88.25	89.71	88.98
BiLSTM-CRF ($L = 3$) + ELMo	88.03	89.04	88.53
BiLSTM-GCN-CRF + ELMo	89.40	89.71	89.55
DGLSTM-CRF ($L = 0$) + ELMo	86.87	85.12	85.99
DGLSTM-CRF ($L = 1$) + ELMo	89.40	89.96	89.68
DGLSTM-CRF ($L = 2$) + ELMo	**89.59**	**90.17**	**89.88**
DGLSTM-CRF ($L = 3$) + ELMo	89.43	90.15	89.79

TABLE 4.3: Performance comparison on the OntoNotes 5.0 English dataset.

We further compare the performance of all models with ELMo (Peters et al., 2018a) representations to investigate whether the effect of dependency would be diminished by the contextualized word representations. With $L = 0$, the ELMo representations largely improve the performance of BiLSTM-CRF compared to the BiLSTM-CRF model with word embeddings only but is still 1 point below our DGLSTM-CRF model. The 2-layer DGLSTM-CRF model outperforms the best BilSTM-CRF baseline with 0.9 points in F_1 ($p < 0.001$). Empirically, we found that among the entities that are correctly

predicted by DGLSTM-CRF but wrongly predicted by BiLSTM-CRF, 47% of them are with length more than 2. Our finding shows the 2-layer DGLSTM-CRF model is able to accurately recognize long entities, which can lead to a higher precision. In addition, 51.9% of the entities that are correctly retrieved by DGLSTM-CRF have the dependency relations "*pobj*", "*nn*" and "*nsubj*", which have strong correlations with certain named entity types (Figure 4.3). Such a result demonstrates the effectiveness of dependency relations in improving the recall of NER.

Model	**Prec.**	**Rec.**	**F_1**
Pradhan et al. (2013)	78.20	66.45	71.85
Lattice LSTM (Z&Y, 2018)	76.34	77.01	76.67
BiLSTM-CRF ($L = 0$)	76.67	67.79	71.95
BiLSTM-CRF ($L = 1$)	**78.45**	74.59	76.47
BiLSTM-CRF ($L = 2$)	77.94	75.33	76.61
BiLSTM-CRF ($L = 3$)	76.17	75.23	75.70
BiLSTM-GCN-CRF	76.35	75.89	76.12
DGLSTM-CRF ($L = 0$)	76.91	70.65	73.65
DGLSTM-CRF ($L = 1$)	77.79	75.29	76.52
DGLSTM-CRF ($L = 2$)	77.40	**77.41**	**77.40**
DGLSTM-CRF ($L = 3$)	77.01	74.90	75.94
Contextualized Word Representation			
BiLSTM-CRF ($L = 0$) + ELMo	75.20	73.39	74.28
BiLSTM-CRF ($L = 1$) + ELMo	**79.20**	79.21	79.20
BiLSTM-CRF($L = 2$) + ELMo	78.49	79.44	78.96
BiLSTM-CRF ($L = 3$) + ELMo	78.54	79.76	79.14
BiLSTM-GCN-CRF + ELMo	78.71	79.29	79.00
DGLSTM-CRF ($L = 0$) + ELMo	76.27	74.61	75.43
DGLSTM-CRF ($L = 1$) + ELMo	78.91	80.22	79.56
DGLSTM-CRF ($L = 2$) + ELMo	78.86	**81.00**	**79.92**
DGLSTM-CRF ($L = 3$) + ELMo	79.30	79.86	79.58

TABLE 4.4: Performance comparison on the OntoNotes 5.0 Chinese Dataset.

OntoNotes Chinese Table 4.4 shows the performance comparison on the Chinese datasets. We compare our models against the state-of-the-art NER model on this dataset, Lattice LSTM (Zhang and Yang, 2018)[13]. Our implementation of the strong BiLSTM-CRF baseline achieves comparable performance against the Lattice LSTM. Similar to the English dataset, our model with $L = 0$ significantly improves the performance compared to the BiLSTM-CRF ($L = 0$) model. Our DGLSTM-CRF model achieves the best performance with $L = 2$ and is consistently better ($p < 0.02$) than the strong BiLSTM-CRF baselines. As we can see from the table, the improvements of the DGLSTM-CRF model mainly come from recall ($p < 0.001$) compared to the BiLSTM model, especially

[13]We run their code on the OntoNotes 5.0 Chinese dataset.

Model	Catalan			Spanish		
	Prec.	**Rec.**	**F_1**	**Prec.**	**Rec.**	**F_1**
BiLSTM-CRF ($L = 0$)	65.91	49.90	56.80	65.97	52.63	58.55
BiLSTM-CRF ($L = 1$)	76.83	63.47	69.51	78.33	69.89	73.87
BiLSTM-CRF ($L = 2$)	73.79	67.63	70.58	77.73	70.91	74.16
BiLSTM-CRF ($L = 3$)	74.75	67.35	70.86	76.41	72.95	74.64
BiLSTM-GCN-CRF	81.25	75.22	78.12	84.10	79.88	81.93
DGLSTM-CRF ($L = 0$)	73.42	61.79	67.10	74.90	61.21	67.38
DGLSTM-CRF ($L = 1$)	81.87	79.28	80.55	83.21	81.19	82.19
DGLSTM-CRF ($L = 2$)	**83.35**	**80.00**	**81.64**	84.05	82.90	83.47
DGLSTM-CRF ($L = 3$)	81.87	80.21	81.03	**84.12**	**83.45**	**83.78**
Contextualized Word Representation						
BiLSTM-CRF ($L = 0$) + ELMo	67.53	64.47	65.96	73.16	69.01	71.03
BiLSTM-CRF ($L = 1$) + ELMo	77.85	76.22	77.03	81.72	79.09	80.38
BiLSTM-CRF($L = 2$) + ELMo	78.61	78.32	78.46	80.89	80.30	80.59
BiLSTM-CRF($L = 3$) + ELMo	79.11	77.32	78.21	80.48	79.45	79.96
BiLSTM-GCN-CRF + ELMo	83.68	83.16	83.42	85.31	85.19	85.25
DGLSTM-CRF ($L = 0$) + ELMo	70.87	65.81	68.25	75.96	72.52	74.20
DGLSTM-CRF ($L = 1$) + ELMo	82.29	82.37	82.33	84.05	84.77	84.41
DGLSTM-CRF ($L = 2$) + ELMo	**84.71**	83.75	**84.22**	**87.79**	**87.33**	**87.56**
DGLSTM-CRF ($L = 3$) + ELMo	84.50	**83.92**	**84.21**	86.74	86.57	86.66

TABLE 4.5: Results on the SemEval-2010 Task 1 datasets.

in the scenario with word embeddings only. Empirically, we also found that those correctly retrieved entities of the DGLSTM-CRF (compared against the baseline) mostly correlate with the following dependency relations: "*nn*", "*nsubj*", "*nummod*". However, DGLSTM-CRF achieves lower precisions against BiLSTM-CRF, which indicates that the DGLSTM-CRF model makes more false-positive predictions. The reason could be the relatively lower ratio of ST(%)[14] as shown in Table 4.2, which means some of the entities do not form subtrees under the complete dependency trees. In such a scenario, the model may not correctly identify the boundary of the entities, which results in lower precision.

SemEval-2010 Table 4.5 shows the results of our models on the SemEval-2010 Task 1 datasets. Overall, we observe substantial improvements of the DGLSTM-CRF on the Catalan and Spanish datasets (with $p < 0.001$ marked in bold against the best performing BiLSTM-CRF baseline), especially for DGLSTM-CRF with ELMo and L larger than 1. With word embeddings, the best DGLSTM-CRF model outperforms the best performing BiLSTM-CRF baseline with more than 10 and 9 points in F_1 on the Catalan and Spanish datasets, respectively. The BiLSTM-GCN-CRF model also performs much better than the BiLSTM-CRF baselines but is worse than the DGLSTM-CRF model with $L \geq 2$. Both precision and recall significantly improve with a large margin compared

[14]Percentage of entities that can form a subtree.

to the best performing BiLSTM-CRF, especially for the recall (with more than 10 points improvement) on these two datasets. With ELMo, the best performing DGLSTM-CRF model outperforms the BiLSTM-CRF baseline with about 6 and 7 points in F_1 on these two datasets, respectively. The substantial improvements show that the structural dependency information is extremely helpful for these two datasets.

With ELMo representations, we observe about 2 and 3 points improvements in F_1 compared with the 1-layer DGLSTM-CRF model on these two datasets, respectively. Empirically, more than 50% of the entities that are correctly predicted by the 2-layer model but not the 1-layer model are with length larger than 2. Also, most of these entities contain the grandchild dependencies "(*sn*, *sn*)" and "(*spec*, *sn*)" where *sn* represents noun phrase and *spec* represents specifier (e.g., determiner, quantifier) in both datasets. Such a finding shows that the 2-layer model is able to capture the interactions given by the grandchild dependencies.

4.3.2 Additional Experiments

CoNLL-2003 English Table 4.6 shows the performance on the CoNLL-2003 English dataset. The dependencies are predicted from Spacy (Honnibal and Montani, 2017). With the contextualized word representations, DGLSTM-CRF outperforms BiLSTM-CRF with 0.2 points in F_1 ($p < 0.09$). The improvement is not significant due to the relatively lower equality of the dependency trees. To further study the effect of the dependencies, we modified the predicted dependencies to ensure each entity form a subtree in the complete dataset. Such modification improves the F_1 to 92.7, which is significantly better ($p < 0.05$) than the BiLSTM-CRF.

Model	**Prec.**	**Rec.**	**F_1**
Peters et al. (2018a) ELMo	-	-	92.2
BiLSTM-CRF + ELMo ($L = 2$)	92.1	92.3	92.2
DGLSTM-CRF + ELMo ($L = 2$)	92.2	92.5	92.4

TABLE 4.6: Performance on the CoNLL-2003 English dataset.

Low-Resource NER Following Cotterell and Duh (2017), we emulate truly low-resource condition with 100 sentences for training. We assume that the contextualized word representations are not available and dependencies are predicted. Table 4.7 shows the NER performance on the SemEval-2010 Task 1 datasets under the low-resource setting. With limited amount of training data, BiLSTM-CRF suffers from low recall and the DGLSTM-CRF largely improves it on these two datasets. Using gold dependencies further significantly improves the precision and recall.

Effect of Dependency Quality To evaluate how the quality of dependency trees affect the performance, we train a state-of-the-art dependency parser (Dozat and Manning, 2017) using our training set and make prediction on the development/test set. We train a BERT-based (Devlin et al., 2019) dependency parser (Dozat and Manning, 2017) using the training set for each of four languages. Specifically, we adopt the

Model	Catalan			Spanish		
	Prec.	**Rec.**	**F_1**	**Prec.**	**Rec.**	**F_1**
BiLSTM-CRF ($L = 1$)	47.88	18.59	26.78	40.77	19.01	25.93
DGLSTM-CRF ($L = 1$)	47.71	31.55	37.98	49.39	31.91	38.77
– with gold dependency	**52.13**	**33.26**	**40.61**	**52.14**	**35.59**	**42.30**

TABLE 4.7: Low-resource NER performance on the SemEval-2010 Task 1 datasets.

`bert-base-uncased` model for English, `bert-base-multilingual-cased` for Catalan and Spanish and `bert-base-chinese` for Chinese. Because the Chinese BERT model is based on characters but not Chinese words which are segmented. We further incorporate a span extractor layer right after BERT encoder for Chinese. We following Lee et al. (2017) to design the span extractor layer. Our code for dependency parser is available at https://github.com/allanj/bidaf_dependency_parsing We implemented the dependency parser using the AllenNLP package (Gardner et al., 2017). Table 4.8 shows the performance (LAS) of the dependency parser on four languages (i.e., OntoNotes English, OntoNotes Chinese, Catalan and Spanish) and the performance of DGLSTM-CRF against the best performing BiLSTM-CRF with ELMo. DGLSTM-CRF even with predicted dependencies is able to consistently outperform the BiLSTM-CRF on four languages. However, the performance is still worse than the DGLSTM-CRF with gold dependencies, especially on the Catalan and Spanish. Such results suggest that it is essential to have high-quality dependency annotations available for the proposed model.

	English	**Chinese**	**Catalan**	**Spanish**
BiLSTM-CRF	88.98	79.20	78.46	80.59
(Dependency LAS)†	(94.89)	(89.28)	(93.25)	(93.35)
DGLSTM-CRF (Predicted)	**89.64**	**79.59**	**82.37**	**83.92**
Improvement Δ	+0.66	+0.39	+3.91	+3.33
DGLSTM-CRF (Gold)	89.88	79.92	84.22	87.56

TABLE 4.8: F_1 performance of DGLSTM-CRF with predicted dependencies against the best performing BiLSTM-CRF. †: LAS is label attachment score which is the metric for dependency evaluation.

Ablation Study Table 4.9 shows the ablation study of the 2-layer DGLSTM-CRF model on the OntoNotes English dataset. With self connection as interaction function, the F_1 drops 0.3 points. The model achieves comparable performance with concatenation as interaction function but F_1 drops about 0.4 points with the addition interaction function. We believe that the addition potentially leads to certain information loss. Without the dependency relation embedding $\mathbf{v}_r$ in the input representation, the F_1 drops about 0.4 points.

Model	Prec.	Rec.	F_1
BiLSTM-CRF + ELMo ($L = 2$)	89.14	88.59	88.87
DGLSTM-CRF + ELMo ($L = 2$)	**89.59**	**90.17**	**89.88**
$-g(\cdot) =$ self connection	89.17	90.08	89.62
$-g(\cdot) =$ Concatenation	89.43	90.09	89.76
$-g(\cdot) =$ Addition	89.24	89.78	89.50
–w/o dependency relation	88.92	89.99	89.46

TABLE 4.9: Ablation study of the DGLSTM-CRF model on the OntoNotes English dataset.

4.4 Analysis

4.4.1 Effectiveness of Dependency Relations

To demonstrate whether the model benefits from the dependency relations, we first select the entities that are correctly predicted by the 2-layer DGLSTM-CRF model but not by the best performing baseline 2-layer BiLSTM-CRF on the OntoNotes English dataset. We draw the heatmap in Figure 4.5 based on these entities. Comparing Figure 4.3 and 4.5, we can see that they are similar in terms of the density. Both of them show consistent relationships between the entity types and the dependency relations. The comparison shows that the improvements partially result from the effect of dependency relations. We also found from our model's predictions that some entity types have strong correlations with the relation pairs on grandchild dependencies.

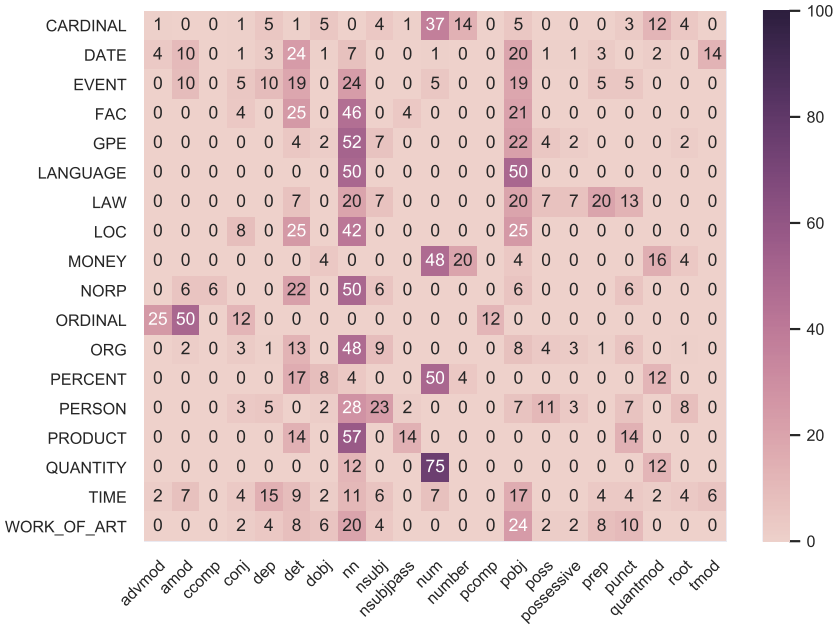


FIGURE 4.5: Correlations between the correctly predicted entities and the dependency relations.

4.4.2 Entity with Different Lengths

Table 4.10 shows the performance comparison with different entity lengths on all datasets. As mentioned earlier, the dependencies as well as the grandchild relations allow our models to capture the long-distance interactions between the words. As shown in the table, the performance of entities with lengths more than 1 consistently improves with DGLSTM-CRF for all languages except Chinese. As we pointed out in the dataset statistics (Table 4.2), the number of entities that form subtrees in OntoNotes Chinese is relatively smaller compared to other datasets. The performance gain is more significant for entities with longer length on the other three languages. We found that, among the improvements of entities with length larger than 2 in English, 85% of them have long-distance dependencies and 30% of them have grandchild dependencies within the entity boundary. The analysis shows that our model that exploits the dependency tree structures is helpful for recognizing long entities.

Dataset	**Model**	**Entity Length**					
		1	**2**	**3**	**4**	**5**	**≥6**
English	BiLSTM-CRF	**91.8**	88.5	83.4	84.0	75.4	76.0
	DGLSTM-CRF	**91.8**	**90.1**	**85.4**	**87.0**	**80.8**	**78.7**
Chinese	BiLSTM-CRF	81.2	74.3	**73.1**	62.8	**70.3**	**57.5**
	DGLSTM-CRF	**82.2**	**75.5**	71.8	**64.1**	58.5	41.1
Catalan	BiLSTM-CRF	80.5	81.0	75.8	56.1	45.0	38.4
	DGLSTM-CRF	**85.4**	**85.1**	**84.1**	**78.9**	**60.9**	**59.3**
Spanish	BiLSTM-CRF	84.2	81.1	81.0	53.3	53.3	37.1
	DGLSTM-CRF	**89.3**	**87.4**	**90.8**	**74.1**	**67.7**	**64.4**

TABLE 4.10: Performance of entities with different lengths on the four datasets: OntoNotes (English), OntoNotes Chinese, Catalan and Spanish.

4.5 Relation Pairs on Grandchild Dependencies

Figure 4.6 visualized the correlations between the entities and the grandchild dependency relation pairs on the OntoNotes English dataset. As we can see from the figure, most of these entities correlate to the "(*nn*, *nn*)" and "(*nn*, *pobj*)" relation pairs on the grandchild dependencies. Such correlations also show that the relation pair information on the grandchild dependencies can be helpful for detecting certain entities.

4.6 Conclusions

Motivated by the relationships between the dependency trees and named entities, we propose a dependency-guided LSTM-CRF model to encode the complete dependency tree and capture such relationships for the NER task. Through extensive experiments

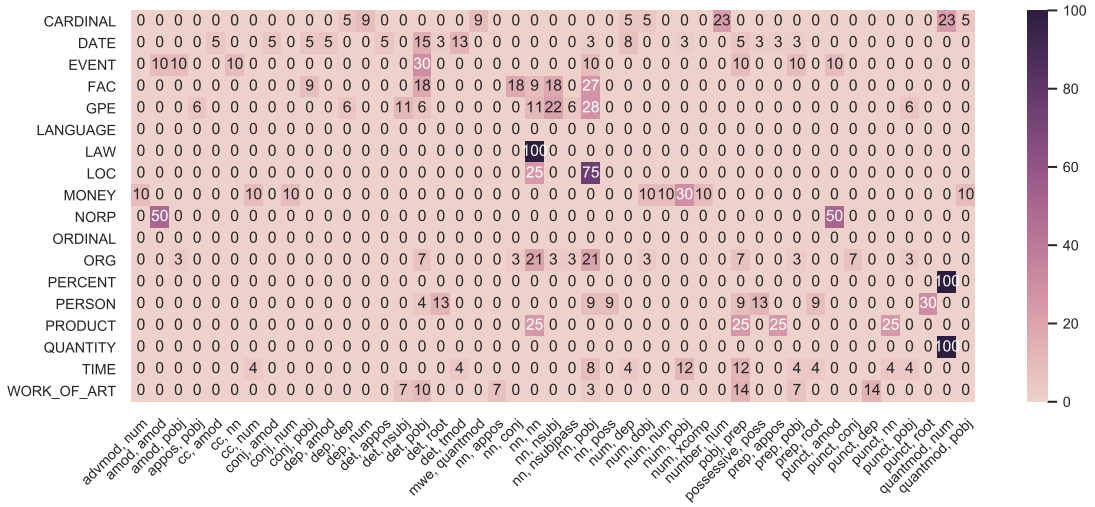


FIGURE 4.6: Correlations between the entity types and the dependency relation pairs on the grandchild dependencies.

on several datasets, we demonstrate the effectiveness of the proposed model in improving the NER performance. Such improvements also generalize to a low-resource setting. Our analysis shows that NER benefits from the dependency relations and long-distance dependencies, which are able to capture the non-local interactions between the words.

As statistics shows that most of the entities form subtrees under the dependency trees, future work includes building a model for joint NER and dependency parsing which regards each entity as a single unit in a dependency tree.

4.7 Learning Latent Dependency for Named Entity Recognition (Future Work)

Dependency tree structures have been proven to have a strong relationship with named entities (Jie and Lu, 2019). The underlying properties (e.g., dependency relations correlate to entity types) are evidently helpful for the named entity recognition (NER) task. However, creating dependency annotations for NER datasets could be expensive and laborious, especially in low-resource languages. We discuss a possible solution where we treat the dependency trees as latent variables for named entity recognition.

4.7.1 Dependency Trees as Latent Variables

In this problem, dependency tree serves as a structured latent variable (Martins et al., 2019) during the inference process as shown in Figure 4.7. Equation 4.5 shows the marginalization over all the trees $\mathbf{h}$. As enumerating all the possible dependency trees $\mathbf{h}$ is intractable, we need some approximations for tractable inference.

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{y}, \mathbf{h}|\mathbf{x}) \tag{4.5}$$

For example, Corro and Titov (2019) proposed a sampling mechanism to sample dependency trees from the tree distribution. They apply the reparameterization trick to make the process differentiable, which in turns make the complete model differentiable. There are some other literature working on the same problem: making the latent variable differentiable (e.g., straight-through estimator (Bengio, Léonard, and Courville, 2013)). Thus, as the intermediate step of our dependency-guided LSTM-CRF model, we aim to apply these techniques for further attempt where we do not require the dependency trees to be available.

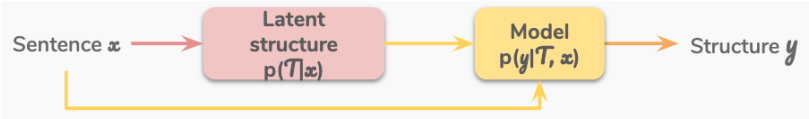


FIGURE 4.7: Dependency trees as latent variables for our NER task.

4.7.2 Multi-task Learning

In order to further improve the performance, we are able to jointly train the dependency dataset where we can have the joint loss from dependency and named entity performance. The nice thing is that we do not require a dataset to have both named entity and dependency annotations. We can train the complete module with separate dependency and named entity datasets.

4.8 Connections between the DGLSTM-CRF and DGM in Chapter 3

We discuss the connections between the DGM in Chapter 3 and the DGLSTM-CRF model in this section.

Dependency-Guided Model based on Semi-Markov CRF The dependency-guided model (DGM) (Jie, Muis, and Lu, 2017) is our first proposed model, which was motivated by the evidence that entities often form subtrees in the dependency trees. It is a model based on the semi-Markov CRF (Sarawagi and Cohen, 2004). In short, the DGM model achieves comparable or better performance with much faster inference speed compared with the semi-Markov CRF.

Dependency-Guided LSTM-CRF The Dependency-Guided LSTM-CRF (DGLSTM-CRF) is proposed based on the motivation to fully encode the dependency tree structures including the dependency relations. Thus, we focus on the design for a better neural architecture to capture the knowledge conveyed from the dependency trees.

Connections Although we applied the dependency information in both DGM and DGLSTM-CRF models, there are some differences where the former is trying to improve the inference speed by refining the search space and the latter is trying to improve the model performance by designing a better neural architecture. Thus, they

are focusing on different aspects and it is definitely possible to integrate these two together. We can apply the DGLSTM to the DGM model which is based on the semi-Markov CRF. However, we actually observe that the performance of the neural semi-Markov model (Ye and Ling, 2018) is actually comparable with the neural CRF (Lample et al., 2016). Hence, it might not be necessary to implement a semi-Markov variant of DGLSTM.

Chapter 5

Named Entity Recognition with Incomplete Annotations

In this chapter, we present a practical scenario of named entity recognition. We first propose an iterative training approach to this problem and further discuss the possibility of using dependency tree information to solve this task.

5.1 Introduction

Named entity recognition (NER) (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) as one of the most fundamental tasks within natural language processing (NLP) has received significant attention. Most existing approaches to NER focused on a supervised setup, where fully annotated named entity information is assumed to be available during the training phase. However, in practice, obtaining high-quality annotations can be a very laborious and expensive process (Snow et al., 2008). One of the common issues with data annotations is there may be incomplete annotations.

Figure 5.1 shows an example sentence with two named entities "*John Lloyd Jones*" and "*BBC radio*" of type PER (person) and ORG (organization) respectively. Following the standard BIOES tagging scheme (Ramshaw and Marcus, 1999; Ratinov and Roth, 2009), the corresponding gold label sequence is shown below the sentence. When the data annotations are incomplete, certain labels may be missing from the label sequence. Properly defining the task is important, and we argue there are two possible potential pitfalls associated with modeling incomplete annotations, especially for the NER task.

Several previous approaches assume the incomplete annotations can be obtained by simply removing either word-level labels (Fernandes and Brefeld, 2011) or span-level labels (Carlson, Gaffney, and Vasile, 2009). As shown in Figure 5.1, under both assumptions (i.e., *A.1* and *A.2*), there will be words annotated with O labels. The former approach may even lead to sub-entity level annotations (e.g., "*radio*" is annotated as part of an entity). However, we argue such assumptions can be largely unrealistic. In practice, annotators are typically instructed to annotate named entities for complete word spans only (Settles, Craven, and Friedland, 2008; Surdeanu, Nallapati, and Manning, 2010). Thus, sub-entity level annotations or O labels [1] should not be assumed to

[1] Why should the O labels be assumed unavailable? This is because the annotators typically do not actively specify the O labels when working on annotations. If the annotator chooses not to annotate a word, it could either mean it is not part of any entity, *or* the word is actually part of an entity but the annotator neglected it in the annotation process (therefore we have incomplete annotations). However,

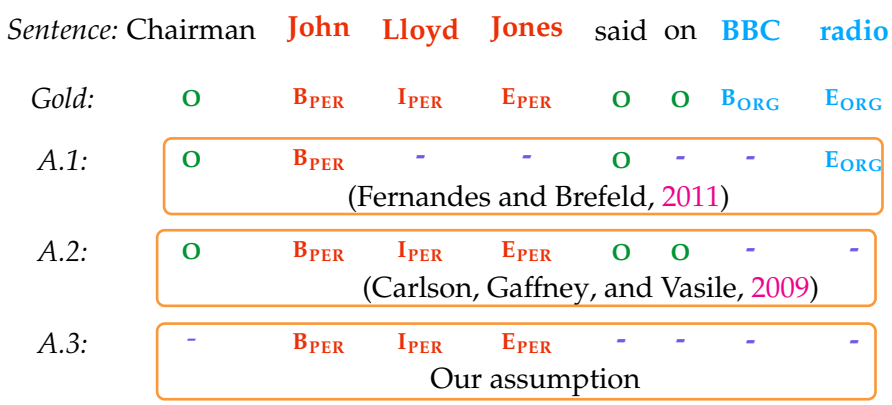


FIGURE 5.1: An example sentence with gold named entity annotations and different assumptions (i.e., *A.1* to *A.3*) on *available labels*. "-" represents a missing label.

be available (*A.3*). Therefore such approaches are making sub-optimal assumptions on the *available labels*.

When the proper assumptions on the available labels are made, one can typically model the missing labels as latent variables and train a latent-variable conditional random fields model (Quattoni, Collins, and Darrell, 2005). One such approach is presented in (Bellare and McCallum, 2007). Their work focused on the citation parsing[2] (i.e., sequence labeling) task which does not suffer from the above issue as no O label is involved. However, though the approach was shown effective in the citation parsing task, we found its effectiveness does not transfer to the NER task even in the absence of the above available labels issue. As we would highlight later, the reason is related to the undesirable assumptions on the *unavailable labels*.

In this work, we tackle the incomplete annotation problem when building an NER system, under a more realistic yet more challenging scenario. We present a novel, effective, yet easy-to-implement approach, and conduct extensive experiments on various datasets and show our approach significantly outperforms several previous approaches.

5.2 Related Work

Previous research efforts on partially annotated data are mostly based on the conditional random fields (CRF) (Lafferty, McCallum, and Pereira, 2001), structured perceptron (Collins, 2002) and max-margin (Tsochantaridis et al., 2005) (e.g. structural support vector machine) models. Bellare and McCallum (2007) proposed a missing label linear-chain CRF[3] which is essentially a latent-variable CRF (Quattoni, Collins, and Darrell, 2005) on citation parsing (McCallum et al., 2000). This model had also been used in

we note that assigning the O label to a word would precisely indicate it is strictly *not* part of any entity, which is not desirable.

[2]The task is to tag the BibTex records with different labels (i.e., "title", "author", "affiliation" and so on).

[3]This model was also named as Partial CRF (Carlson, Gaffney, and Vasile, 2009) and EM Marginal CRF (Greenberg et al., 2018).

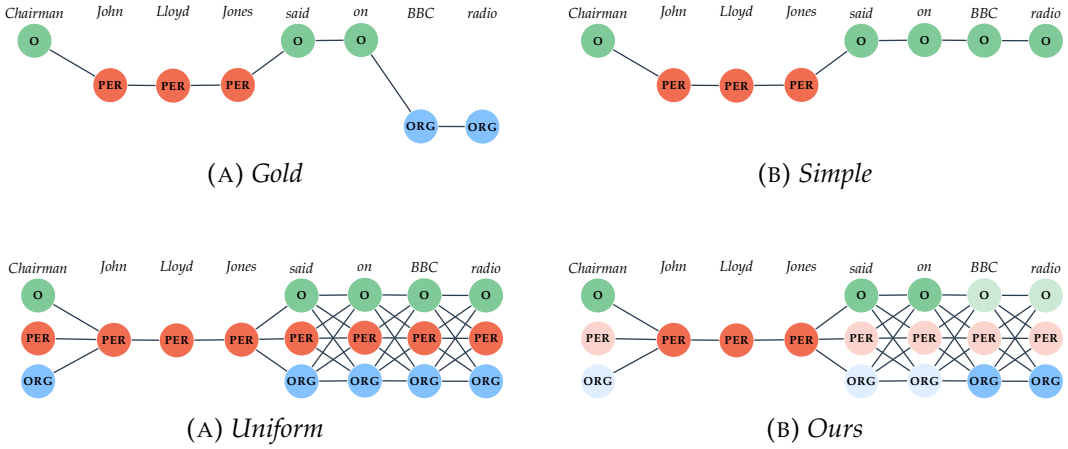
part-of-speeching tagging and segmentation task with incomplete annotations (Tsuboi et al., 2008; Liu et al., 2014; Yang and Vozila, 2014). Yang et al. (2018) showed the effectiveness of such a model on Chinese NER with incomplete annotations due to the fact that they required a certain number of fully annotated data to perform joint training. Greenberg et al. (2018) applied this model on a biomedical NER task and achieved promising performance with incomplete annotations. However, in their assumption for the incomplete annotations, the O labels are still considered, which we believe is not realistic. Carlson, Gaffney, and Vasile (2009) modified the structured perceptron algorithm and defined features only on the tokens with annotated labels in partially labeled sequences. Fernandes and Brefeld (2011) and Lou, Hamprecht, and HCI (2012) proposed to use a large-margin learning framework similar to structured support vector machines with latent variables (Yu and Joachims, 2009).

5.3 Approach

Given the input word sequence $\mathbf{x}$, the NER task is to predict a label sequence $\mathbf{y}$ that encodes the NER information (e.g., in a form following the BIOES tagging scheme). Given a training set that consists of completely labeled data $\mathcal{D}$, one can tackle this problem using a standard linear-chain conditional random field (CRF) (Lafferty, McCallum, and Pereira, 2001) whose loss function is as follows:[4]

$$\mathcal{L}(\mathbf{w}) = -\sum_{i} \log p_{\mathbf{w}}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \tag{5.1}$$

where $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ is the i-th instance from $\mathcal{D}$.



(A) *Gold* (B) *Simple*

(A) *Uniform* (B) *Ours*

FIGURE 5.3: Graphical illustrations on different assumptions on *unavailable labels*, where the entity "*John Lloyd Jones*" of type PER is labeled but "*BBC radio*" of type ORG is missing. Each path refers to one possible complete label sequence, and the density of the color indicates probability (we excluded B and E tags for brevity).

[4] In practice, we also have an L_2 regularization term, which we exclude from the formula for brevity.

Now, assume we have an incomplete label sequence $\mathbf{y}_p^{(i)}$. From such a $\mathbf{y}_p^{(i)}$ we should be able to derive a set of all possible complete label sequences that are compatible with (i.e., contain) the incomplete label sequence, and let us call this set $\mathcal{C}(\mathbf{y}_p^{(i)})$. We can rewrite the above function as:

$$\mathcal{L}(\mathbf{w}) = -\sum_i \log \sum_{\mathbf{y} \in \mathcal{C}(\mathbf{y}_p^{(i)})} q_{\mathcal{D}}(\mathbf{y}|\mathbf{x}^{(i)}) p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}^{(i)})$$

We illustrate in Figure 5.3 several previous approaches as well as our approach. In this example, the entity *BBC radio* of type ORG is not annotated. Figure 5.3(a) shows a single path that corresponds to the gold label sequence. Figure 5.3(b) illustrates a naive approach, where we regard all the missing labels as O labels. This essentially assumes that the q distribution in the above equation puts all probability mass to this single label sequence, which is an incorrect assumption.

Now let us look at what assumptions on q have been made by the existing approach of (Bellare and McCallum, 2007). The model regards the missing labels as latent variables and learns a latent variable CRF using the following loss:

$$-\sum_i \log \sum_{\mathbf{y} \in \mathcal{C}(\mathbf{y}_p^{(i)})} p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}^{(i)}) \tag{5.2}$$

The resulting model is called *missing label linear-chain CRF* (*M-CRF*) [5]. As we can see from the above function, this is essentially equivalent to say q is a uniform distribution that assigns equal probabilities to all possible complete label sequences in $\mathcal{C}(\mathbf{y}_p^{(i)})$.

We believe such an assumption on q that describes *unavailable labels* can be improved. As we can see from the above example in Figure 5.3(d), a more desirable assumption about q is to put more probability mass to a path that is close to the gold path. In practice, their approach worked for the task of citation parsing, where the q distribution may not deviate much from the uniform distribution (Figure 5.3(c)) in such a task. However, in the task of NER, we find such a simple treatment to the q distribution often leads to sub-optimal results (as we can see in the experiments later) as the q distribution is highly skewed due to the large amount of O labels. This observation motivates us to find a proper way to define q that can approximate the gold label distribution in this work.

5.3.1 Estimating q

Inspired by the *classifier stacking* technique used in Nivre and McDonald (2008), we empirically found that a reasonable q distribution can be acquired in a k-fold cross-validation fashion.

We first start with an initialization step where we assign specific labels to words without labels, forming complete label sequences (we will discuss our initialization strategy in experiments). Next, we perform k-fold cross-validation on the training set.

[5] Similar assumptions have also been made by (Carlson, Gaffney, and Vasile, 2009; Fernandes and Brefeld, 2011), but they used structured perceptron (Collins, 2002) instead.

Specifically, each time we train a model with (k-1) folds of the data and based on the learned model we define our q distribution.

We describe two different ways of defining the q distribution, namely the *hard* approach, and the *soft* approach. In the *hard* approach, the resulting q distribution is a collapsed distribution that assigns probability 1 to a single complete label sequence, whereas in the *soft* approach each possible label sequence will get a certain probability score.

In the *hard* approach, after training a model from (k-1) folds, we apply a constrained Viterbi procedure [6] to the sentences in the remaining fold. In the *soft* approach, we use a constrained version of the forward-backward procedure and calculate the marginal probabilities associated with each label at each unlabeled position. The score of each complete label sequence can then be calculated as a product of all such marginal probabilities. We note that in the above procedure the estimation to q depends on the initialization. Thus we iterate the above procedure, which allows us to converge to an improved q.

5.4 Experiments

We conduct experiments on two standard NER datasets – CoNLL-2003 English and CoNLL-2002 Spanish datasets that consist of news articles. We notice that incomplete annotation issue is very common in the industry setup. Therefore we also consider two new datasets from industry – Taobao and Youku datasets[7] consisting of product and video titles in Chinese. We crawled and manually annotated such data with named entities[8]. Table 5.1 shows the statistics of the datasets. The last two columns show the number of entity types and the percentage of words (i.e., c in Table 5.1) that are parts of an NE. Based on our assumption on the *available labels* in Section 5.1, we randomly remove a certain number of entities as well as all O labels and use ρ to represent the ratio of annotated entities. For example, $\rho = 0.6$ means we keep 60% of all the entities and remove the annotations of 40% of the entities. Meanwhile, the O labels are considered unavailable.

Dataset	Training		Validation		Test		Entities	
	#entity	#sent	#entity	#sent	#entity	#sent	#	c (%)
CoNLL-2003	23,499	14,041	5,942	3,250	5,648	3,453	4	23.0
CoNLL-2002	18,796	8,322	4,338	1,914	3,559	1,516	4	12.4
Taobao	29,397	6,000	4,941	998	4,866	1,000	4	51.0
Youku	12,754	8,001	1,580	1,000	1,570	1,001	3	41.7

TABLE 5.1: Data statistics for the datasets.

Taobao is an e-commerce site with various types of products. We crawled the product titles from the website and annotate the titles with 9 entity types. Table 5.2 shows

[6] The algorithm will ensure the resulting complete label sequence is compatible with the incomplete label sequence.

[7] http://www.taobao.com/ and http://www.youku.com/

[8] Details of all datasets can be found in the supplementary material.

the detailed entity information of the Taobao dataset. The leftmost column represent the categorized entity types (i.e., PATTERN, PRODUCT, BRAND and MISC) we used in the experiments.

Youku is a video-streaming website where a number of videos from various domains are presented. We crawled the video titles from the website and again annotate them with 9 entity types. Table 5.3 shows the detailed entity information of the Youku dataset. Specifically, we group the entities into three types (i.e., FIGURE, PROGRAM and MISC) for our experiments.

We further found that there are more entity labels per sentence on these two industry datasets compare to the standard datasets (i.e., CoNLL-2003 and CoNLL-2002). For example, there are 51% of entity labels per sentence (on average) in Taobao dataset whereas there are only 23% of entity labels per centence (on average) in CoNLL-2003 dataset. Because of the high ratio of entity labels in Taobao and Youku datasets, the missing label CRF (M-CRF) can perform a lot less worse compared to the M-CRF on CoNLL datasets.

Grouped Type	**Entity Type**	**#Entity**
PATTERN	Model Type	2,173
PRODUCT	Product Description	5,506
	Core Product	21,958
BRAND	Brand Description	331
	Core Brand	3,430
MISC	Location	1,893
	Person	367
	Literature	814
	Product Specification	2,732

TABLE 5.2: The entity information for the Taobao dataset.

Grouped Type	**Entity Type**	**#Entity**
FIGURE	Figure	4,402
PROGRAM	Variety Show	1,349
	Movie	1,303
	Animation	3,133
	TV Drama	3,087
MISC	Character	446
	Number	1,022
	Location	523
	Song	640

TABLE 5.3: The entity information for the Youku dataset.

We follow Lample et al. (2016) and apply the bidirectional long short-term memory (Hochreiter and Schmidhuber, 1997) (BiLSTM) networks as the neural architecture to all baselines and our approaches. Specifically, we implement the following baselines: a *Simple* model which is a linear-chain LSTM-CRF model and we treat all missing labels as O; the missing label CRF (Bellare and McCallum, 2007; Greenberg et al., 2018) (LSTM-M-CRF) model; the partial perceptron (Carlson, Gaffney, and Vasile, 2009) (LSTM-PP) model, which is a structured perceptron (Collins, 2002) but only considers the scores on the words with available labels; the transductive perceptron (Fernandes and Brefeld, 2011) (LSTM-TP) model where they introduce a Hamming loss function during the perceptron training process; lastly, we train an LSTM-CRF (Lample et al., 2016) with complete annotations as the upper bound (*Complete*). For English and Spanish, we use exactly the same embeddings used in Lample et al. (2016). We train our Chinese character embeddings on the Chinese Gigaword[9] corpus. The resulting implementation achieves 90.9 and 85.8 F-scores on CoNLL-2003 English and CoNLL-2002 Spanish datasets, respectively. These benchmark results are comparable with the results reported in the state-of-the-art NER systems (Lample et al., 2016; Ma and Hovy, 2016; Reimers and Gurevych, 2017).

For initialization in our approaches, we run the *Simple* model on each fold and use the results to initialize our q distribution[10].

Throughout the experiments, we apply the bidirectional long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks as our neural architecture (Lample et al., 2016) for conditional random fields (CRF) (Lafferty, McCallum, and Pereira, 2001). Specifically, the hidden size of LSTM is set to 100, hidden size of character-level LSTM is set to 50, dropout is set to 0.5. During training, we use the Adam optimizer (Kingma and Ba, 2014) with a batch size of 1 and clipping value of 5. Our model is trained with 100 epochs. We select the above hyperparameters based on the best performance on development set.

5.4.1 Baseline Systems

Partial Perceptron We augment the partial perceptron model (Carlson, Gaffney, and Vasile, 2009) with BiLSTM as the neural architecture. In this model, we only consider the scores involved the tokens with available (i.e., annotated) labels during the training process. Algorithm 1 shows the procedure of training a partial perceptron.

Transductive Perceptron This model (Fernandes and Brefeld, 2011) augment an additional Hamming loss function during the update procedure in the structured perceptron. Essentially, they applied the max-margin training strategy in the structured perceptron. First, we obtain a pseudo ground-truth label sequence through a constrained

[9] https://catalog.ldc.upenn.edu/LDC2003T09

[10] Similar to the EM procedure, a good initialization is crucial for our approach. We found using random initialization can lead to substantially worse results and a better initialization can be used to further improve the results. We perform iterative training for our *hard* and *soft* approaches. Empirically, we set the maximum iteration number to 10, which is enough for our approaches to converge and have a stable q distribution.

Algorithm 1 Partial Perceptron

Require: Training data: set of $(\mathbf{x}, \mathbf{y}_p) \in \mathcal{D}$
Ensure: Model parameters $\mathbf{w}$

```
w = 0
for i = 1 ... T do    // T iterations
    for (x, y_p) ∈ D do
        z = arg max_z w · f(x, z)
        if z ≠ y_p then
            //check positions with gold labels
            update(w)
        end if
    end for
end for
```

Viterbi decoding[11]:

$$\mathbf{y}_{pseudo} = \arg\max_{\mathbf{y} \in \mathcal{C}(\mathbf{y}_p)} \mathbf{w}^{\mathbf{T}} \mathbf{f}(\mathbf{x}, \mathbf{y}) \tag{5.3}$$

In other words, we first use the current model parameters to obtain a label sequence as pseudo gold sequence for structured perceptron training. Secondly, they obtain the prediction by max-margin decoding procedure:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} [loss(\mathbf{y}_{pseudo}, \mathbf{y}) + \mathbf{w}^{\mathbf{T}} \mathbf{f}(\mathbf{x}, \mathbf{y})] \tag{5.4}$$

where the loss function is a Hamming loss. Lastly, we perform a perceptron update:

$$\mathbf{w}' = \mathbf{w} + \mathbf{f}(\mathbf{x}, \mathbf{y}_{pseudo}) - \mathbf{f}(\mathbf{x}, \hat{\mathbf{y}}) \tag{5.5}$$

The Hamming loss in the transductive perceptron is defined as follows:

$$loss(\mathbf{y}_{pseudo}, \mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \lambda(t) \tag{5.6}$$

where $\lambda(t) = \lambda_L$ when t is the time step that involves available labels, and $\lambda(t) = \lambda_U$ when t is the time step that involves unavailable labels. During our experiments, we set $\lambda_L = 1$ and $\lambda_U = 0.1$.

Main Results Table 5.4 presents the comparisons among all approaches on four datasets with $\rho = 0.5$ and $k = 2$. Our preliminary experiments show that a larger k value have a negligible effect on the results. A similar finding was also reported in Nivre and McDonald (2008). The *Simple* model has high precision and low recall as it treats unknown labels as O. Previous models for incomplete annotations achieve a much lower F-score compared to the *Simple* model and our approaches. Due to their uniform assumption

[11] This process guarantees the pseudo ground-truth sequence always contains the available labels.

on q over the missing labels, these models typically can recall more entities. The partial perceptron (Carlson, Gaffney, and Vasile, 2009) among these three models yields a relatively lower recall as features are not defined over the words with missing labels.

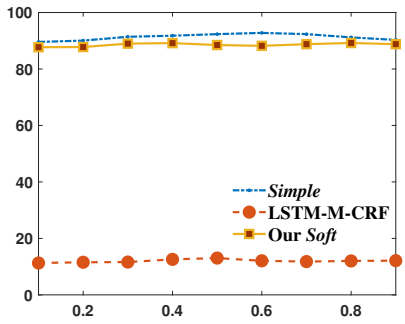
Approach	CoNLL-2003			CoNLL-2002			Taobao			Youku		
	P.	*R.*	*F.*	*P.*	*R.*	*F.*	*P.*	*R.*	*F.*	*P.*	*R.*	*F.*
Simple	93.6	68.6	79.2	86.8	57.0	68.8	83.1	46.7	59.8	91.1	49.1	63.8
LSTM-M-CRF	13.0	90.5	22.8	6.2	84.5	11.6	33.0	83.0	47.2	17.6	83.7	29.1
LSTM-Partial Perceptron	27.6	82.0	41.3	22.3	66.3	33.3	26.6	59.7	36.8	19.7	69.0	30.6
LSTM-Transductive Perceptron	11.7	90.5	20.7	6.1	84.3	11.4	32.9	82.2	47.0	16.0	81.9	26.7
Ours (*hard*)	88.1	89.9	89.0	80.8	82.1	81.5	69.3	77.4	73.1	77.2	79.8	78.5
Ours (*soft*)	89.0	90.1	**89.5**	81.3	82.7	**82.0**	69.7	78.1	**73.7**	78.1	79.6	**78.8**
Complete	91.0	90.8	90.9	85.7	85.8	85.8	82.3	82.6	82.4	83.0	81.7	82.4

TABLE 5.4: Performance comparison between different baseline models and our approaches on 4 datasets with $\rho = 0.5$ (for *Complete* model, $\rho = 1.0$).

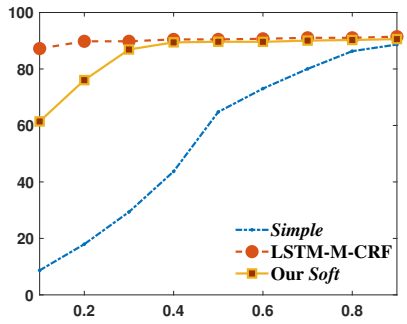
The difference in F-score between these three models and the *Simple* model is more significant on the two CoNLL datasets than on Taobao and Youku. As shown in Table 5.1, the latter two datasets have more words labeled as parts of entities (i.e., a higher c). This means these industrial datasets have less O labels, making such baseline models suffer less from their assumptions on the *unavailable labels*. With a properly learned q distribution, our approaches improves the recall score over the *Simple* model while preserving a high precision. Our *soft* approach consistently achieves a better F-score compared with the *hard* approach on all datasets with $p < 0.001$. Compared to the *Complete* upper bound, our *soft* approach are still more than 3% lower in F-score on the CoNLL-2002, Taobao and Youku datasets. However, we can see that the *soft* approach achieves much higher performance compared to this variant on other datasets. We attribute this phenomenon to our approaches' ability in retrieving most of the entities in the training set. Empirically, we found our *soft* approach can recover 94% of the entities in the training set of the CoNLL-2003 dataset.

The overall results show the underlying scenario is challenging for commonly adopted models in handling incomplete annotations and our approaches can achieve better performance compared with them.

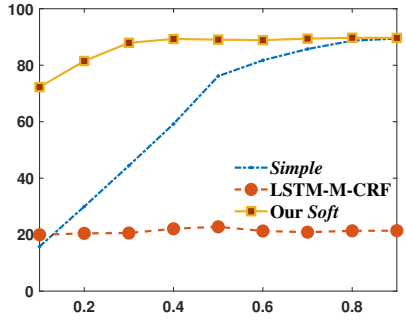
Effect of ρ We conduct experiments with different ρ from 0.1 to 0.9 for our *soft* approach against the *Simple* and LSTM-M-CRF models. Figure 5.4 shows how the precision, recall and F-score on CoNLL-2003 change as we increase ρ. The F-score of the *Simple* baseline increases progressively as ρ increases. LSTM-M-CRF always maintains a low F-score which is not sensitive to different ρ values because of their high recall and low precision values as we can see in Figure 5.4 (a, b). The improvement of our approach attributes to the increase of recall as the precision is constantly high and stable. We can see that our *soft* approach performs particularly well when ρ is larger than 0.3 which indicates a modest amount of missing labels in practice.



(A) Precision



(B) Recall



(C) F-score

FIGURE 5.4: Precision, Recall and F-score with different ρ on CoNLL-2003 dataset.

5.5 Dependency-based Solution: As a Future Work

Essentially, the problem mentioned above is the missing annotations of O labels (*i.e.*, false negative). Observing the relationships between the named entities and dependency trees, we can somehow infer that some dependency relations also have strong correlations with the O labels. On the other hand, if some candidate spans are not forming subtrees, it is very likely that the span is not an entity.

Using such relationships, we present a dependency-based solution which can efficiently solve the incomplete annotation problem rather than iterative training. The underlying solution is a future work for solving this problem.

5.5.1 Entity Candidate Selection

Given the properties we observed in Chapter 3 and Chapter 4, we identify some entity candidates and non-entity words before actual training. Specifically, we can perform the following pre-processing and then train a marginal CRF (Greenberg et al., 2018) but in a semi-Markov manner.

- Obtain the dependency parse trees with existing dependency parser.
- As entities are very likely to form subtrees, for those spans that cannot form subtree, we eliminate those spans in a semi-Markov model.

- On the other hand, we also focus on the dependency relations. If some word are associated with dependency relations (e.g., *neg*) are not possible to be entities, we also eliminate them to be non-entity words.

After the elimination process is done, we have a constrained space of named entities. We can train the model by maximizing the marginalized log-likelihood.

5.6 Conclusions

In this work, we identified several limitations associated with previous assumptions when performing sequence labeling with incomplete annotations, and focused on the named entity recognition task. We present a novel and easy-to-implement solution that works under a realistic and challenging assumption on the incomplete annotations. Through extensive experiments and analysis, we demonstrate the effectiveness of our approach. Finally, we present a potential dependency-based solution as a future work.

Although we focused on the task of named entity recognition in this work, we believe the proposed approach may find applications in some other sequence labeling tasks or other more general structured prediction problems where the issue of incomplete annotations is involved.

Chapter 6

Dependency-based Hybrid Trees for Semantic Parsing

6.1 Introduction

Semantic parsing is a fundamental task within the field of natural language processing (NLP). Consider a natural language (NL) sentence and its corresponding meaning representation (MR) as illustrated in Figure 6.2. Semantic parsing aims to transform the natural language sentences into machine interpretable meaning representations automatically. The task has been popular for decades and keeps receiving significant attention from the NLP community. Various systems (Zelle and Mooney, 1996; Kate, Wong, and Mooney, 2005; Zettlemoyer and Collins, 2005; Liang, Jordan, and Klein, 2011) were proposed over the years to deal with different types of semantic representations. Such models include structure-based models (Wong and Mooney, 2006; Lu et al., 2008; Kwiatkowski et al., 2010; Jones, Johnson, and Goldwater, 2012) and neural network based models (Dong and Lapata, 2016; Cheng et al., 2017).

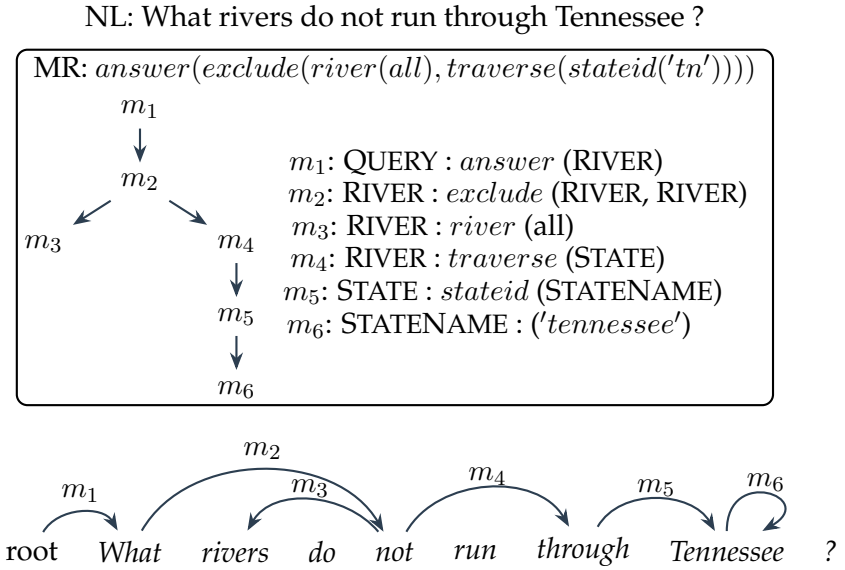


FIGURE 6.1: Top: natural language (NL) sentence; middle: meaning representation (MR); bottom: dependency-based hybrid tree representation.

Following various previous research efforts (Wong and Mooney, 2006; Lu et al., 2008; Jones, Johnson, and Goldwater, 2012), in this work, we adopt a popular class of semantic formalism – logical forms that can be equivalently represented as tree structures. The tree representation of an example MR is shown in the middle of Figure 1.1. One challenge associated with building a semantic parser is that the exact correspondence between the words and atomic semantic units are not explicitly given during the training phase. The key to the building of a successful semantic parsing model lies in the identification of a good *joint* latent representation of both the sentence and its corresponding semantics. Example joint representations proposed in the literature include a chart used in phrase-based translation (Wong and Mooney, 2006), a constituency tree-like representation known as *hybrid tree* (Lu et al., 2008), and a CCG-based derivation tree (Kwiatkowski et al., 2010).

Previous research efforts have shown the effectiveness of using dependency structures to extract semantic representations (Debusmann et al., 2004; Cimiano, 2009; Bédaride and Gardent, 2011; Stanovsky et al., 2016). Recently, Reddy et al. (2016) and Reddy et al. (2017) proposed a model to construct logical representations from sentences that are parsed into dependency structures. Their work demonstrates the connection between the dependency structures of a sentence and its underlying semantics. Although their setup and objectives are different from ours where externally trained dependency parsers are assumed available and their system was trained to use the semantics for a specific down-stream task, the success of their work motivates us to propose a novel joint representation that can explicitly capture dependency structures among words for the semantic parsing task.

In this work, we propose a new joint representation for both semantics and words, presenting a new model for semantic parsing. Our main contributions can be summarized as follows:

- We present a novel *dependency-based hybrid tree* representation that captures both words and semantics in a joint manner. Such a dependency tree reveals semantic dependencies between words which are easily interpretable.
- We show that exact dynamic programming algorithms for inference can be designed on top of our new representation. We further show that the model can be integrated with neural networks for improved effectiveness.
- Extensive experiments conducted on the standard multilingual GeoQuery dataset show that our model outperforms the state-of-the-art models on 7 out of 8 languages. Further analysis confirms the effectiveness of our dependency-based representation.

To the best of our knowledge, this is the first work that models the semantics as latent dependencies between words for semantic parsing.

6.2 Related Work

The literature on semantic parsing has focused on various types of semantic formalisms. The λ-calculus expressions (Zettlemoyer and Collins, 2005) have been popular and

widely used in semantic parsing tasks over recent years (Dong and Lapata, 2016; Gardner and Krishnamurthy, 2017; Reddy et al., 2016; Reddy et al., 2017; Susanto and Lu, 2017a; Cheng et al., 2017). Dependency-based compositional semantics (DCS)[1] was introduced by Liang, Jordan, and Klein (2011), whose extension, λ-DCS, was later proposed by Liang (2013). Various models (Berant et al., 2013; Wang, Berant, and Liang, 2015; Jia and Liang, 2016) on semantic parsing with the λ-DCS formalism were proposed. In this work, we focus on the tree-structured semantic formalism which has been examined by various research efforts (Wong and Mooney, 2006; Kate and Mooney, 2006; Lu et al., 2008; Kwiatkowski et al., 2010; Jones, Johnson, and Goldwater, 2012; Lu, 2014; Zou and Lu, 2018).

Wong and Mooney (2006) proposed the WASP semantic parser that regards the task as a phrase-based machine translation problem. Lu et al. (2008) proposed a generative process to generate natural language words and semantic units in a joint model. The resulting representation is called *hybrid tree* where both natural language words and semantics are encoded into a joint representation. The UBL-s (Kwiatkowski et al., 2010) parser applied the CCG grammar (Steedman, 1996) to model the joint representation of both semantic units and contiguous word sequences which do not overlap with one another. Jones, Johnson, and Goldwater (2012) applied a generative process with Bayesian tree transducer and their model also simultaneously generates the meaning representations and natural language words. Lu (2014) and Lu (2015) proposed a discriminative version of the hybrid tree model of (Lu et al., 2008) where richer features can be captured. Dong and Lapata (2016) proposed a sequence-to-tree model using recurrent neural networks where the decoder can branch out to produce tree structures. Susanto and Lu (2017b) augmented the discriminative hybrid tree model with multilayer perceptron and achieved state-of-the-art performance.

There exists another line of work that applies given syntactic dependency information to semantic parsing. Titov and Klementiev (2011) decomposed a syntactic dependency tree into fragments and modeled the semantics as relations between the fragments. Poon (2013) learned to derive semantic structures based on syntactic dependency trees predicted by the Stanford dependency parser. Reddy et al. (2016) and Reddy et al. (2017) proposed a linguistically motivated procedure to transform syntactic dependencies into logical forms. Their semantic parsing performance relies on the quality of the syntactic dependencies. Unlike such efforts, we do not require external syntactic dependencies, but model the semantic units as latent dependencies between natural language words.

[1] Unlike ours, their work captures dependencies between semantic units but not natural language words.

6.3 Approach

6.3.1 Variable-free Semantics

The variable-free semantic representations in the form of FunQL (Kate, Wong, and Mooney, 2005) used by the de-facto GeoQuery dataset (Zelle and Mooney, 1996) encode semantic compositionality of the logical forms (Cheng et al., 2017). In the tree-structured semantic representations as illustrated in Figure 1.1, each tree node is a semantic unit of the following form:

$$m_i \equiv \tau_\alpha : p_\alpha(\tau^*_\beta)$$

where m_i denotes the complete semantic unit, which consists of semantic type τ_α, function symbol p_α and an argument list of semantic types τ^*_β (here $*$ denotes that there can be 0, 1, or 2 semantic types in the argument list. This number is known as the *arity* of m_i). Each semantic unit can be regarded as a function that takes in other (partial) semantic representations of certain types as arguments and returns a semantic representation of a specific type. For example in Figure 1.1, the root unit is represented by m_1, the type of this unit is QUERY, the function name is $answer$ and it has a single argument RIVER which is a semantic type. With recursive function composition, we can obtain a complete MR as shown in Figure 1.1.

6.3.2 Dependency-based Hybrid Trees

To jointly encode the tree-structured semantics $\boldsymbol{m}$ and a natural language sentence $\boldsymbol{n}$, we introduce our novel *dependency-based hybrid tree*. Figure 6.2 (right) shows the two equivalent ways of visualizing the dependency-based hybrid tree based on the example given in Figure 1.1. In this example, the bold symbol $\boldsymbol{m}$ is the tree-structured semantics $m_1(m_2(m_3, m_4(m_5(m_6))))$ and $\boldsymbol{n}$ is the sentence $\{w_1, w_2, \cdots, w_8\}$[2]. Our dependency-based hybrid tree $\boldsymbol{t}$ consists of a set of dependencies between the natural language words, each of which is labeled with a semantic unit. Formally, a dependency arc is represented as (w_p, w_c, m_i), where w_p is the *parent* of this dependency, w_c is the *child*, and m_i is the semantic unit that serves as the *label* for the dependency arc. A valid dependency-based hybrid tree (with respect to a given semantic representation) allows one to recover the correct semantics from it. Thus, one constraint is that for any two adjacent dependencies (w_p, w_c, m_i) and (w'_p, w'_c, m_j), where $w_c \equiv w'_p$, m_i must be the parent of m_j in the tree-structured representation $\boldsymbol{m}$. For example, in Figure 6.2, the dependencies (*not*, *through*, m_4) and (*through*, *Tennessee*, m_5) satisfy the above condition. However, we cannot replace (*through*, *Tennessee*, m_5) with, for example, (*through*, *Tennessee*, m_6), since m_6 is not the child of m_4. Furthermore, the number of children for a word in the dependency tree should be consistent with the arity of the corresponding semantic unit that points to it. For example, "*not*" has 2 children in our dependency-based hybrid tree representation because the semantic unit m_2 (i.e., RIVER : $exclude$ (RIVER, RIVER)) has arity 2. Also, "*rivers*" is the leaf as m_3, which points to it, has arity 0. We will discuss in Section 6.3.4 on how to derive the set of allowable dependency-based hybrid trees for a given $(\boldsymbol{m}, \boldsymbol{n})$ pair.

[2]We also introduce a special token "*root*" as w_0.

To understand the potential advantages of our new joint representation, we compare it with the *relaxed hybrid tree* representation (Lu, 2014), which is illustrated on the left of Figure 6.2. We highlight some similarities and differences between the two representations from the *span level* and *word level* perspectives.

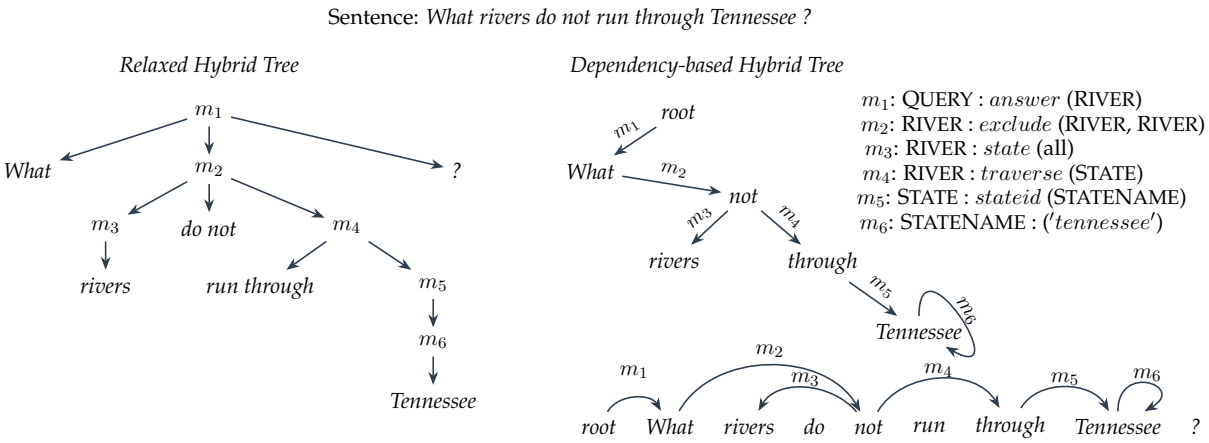


FIGURE 6.2: The *relaxed hybrid tree* (left) (Lu, 2014) and our *dependency-based hybrid tree* (right) as well as the flat representation (bottom right) of the example in Figure 1.1.

In a relaxed hybrid tree representation, words and semantic units jointly form a constituency tree-like structure, where the former are leaves and the latter are internal nodes of such a joint representation. Such a representation is able to capture alignment between the natural language words and semantics at the span level.[3] For example, m_2 covers the span from "*rivers*" to "*Tennessee*", which allows the interactions between the semantic unit and the span to be captured. Similarly, in our dependency-based hybrid tree, such span level word-semantics correspondence can also be captured. For example, the arc between "*not*" and "*through*" is labeled by the semantic unit m_4. This also allows the interactions between m_4 and words within the span from "*not*" to "*through*" to be captured.

While both models are able to capture the span-level correspondence between words and semantics, we can observe that in the relaxed hybrid tree, some words within the span are more directly related to the semantic unit (e.g., "*do not*" are more related to m_2) and some are not. Specifically, in their representation, the span level information assigned to the parent semantic unit always contains the span level information assigned to all its child semantic units. This may not always be desirable and may lead to irrelevant features. In fact, Lu (2014) also empirically showed that the span-level features may not always be helpful in their representation. In contrast, in our dependency-based hybrid tree, the span covered by m_2 is from "*What*" to "*not*", which only consists of the span level information associated with its first child semantic units. Therefore, our representation is more flexible in capturing the correspondence between words and semantics at the span level, allowing the model to choose the relevant span for features.

Furthermore, our representation can also capture precise interactions between words through dependency arcs labeled with semantic units. For example, the semantic unit

[3] We refer readers to (Lu, 2014) for more details.

m_4 on the dependency arc from "*not*" to "*through*" in our representation can be used to capture their interactions. However, such information could not be straightforwardly captured in a relaxed hybrid tree, which is essentially a constituency tree-like representation. In the same example, consider the word "*not*" that bridges two arcs labeled by m_2 and m_4. Lexical features defined over such arcs can be used to indirectly capture the interactions between semantic units and guide the tree construction process. We believe such properties can be beneficial in practice, especially for certain languages. We will examine their significance in our experiments later.

6.3.3 Expressiveness of Dependency-based Hybrid Trees

The original hybrid tree representation (Lu, 2014) belongs to the class of constituency parse tree (Chomsky, 2002). The context-free grammars (CFGs) are able to precisely describe our languages and help us understand the complexity of languages (Johnson, 1998). While a constituency parse tree represents the nesting of multi-word constituents, a dependency parse tree (Kübler, McDonald, and Nivre, 2009) represents the dependencies between individual words. There has been much discussion on the dependency grammars and the phrase structure grammars (Miller, 1999). Though the constituency parse trees have been dominant for years, the syntactic annotation has shifted the focus to the dependency structures[4] (Eisenstein, 2018). The dependency trees are able to describe the direct grammatical relations which can be extracted from the constituents in the phrase structure parse tree (De Marneffe, MacCartney, and Manning, 2006). Thus, we believe that there is a formal proof for the expressiveness of the proposed dependency-based hybrid tree representations compared with the hybrid tree representations. Because our dependency-based hybrid trees are not exactly same as traditional dependency trees where each word is attached to exactly one parent. We will leave the formal proof for future work.

6.3.4 Dependency Patterns

To define the set of allowable dependency-based hybrid tree representation so as to allow us to perform exact inference later, we introduce the *dependency patterns* as shown in Table 6.1. We use **A**, **B** or **C** to denote the abstract semantic units with arity 0, 1, and 2, respectively. We use **W** to denote a contiguous word span, and **X** and **Y** to denote the first and second child semantic unit, respectively.

Abstract Semantic Unit	Arity	Dependency Pattern
A	0	**W**, **W**
B	1	**X**, **W** **X**, **X** **W**
C	2	**X** **Y**, **Y** **X**

TABLE 6.1: List of dependency patterns.

We explain these patterns with concrete cases in Figure 6.3 based on the example in Figure 6.2. For the first case, the semantic unit m_3 has arity 0, the pattern involved

[4]Up to now, we have 157 treebanks for 90 languages in Universal Dependencies.

is **WW**, indicating both the left-hand and right-hand sides of "*rivers*" (under the dependency arc with semantic unit m_3) are just word spans (**W**, whose length could be zero). In the second case, the semantic unit m_4 has arity 1, the pattern involved is **WX**, indicating the left-hand side of "*through*" (under the arc of semantic unit m_4) is a word span and the right-hand side should be handled by the first child of m_4 in the semantic tree, which is m_5 in this case. In the third case, the semantic unit m_2 has two arguments, and the pattern involved in the example is **XY**, meaning the left-hand and right-hand sides should be handled by the first and second child semantic units (i.e., m_3 and m_4), respectively.[5] The final case illustrates that we also allow self-loops on our dependency-based hybrid trees, where an arc can be attached to a single word.[6] To avoid an infinite number of self-loops over a word, we set a maximum depth c to restrict the maximum number of recurrences, which is similar to the method introduced in (Lu, 2015).

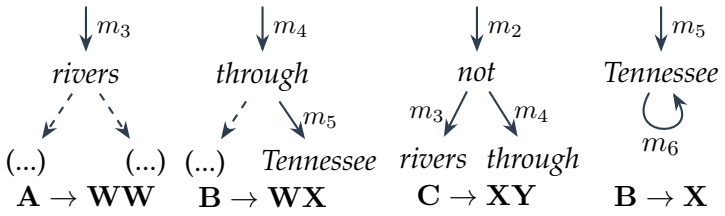


FIGURE 6.3: Example dependency patterns used in the dependency-based hybrid tree of Figure 6.2.

Based on the dependency patterns, we are able to define the set of all possible allowable *dependency-based hybrid tree* representations. Each representation essentially belongs to a class of *projective* dependency trees where semantic units appear on the dependency arcs and (some of the) words are selected as nodes. The semantic tree can be constructed by following the arcs while referring to the dependency patterns involved.

6.3.5 Model

Given the natural language words $\boldsymbol{n}$, our task is to predict $\boldsymbol{m}$, which is a tree-structured meaning representation, consisting of a set of semantic units as the nodes in the semantic tree. We use $\boldsymbol{t}$ to denote a dependency-based hybrid tree (as shown in Figure 6.2), which jointly encodes both natural language words and the gold meaning representation. Let $\mathcal{T}(\boldsymbol{n}, \boldsymbol{m})$ denote all the possible dependency-based hybrid trees that contain the natural language words $\boldsymbol{n}$ and the meaning representation $\boldsymbol{m}$. We adopt the widely-used structured prediction model conditional random fields (CRF) (Lafferty, McCallum, and Pereira, 2001). The probability of a possible meaning representation $\boldsymbol{m}$ and dependency-based hybrid tree $\boldsymbol{t}$ for a sentence $\boldsymbol{n}$ is given by:

$$P_{\mathbf{w}}(\boldsymbol{m}, \boldsymbol{t}|\boldsymbol{n}) = \frac{e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})}}{\sum_{\boldsymbol{m}',\boldsymbol{t}'\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m}')} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m}',\boldsymbol{t}')}}$$

[5] Analogously, the pattern **YX** would mean m_4 handles the left-hand side and m_3 right-hand side.

[6] The limitations associated with disallowing such a pattern have been discussed in the previous work of (Lu, 2015).

where $\mathbf{f}(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$ is the feature vector defined over the $(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$ tuple, and $\mathbf{w}$ is the parameter vector. Since we do not have the knowledge of the "true" dependencies during training, $\boldsymbol{t}$ is regarded as a latent-variable in our model. We marginalize $\boldsymbol{t}$ in the above equation and the resulting model is a latent-variable CRF (Quattoni, Collins, and Darrell, 2005):

$$\begin{aligned} P_{\mathbf{w}}(\boldsymbol{m}|\boldsymbol{n}) &= \sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{n}, \boldsymbol{m})} P_{\mathbf{w}}(\boldsymbol{m}, \boldsymbol{t}|\boldsymbol{n}) \\ &= \frac{\sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{n}, \boldsymbol{m})} e^{\mathbf{w} \cdot \mathbf{f}(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})}}{\sum_{\boldsymbol{m}', \boldsymbol{t}' \in \mathcal{T}(\boldsymbol{n}, \boldsymbol{m}')} e^{\mathbf{w} \cdot \mathbf{f}(\boldsymbol{n}, \boldsymbol{m}', \boldsymbol{t}')}} \end{aligned} \tag{6.1}$$

Given a dataset $\mathcal{D}$ of $(\boldsymbol{n}, \boldsymbol{m})$ pairs, our objective is to minimize the negative log-likelihood:[7]

$$\mathcal{L}(\mathbf{w}) = -\sum_{(\boldsymbol{n}, \boldsymbol{m}) \in \mathcal{D}} \log \sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{n}, \boldsymbol{m})} P_{\mathbf{w}}(\boldsymbol{m}, \boldsymbol{t}|\boldsymbol{n}) \tag{6.2}$$

The gradient for model parameter w_k is:

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_k} =& \sum_{(\boldsymbol{n}, \boldsymbol{m}) \in \mathcal{D}} \sum_{\boldsymbol{m}', \boldsymbol{t}} \mathbf{E}_{P_{\mathbf{w}}(\mathbf{m}', \boldsymbol{t}|\boldsymbol{n})}[f_k(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})] \\ & - \sum_{(\boldsymbol{n}, \boldsymbol{m}) \in \mathcal{D}} \sum_{\boldsymbol{t}} \mathbf{E}_{P_{\mathbf{w}}(\boldsymbol{t}|\boldsymbol{n}, \boldsymbol{m})}[f_k(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})] \end{aligned}$$

where $f_k(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$ represents the number of occurrences of the k-th feature. With both the objective and gradient above, we can minimize the objective function with standard optimizers, such as L-BFGS (Liu and Nocedal, 1989) and stochastic gradient descent. Calculation of these expectations involves all possible dependency-based hybrid trees. As there are exponentially many such trees, an efficient inference procedure is required. We will present our efficient algorithm to perform exact inference for learning and decoding in the next section.

6.3.6 Learning and Decoding

We propose dynamic-programming algorithms to perform efficient and exact inference, which will be used for calculating the objective and gradients discussed in the previous section. The algorithms are inspired by the inside-outside style algorithm (Baker, 1979), graph-based dependency parsing (Eisner, 2000; Koo and Collins, 2010; Shi, Huang, and Lee, 2017), and the *relaxed hybrid tree* model (Lu, 2014; Lu, 2015). As discussed in Section 6.3.4, our latent dependency trees are projective as in traditional dependency parsing (Eisner, 1996; Nivre and Scholz, 2004; McDonald, Crammer, and Pereira, 2005) – the dependencies are non-crossing with respect to the word order (see bottom of Figure 1.1).

[7] We ignore the L_2 regularization term for brevity.

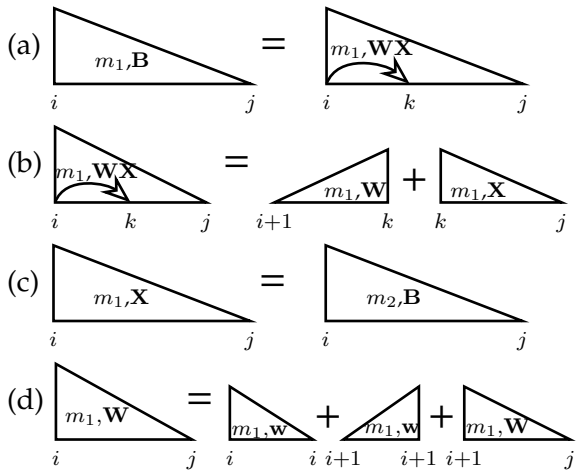


FIGURE 6.4: The dynamic-programing structures and derivation of our model. The other direction is symmetric. See supplementary material for the complete structures.

The objective function in Equation 6.2 can be further decomposed into the following form[8]:

$$\mathcal{L}(\mathbf{w}) = -\sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \log \sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m})} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})} + \sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \log \sum_{\boldsymbol{m}',\boldsymbol{t}'\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m}')} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m}',\boldsymbol{t}')}$$

We can see the first term is essentially the combined score of all the possible latent structures containing the pair $(\boldsymbol{n}, \boldsymbol{m})$. The second term is the combined score for all the possible latent structures containing $\boldsymbol{n}$. We show how such scores can be calculated in a factorized manner, based on the fact that we can recursively decompose a dependency-based hybrid tree based on the dependency patterns we introduced.

Formally, we introduce two interrelated dynamic-programming structures that are similar to those used in graph-based dependency parsing (Eisner, 2000; Koo and Collins, 2010; Shi, Huang, and Lee, 2017), namely *complete span* and *complete arc span*. Figure 6.4a shows an example of *complete span* (left) and *complete arc span* (right). The *complete span* (over $[i, j]$) consists of a headword (at i) and its descendants on one side (they altogether form a subtree), a dependency pattern and a semantic unit. The *complete arc span* is a span (over $[i, j]$) with a dependency between the headword (at i) and the modifier (at k). We use $C_{i,j,p,m}$ to denote a complete span, where i and j represent the indices of the headword and endpoint, p is the dependency pattern and m is the semantic unit. Analogously, we use $A_{i,k,j,p,m}$ to denote a complete arc span where i and k are used to denote the additional dependency from the word at the i-th position as headword to the word at the k-th position as modifier.

As we can see from the derivation in Figure 6.4, each type of span can be constructed from smaller spans in a bottom-up manner. Figure 6.4a shows that a complete span is constructed from a complete arc span following the dependency patterns in Table 6.1. Figure 6.4b shows a complete arc span can be simply constructed from two smaller complete spans based on the dependency pattern. In Figure 6.4c and 6.4d, we further

[8]Regularization term is excluded for brevity.

show how such two complete spans with pattern $\mathbf{X}$ (or $\mathbf{Y}$) and $\mathbf{W}$ can be constructed. Figure 6.4c illustrates how to model a transition from one semantic unit to another where the parent is m_1 and the child is m_2 in the semantic tree. If m_2 has arity 1, then the pattern is $\mathbf{B}$ following the dependency patterns in Table 6.1. For spans with a single word, we use the lowercase $\mathbf{w}$ as the pattern to indicate this fact, as shown in Figure 6.4d. They are the atomic spans used for building larger spans. As the complete span in Figure 6.4d is associated with pattern $\mathbf{W}$, which means the words within this span are under the semantic unit m_1, we can incrementally construct this span with atomic spans. We illustrate the construction of a complete dependency-based hybrid tree in the supplementary material.

Our final goal during training for a sentence $\boldsymbol{n} = \{w_0, w_1, \cdots, w_N\}$ is to construct all the possible *complete spans* that cover the interval $[0, N]$, which can be represented as $C_{0,N,\cdot,\cdot}$. Similar to the chart-based dependency parsing algorithms (Eisner, 1996; Eisner, 2000; Koo and Collins, 2010), we can obtain the *inside* and *outside* scores using our dynamic-programming derivation in Figure 6.4 during the inference process, which can then be used to calculate the objective and feature expectations. Since the spans are defined by at most three free indices, the dependency pattern and the semantic unit, our dynamic-programming algorithm requires $\mathcal{O}(N^3M)$ time[9] where M is the number of semantic units. The resulting complexity is the same as the relaxed hybrid tree model (Lu, 2014).

During decoding, we can find the optimal (tree-structured) meaning representation $\boldsymbol{m}^*$ for a given input sentence $\boldsymbol{n}$ by the Viterbi algorithm. This step can also be done efficiently with our dynamic-programming approach, where we switch from marginal inference to MAP inference:

$$\boldsymbol{m}^*, \boldsymbol{t}^* = \underset{\boldsymbol{m},\boldsymbol{t} \in \mathcal{T}(\boldsymbol{n},\boldsymbol{m})}{\arg\max} \; e^{\mathbf{w} \cdot \mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})} \tag{6.3}$$

A similar decoding procedure has been used in previous work (Lu, 2014; Durrett and Klein, 2015) with CKY-based parsing algorithm.

6.3.7 Features

As shown in Equation 6.1, the features are defined on the tuple $(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$. With the dynamic-programming procedure, we can define the features over the structures in Figure 6.2. Our feature design is inspired by the hybrid tree model (Lu, 2015) and graph-based dependency parsing (McDonald, Crammer, and Pereira, 2005). Table 6.2 shows the feature templates for the example in Figure 6.2. Specifically, we define simple unigram features (concatenation of a semantic unit and a word that directly appears under the unit), pattern features (concatenation of the semantic unit and the child pattern) and transition features (concatenation of the parent and child semantic units). They form our basic feature set.

Additionally, with the structured properties of dependencies, we can define dependency-related features (McDonald, Crammer, and Pereira, 2005). We use the parent (head)

[9] We omit a small constant factor associated with patterns.

Feature Type	Examples
Word	"m_4 & *run*", "m_4 & *through*"
Pattern	"m_2 & $\mathbf{XY}$", "m_4 & $\mathbf{WX}$"
Transition	"m_2 & m_3", "m_2 & m_4"
Head word	"m_2 & *What*", "m_4 & *not*"
Modifier word	"m_2 & *not*", "m_4 & *through*"
Bag of words	"m_4 & *not*", "m_4 & *run*", "m_4 & *through*"

TABLE 6.2: Features for the example in Figure 6.2.

and child (modifier) words of the dependency as features. We also use the bag-of-words covered under a dependency as features. The dependency features are useful in helping improve the performance as we can see in the experiments section.

6.3.8 Neural Component

Following the approach used in Susanto and Lu (2017b), we could further incorporate neural networks into our latent-variable graphical model. The integration is analogous to the approaches described in the neural CRF models (Do and Artieres, 2010; Durrett and Klein, 2015; Gormley, 2015; Lample et al., 2016), where we use neural networks to learn distributed feature representations within our graphical model.

We employ a neural architecture to calculate the score associated with each dependency arc (w_p, w_c, m) (here w_p and w_c are the parent and child words in the dependency and m is the semantic unit over the arc), where the input to the neural network consists of words (i.e., (w_p, w_c)) associated with this dependency and the neural network will calculate a score for each possible semantic unit, including m. The two words are first mapped to word embeddings $\mathbf{e}_p$ and $\mathbf{e}_c$ (both of dimension d). Next, we use a bilinear layer[10] (Socher et al., 2013; Chen, Bolton, and Manning, 2016) to capture the interaction between the parent and the child in a dependency:

$$r_i = \mathbf{e}_p^{\mathbf{T}} \mathbf{U}_i \mathbf{e}_c$$

where r_i represents the score for the i-th semantic unit and $\mathbf{U}_i \in \mathbb{R}^{d \times d}$. The scores are then incorporated into the probability expression in Equation 6.1 during learning and decoding. As a comparison, we also implemented a variant where our model directly takes in the average embedding of $\mathbf{e}_p$ and $\mathbf{e}_c$ as additional features, without using our neural component.

6.4 Experiments

Data and evaluation methodology We conduct experiments on the publicly available variable-free version of the GeoQuery dataset, which has been widely used for

[10] Empirically, we also tried multilayer perceptron but the bilinear model gives us better results.

semantic parsing (Wong and Mooney, 2006; Lu et al., 2008; Jones, Johnson, and Goldwater, 2012). The dataset consists of 880 pairs of natural language sentences and the corresponding tree-structured semantic representations. This dataset is annotated with eight languages. The original annotation of this dataset is English (Zelle and Mooney, 1996) and Jones, Johnson, and Goldwater (2012) annotated the dataset with three more languages: German, Greek and Thai. Lu and Ng (2011) released the Chinese annotation and Susanto and Lu (2017b) annotated the corpus with three additional languages: Indonesian, Swedish and Farsi. In order to compare with previous work (Jones, Johnson, and Goldwater, 2012; Lu, 2015), we follow the standard splits with 600 instances for training and 280 instances for testing. To evaluate the performance, we follow the standard evaluation procedure used in various previous works (Wong and Mooney, 2006; Lu et al., 2008; Jones, Johnson, and Goldwater, 2012; Lu, 2015) to construct the Prolog query from the tree-structured semantic representation using a standard and publicly available script. The queries are then used to retrieve the answers from the GeoQuery database, and we report accuracy and F_1 scores.

Hyperparameters We set the maximum depth c of the semantic tree to 20, following Lu (2015). The L_2 regularization coefficient is tuned from 0.01 to 0.05 using 5-fold cross-validation on the training set. The Polyglot (Al-Rfou, Perozzi, and Skiena, 2013) multilingual word embeddings[11] (with 64 dimensions) are used for all languages. We use L-BFGS (Liu and Nocedal, 1989) to optimize the DEPHT model until convergence and stochastic gradient descent (SGD) with a learning rate of 0.05 to optimize the neural DEPHT model. We implemented our neural component with the Torch7 library (Collobert, Kavukcuoglu, and Farabet, 2011). Our complete implementation is based on the StatNLP[12] structured prediction framework (Lu, 2017).

6.4.1 Baseline Systems

We run the released systems of several state-of-the-art semantic parsers, namely the WASP parser (Wong and Mooney, 2006), HYBRIDTREE model (Lu et al., 2008), UBL system (Kwiatkowski et al., 2010), *relaxed hybrid tree* (RHT) (Lu, 2015)[13], the sequence-to-tree (SEQ2TREE) model (Dong and Lapata, 2016), the *neural hybrid tree* (NEURAL HT) model (Susanto and Lu, 2017b), and the multilingual semantic parser (Susanto and Lu, 2017a) with single language (MSP-SINGLE) as input. The results for TREETRANS (Jones, Johnson, and Goldwater, 2012) are taken from their paper.

6.4.2 Results and Discussion

Table 6.3 (top) shows the results of our dependency-based hybrid tree model compared with non-neural models which achieve state-of-the-art performance on the GeoQuery dataset. Our model DEPHT achieves competitive performance and outperforms the previous best system RHT on 6 languages.

Improvements on the Indonesian dataset are particularly striking (+11.8 absolute points in F_1). We further investigated the outputs from both systems on Indonesian by

[11] The embeddings are fixed to avoid overfitting.
[12] https://gitlab.com/sutd_nlp/statnlp-core
[13] (Lu, 2015) is an extension of the original *relaxed hybrid tree* (Lu, 2014), which reports improved results.

Type	System/Model	English (en)		Thai (th)		German (de)		Greek (el)	
		Acc.	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*
Non-Neural	WASP	71.1	77.7	71.4	75.0	65.7	74.9	70.7	78.6
	HYBRIDTREE	76.8	81.0	73.6	76.7	62.1	68.5	69.3	74.6 6
	UBL	82.1	82.1	66.4	66.4	75.0	75.0	73.6	73.7
	TREETRANS	79.3	79.3	78.2	78.2	74.6	74.6	75.4	75.4
	RHT	86.8	86.8	80.7	80.7	75.7	75.7	79.3	79.3
Neural	SEQ2TREE†	84.5	-	71.9	-	70.3	-	73.1	-
	MSP-SINGLE†	83.5	-	72.1	-	69.3	-	74.2	-
	NEURAL HT (J=0)	87.9	87.9	82.1	82.1	75.7	75.7	81.1	81.1
	NEURAL HT (J=1)	88.6	88.6	84.6	84.6	76.8	76.8	79.6	79.6
	NEURAL HT (J=2)	**90.0**	**90.0**	82.1	82.1	73.9	73.9	80.7	80.7
Non-Neural	(This work) DEPHT	86.8	86.8	81.8	81.8	76.1	76.1	80.4	80.4
Non-Neural	(This work) DEPHT + embedding	87.5	87.5	83.9	83.9	75.0	75.0	81.1	81.1
Neural	(This work) DEPHT + NN	89.3	89.3	**86.7**	**86.7**	**78.2**	**78.2**	**82.9**	**82.9**

TABLE 6.3: Performance comparison with state-of-the-art models on GeoQuery dataset. († represents the system is using lambda-calculus expressions as meaning representations.)

Type	System/Model	Chinese (zh)		Indonesian (id)		Swedish (sv)		Farsi (fa)	
		Acc.	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*
Non-Neural	WASP	48.2	51.6	74.6	79.8	63.9	71.5	46.8	54.1
	HYBRIDTREE	56.1	58.4	66.4	72.8	61.4	70.5	51.8	58.6
	UBL	63.8	63.8	73.8	73.8	78.1	78.1	64.4	64.4
	TREETRANS	-	-	-	-	-	-	-	-
	RHT	76.1	76.1	75.0	75.0	79.3	79.3	73.9	73.9
Neural	SEQ2TREE†	73.3	-	80.7	-	80.8	-	70.5	-
	MSP-SINGLE†	74.9	-	79.8	-	77.5	-	72.2	-
	NEURAL HT (J=0)76.8	76.8	76.1	76.1	81.1	81.1	75.0	75.0	
	NEURAL HT (J=1)	75.4	75.4	78.6	78.6	82.9	82.9	76.1	76.1
	NEURAL HT (J=2)	81.1	81.1	81.8	81.8	83.9	83.9	74.6	74.6
Non-Neural	(This work) DEPHT	81.4	81.4	86.8	86.8	85.4	85.4	73.9	73.9
Non-Neural	(This work) DEPHT + embedding	81.4	81.4	87.5	87.5	87.1	87.1	73.6	73.6
Neural	(This work) DEPHT + NN	**82.9**	**82.9**	**88.7**	**88.7**	**87.3**	**87.3**	**77.9**	**77.9**

TABLE 6.4: Performance comparison with state-of-the-art models on GeoQuery dataset. († represents the system is using lambda-calculus expressions as meaning representations.)

doing error analysis. We found 40 instances that are incorrectly predicted by RHT are correctly predicted by DEPHT. We found that 77.5% of the errors are due to incorrect alignment between words and semantic units as shown in Table 6.5.

Error Cause	%
Missing semantic unit	12.5
Incorrect semantic unit	10.0
Incorrect alignment	77.5

TABLE 6.5: Different types of errors in the prediction of *relaxed hybrid tree* model.

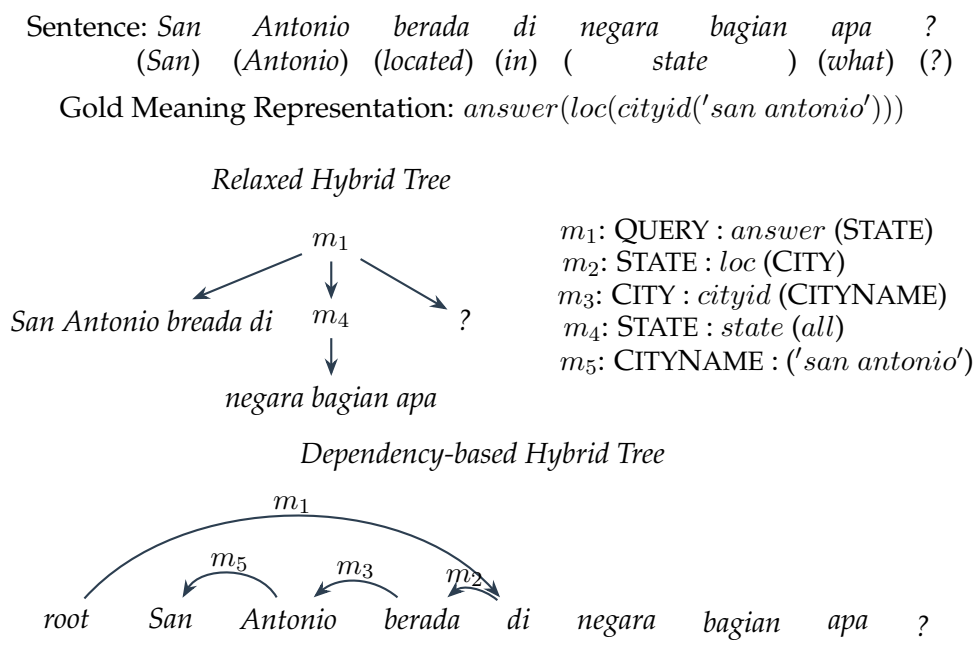


FIGURE 6.5: Example results from DEPHT and RHT on Indonesian.

Figure 6.5 shows an example of such errors where the relaxed hybrid tree fails to capture the correct alignment.

We can see the question is asking "*What state is San Antonio located in?*". However, the natural language word order in Indonesian is different from English, where the phrase "*berada di*" that corresponds to m_2 (i.e., loc) appears between "*San Antonio*" (which corresponds to m_5 – $'san\ antonio'$) and "*what*" (which corresponds to m_1 – $answer$). Such a structural non isomorphism issue between the sentence and the semantic tree makes the relaxed hybrid tree parser unable to produce a joint representation with valid word-semantics alignment. This issue makes the RHT model unable to predict the semantic unit m_2 (i.e., loc) as RHT has to align the words "*San Antonio*" which should be aligned to m_5 before aligning "*berada di*". However, m_5 has arity 0 and cannot have m_2 as its child. Thus, it would be impossible for the RHT model to predict such a meaning representation as output. In contrast, we can see that our dependency-based hybrid tree representation appears to be more flexible in handling such cases. The dependency between the two words "*di*" (*in*) and "*berada*" (*located*) is also well captured by the arc between them that is labeled with m_2. The error analysis reveals the flexibility of our joint representation in different languages in terms of the word ordering, indicating that the novel dependency-based joint representation is more robust and suffers less from language-specific characteristics associated with the data.

Effectiveness of dependency To investigate the helpfulness of the features defined over latent dependencies, we conduct ablation tests by removing the dependency-related features. Table 6.6 shows the performance of augmenting different dependency

	en	th	de	el	zh	id	sv	fa
DEPHT basic	75.0	82.1	70.4	74.6	76.1	71.9	73.9	69.3
BASIC+HM feats.	80.7	**83.9**	**75.7**	79.2	81.1	85.0	81.1	72.5
BASIC+BOW feats.	**86.1**	83.2	73.9	**79.3**	**81.4**	**86.1**	**85.4**	**73.2**
DEPHT	86.8	81.8	76.1	80.4	81.4	86.8	85.4	73.9

TABLE 6.6: F_1 scores of our model with different dependency features.

features in our DEPHT model with basic features. Specifically, we investigate the performance of head word and modifier word features (HM) and also the bag-of-words features (BOW) that can be extracted based on dependencies. It can be observed that dependency features associated with the words are crucial for all languages, especially the BOW features.

Effectiveness of neural component The bottom part of Table 6.3 shows the performance comparison among models that involve neural networks. Our DEPHT model with embeddings as features can outperform neural baselines across several languages (i.e., Chinese, Indonesian and Swedish). From the table, we can see the neural component is effective, which consistently gives better results than DEPHT and the approach that uses word embedding features only. Susanto and Lu (2017b) presented the NEURAL HT model with different window size J for their multilayer perceptron. Their performance will differ with different window sizes, which need to be tuned for each language. In our neural component, we do not require such a language-specific hyperparameter, yet our neural approach consistently achieves the highest performance on 7 out of 8 languages compared with all previous approaches. As both the embeddings and the neural component are defined on the dependency arcs, the superior results also reveal the effectiveness of our dependency-based hybrid tree representation.

6.5 Conclusions

In this work, we present a novel *dependency-based hybrid tree* model for semantic parsing. The model captures the underlying semantic information of a sentence as latent dependencies between the natural language words. We develop an efficient algorithm for exact inference based on dynamic-programming. Extensive experiments on benchmark dataset across 8 different languages demonstrate the effectiveness of our newly proposed representation for semantic parsing.

Future work includes exploring alternative approaches such as transition-based methods (Nivre, Hall, and Nilsson, 2006; Chen and Manning, 2014) for semantic parsing with latent dependencies, applying our dependency-based hybrid trees on other types of logical representations (e.g., lambda calculus expressions and SQL (Finegan-Dollak et al., 2018)) as well as multilingual semantic parsing (Jie and Lu, 2014; Susanto and Lu, 2017a).

6.6 Dependency-based Hybrid Tree for SQL Parsing

As SQL is getting popular recently, one natural question to ask is that *Can we solve SQL parsing with Dependency-based Hybrid Trees?* Because the SQL itself is actually a sequential structure but not in a tree format, it could be difficult to solve it with dependency-based hybrid trees as well as designing certain grammar structures. However, sequential structure is a special tree structure where each node has only one child. It is still possible to employ the dependency-based hybrid tree models for tackling the SQL problems. But we need to design different grammars and hybrid pattern under such a special meaning representation.

Chapter 7

Potential Research Directions

We propose some potential research directions by further making use of the relationship between dependencies and our downstream structured prediction tasks (i.e., NER and semantic parsing). Specifically, the property that "*entities often form subtrees*" allows us to design a joint model for joint dependency parsing and named entity recognition (Section 7.1). On the other hand, as we show that our dependency-based hybrid trees (Jie and Lu, 2018) not only have a good performance on English but especially on other languages where word order matters (Ahmad et al., 2019). The natural question for NER is: *Is the linear-chain structure always the best for every language?* By answering this question, we attempt to build the NER system exploring different graphical models besides just linear-chain structures. Finally, as the dependency-based hybrid trees are flexible, we could propose a universal dependency-based hybrid tree framework for different kinds of meaning representations.

7.1 Joint Dependency Parsing and Named Entity Recognition

Dependency grammar has gained significant attention in research field thanks to its efficiency and simplicity compared to phrase-structure parsing. Unlike a phrase structure, there is no phrasal node in a dependency structure; each node in a dependency structure represents a word-token in a sentence except for a root node. Dependencies refer to both syntactic and semantic relations between nodes. Using the dependency information can enhance the performance for some tasks like named entity recognition (NER), semantic role labeling (SRL) and semantic parsing (Kazama and Torisawa, 2008; Ling and Weld, 2012; Johansson and Nugues, 2008; Hacioglu, 2004; Poon and Domingos, 2009). In dependency parsing, the basic first-order model (Eisner, 1996) is defined by a decomposition of a tree into head-modifier dependencies. The syntactic relationship is represented by a directed arc between a *head* word and a *modifier* word. The $head$ word is considered more important in this relationship and the *modifier* word is supplementing the meaning of the *head* word. For example, Figure 7.1 shows an example of a dependency structure, which contains a dependency from the head word "$from$" to the modifier word "$States$" and this dependency infers some relationship between this two words.

However, this kind of dependency only captures the relationship between individual words while ignoring the integrity of multiple words. As in the example shown in Figure 7.1, "$United\ States$" is an entity with the type of *location*. The words "$United\ States$" (the rounded rectangle in Figure 7.1) should be considered a group in the dependency structure. Thus we do not need to parse the whole entity into single words,

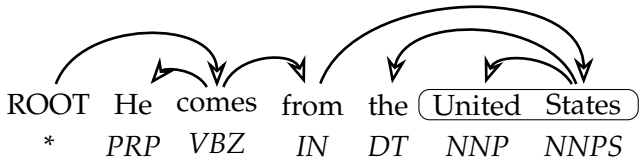


FIGURE 7.1: An example dependency structure given a sentence and its POS tags. "$United\ States$" is a *location* entity.

such that we can capture the relationship between $from$ and $United\ States$ instead of $from$ and $States$. One of the advantages of doing this is helping us to extract the predicate information like "comes_from(*He*, /ENTITY/*United States*)" in this example. This kind of structure with embedded entity information will further facilitate the other applications like SRL and semantic parsing. To the best of our knowledge, this is the first work focusing on the task of joint dependency parsing and named entity recognition.

We would like to present a new joint model for efficient dependency parsing and named entity recognition, which is a feature-based CRF model over hybrid-tree structure (Lu et al., 2008; Klein and Manning, 2004).

7.1.1 Eisner's First-order Parser

In our joint model, we need to maintain the original dependency structure which is the first-order model introduced by (Eisner, 2000). The first-order dynamic programming structure and derivation are shown in Figure 7.2. We summarize the dynamic programming derivation here and later we can understand our joint model clearly.

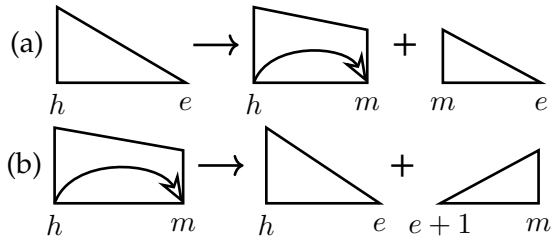


FIGURE 7.2: The dynamic-programing structures and derivation of the first-order model. (a) complete span, (b) incomplete span. Only right-direction derivation is shown due to space limit.

There are two basic data structures in this dynamic-programming derivation: one is called *complete* span and another one is called *incomplete* span. A complete span, referred to Figure 7.2(a), consist of a head word h and a end word e. In this span, only the head word has no parent, all the other words in the span are already attached to a parent word. For the incomplete span, referred to Figure 7.2(b), there is still a head word h but the word m at the other end is not complete yet. There is also an explicit dependency from a head word h to a modifier word m indicated on the span.

Followed the notation from this work (Koo and Collins, 2010), a complete span is denoted as $C_{h,e}$ and an incomplete span is denoted as $I_{h,m}$. As shown in Figure 7.2, a complete span can be decomposed to the combination of an incomplete span and a

complete span. The index m in Figure 7.2(a) is the split point for this decomposition. An incomplete span can be decomposed to two complete spans where there is also a free index e in Figure 7.2(b). To parse a sentence into a dependency structure, we are essentially finding a construction process that can form the complete span from the first word (root) to the last word. The dependency structure can then be extracted from all the incomplete spans where contains an arc from the h to m. Using a chart-parsing algorithm like CKY can help us efficiently parse the sentence into this structure.

7.1.2 Joint Dependency and Named Entity Model

In order to apply the dynamic programming technique in our joint model, we used similar derivation in the first-order parsing model. Figure 7.3 shows the dynamic programming derivation of our joint model. The intuition behind is that each span in the derivation can be represented as a type, which can be *person*, *organization*, *GPE*, *MISC*. Besides entity type, we introduced two more types that can facilitate our model later. A type ***E*** which indicates there is at least one entity inside this span and a type ***N*** which indicates there is no entity inside this span. We use a label ***T*** to include all this three types: *E*, *N* and entity types.

In addition to complete spans and incomplete spans in the first-order model, we augmented the span type with *general* spans and *typed* spans. For a general span, there is no type information attached with it and but it has a symbol that can record its parent's type as in Figure 7.3(a) and Figure 7.3(b) the children have a symbol to store the type of their parent. The general spans can be decomposed to a typed span with a specific entity type as shown in Figure 7.3(a) and Figure 7.3(b). The type *T'* attached with the child and the *L* in the parent's symbol may not be the same since the child span might have other types that are not required to be same as the larger span. For example, a span (*h*,*m*) with a type *E* can have a smaller span inside that is attached with a type *person*.

Following the above example, we should define some rules that what types that smaller spans can have according to the type of their grandparents. Table 7.1 shows the rule that can be applied in our model. Since we can only look at the relationship between children and parent in our derivation, we are able to capture the exact relationship between a larger typed span and the two smaller typed spans which compose the larger span. Since type *E* means at least one entity inside this span, so the child type

parent type	child type
E'	*E*, *N*, *person*
N'	*N*
person'	*person*

TABLE 7.1: Derivation rule for the typed spans. Derivation for other entity types like GPE are same as *person*. They are not shown in the table due to space limit.

can be either *E*, *N* or *person*. If the parent type is already *N* or *person*, then we define that the child type must be same as the parent type since there is also no nested entities in our dataset.

To parse a sentence, our model tries to find an optimal construction of the sentence over all the possible hybrid-tree structure built by the derivation. This can be done similarly as the first-order parsing by the chart-parsing technique. The decomposition process (Figure 7.3(a) and 7.3(b)) of the typed span is same as first-order parsing since the entity type T is not the free parameters like the third free index (the split point). Since each derivation is at most defined by two fixed indices and an entity type, this parsing algorithm will need $\mathcal{O}(n^2|T|)$ space. The complexity of our parsing algorithm is still $\mathcal{O}(n^3|T|)$, where n is the length of a sentence.

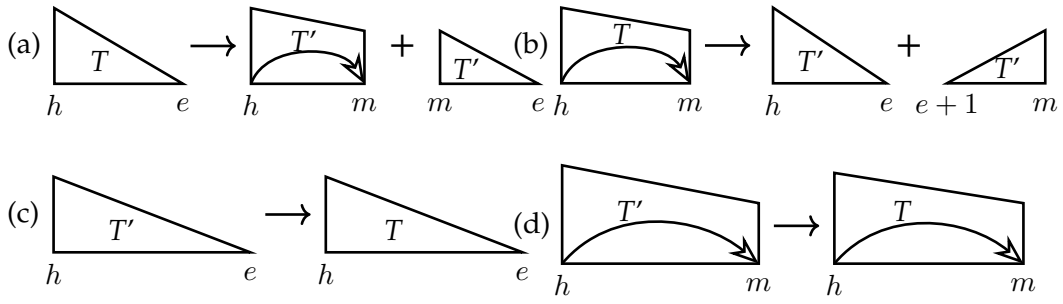


FIGURE 7.3: The dynamic-programming structures and derivation of our joint dependency parsing and named entity recognition model. Typed spans are decomposed to general spans and general spans are decomposed to typed spans.

.

By jointly parsing the entity and dependency together, they can interact and help each other in the joint model. We found that most of the named entities in our dataset have an outmost dependency that headed by either the beginning or the end of an entity. Inspired by this structure, we can then extract the joint features in this kind of span and as well as their entity features and dependency features. For example, the dependency structure of an organization entity *National University of Singapore* is shown in Figure 7.4. With the derivation in Figure 7.3, the above entity forms an incomplete span from *National* to *Singapore*. This typed span is headed by *Singapore* and ended at *National*. We can then extract the whole entity features with the help of dependency information. In a similar way, the entity information can help identify the head or modifier to find the best dependency structure.

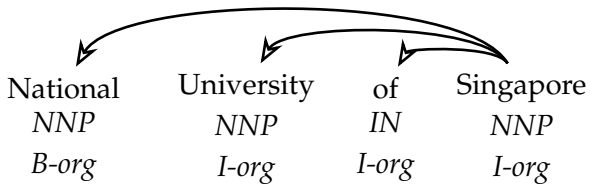


FIGURE 7.4: An organization entity in dependency structure.

7.1.3 Log-Linear Modeling

Conditional random fields (CRFs)(Lafferty, McCallum, and Pereira, 2001) have been widely used in natural language processing problems like part-of-speech tagging, named entity recognition (McCallum and Li, 2003) and semantic role labeling (Cohn and Blunsom, 2005). In this work, we also defined a discriminative log-linear model over the dynamic programming derivation.

To be more specific, the probability of predicting a possible output $\mathbf{y}$ (a dependency tree structure with entity information in each span) given an input sentence $\mathbf{x}$ is given as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(\mathbf{w^T f}(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w^T f}(\mathbf{x}, \mathbf{y}'))} \tag{7.1}$$

where $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is the feature vector defined on the sentence $\mathbf{x}$ and the tree structure $\mathbf{y}$. The best tree structure should satisfy that $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$.

We aim to minimize the negative joint log-likelihood with L_2 regularization for our dataset, which defines the objective function as:

$$\mathcal{L}(\mathbf{w}) = \sum_i \log \sum_{\mathbf{y}'} \exp(\mathbf{w}^T \mathbf{f}(\mathbf{x}_i, \mathbf{y}')) - \sum_i \mathbf{w}^T \mathbf{f}(\mathbf{x}_i, \mathbf{y}_i) + \lambda \mathbf{w}^T \mathbf{w} \tag{7.2}$$

where $(\mathbf{x}_i, \mathbf{y}_i)$ is the i-th training instance and λ is the L_2 regularization parameter which is set to 0.7 in our experiments. The partial derivative of $\mathcal{L}$ with respect to each parameter w_k is:

$$\frac{\partial \mathcal{L}}{\partial w_k} = \mathbf{E}_{p(\mathbf{y}'|\mathbf{x}_i)} \left[f_k(\mathbf{x}_i, \mathbf{y}') \right] - \sum_i f_k(\mathbf{x}_i, \mathbf{y}_i) + 2\lambda w_k \tag{7.3}$$

The convexity and differentiability of objective function guarantees convergence to the global optimum, and using inside-outside algorithm can efficiently compute the above gradient. There are numbers of techniques can be used for optimizing objective function. We used L-BFGS (Byrd et al., 1995) as our optimization method.

7.1.4 Preliminary Experiments

We conduct our experiments on two datasets, one is LDC2011T01 SemEval-2010 Task 1 OntoNotes English corpus [1], which is the only one we can find has both the annotated dependency and named entity information. The other one is the broadcast news from LDC2013T19 OntoNotes Release 5.0 [2]. Following the work on named entity recognition (Finkel and Manning, 2009), we use the following 6 types of documents: ABC, CNN, MNB, NBC, PRI and VOA. However, there is no dependency annotation in OntoNotes 5.0 dataset. We applied the LTH[3] converter (Johansson and Nugues, 2007) to convert the constituent parse tree to dependency structure [4].

[1] https://catalog.ldc.upenn.edu/LDC2011T01

[2] https://catalog.ldc.upenn.edu/LDC2013T19

[3] http://nlp.cs.lth.se/software/treebank_converter/

[4] There are some tags (e.g. META, EDITED) from OntoNotes treebank do not appeared in the penn treebank. We removed those sentences that cannot be converted. Furthermore, we also abandon the sentences with non-projective dependency after conversion.

7.1.5 Dataset

This English corpus contains both the dependency relationships and named entities information, as well as the POS tag. In general, we consider the following three entity types in the dataset: PERSON, ORGANIZATION, GPE (geo-political entity, such as a city or a country). There are total 3,358 sentences for training, 684 sentences in development set and 1,058 sentences for testing. Since we only focus on the projective dependency parsing, the sentences with non-projective dependency are eliminated in our dataset.

	Training+Dev		Testing	
	invalid	#entity	invalid	#entity
PER	18	1,457	4	346
GPE	41	1,356	8	380
ORG	254	1,199	69	382
#total	313	4,012	91	1,108

TABLE 7.2: Dataset Statistics. The total number of entities and the number of "invalid" entities in training and testing data. "Invalid" entities are the special case describe in Section **??**.

Table 7.2 shows the statistics of our dataset. For "invalid", the entity itself cannot form a continuous subtree. We have a total of 4,012 entities in our training and development data and 1,108 entities in testing data. It is obvious to see that there are few invalid entities for PERSON and GPE.

7.1.6 Results

Table 7.3 shows the NER performance of our joint model compared to linear-chain CRF model on the SemEval 2010 dataset. At the moment, the results of the joint model do not surpass the performance of the linear-chain CRF model. Such a result gives us some guidance to further improve our model as our joint model loses some information regarding the label transition.

	Just NER			Our Joint Model		
	Precision	Recall	F-measure	Precision	Recalll	F-measure
PERSON	79.08%	79.77%	79.42%	79.55%	80.92%	80.23%
GPE	78.77%	87.89%	83.08%	73.72%	87.11%	79.86%
ORGANIZATION	67.53%	54.45%	60.29%	59.01%	43.72%	50.23%
MISC	80.11%	63.32%	70.73%	75.79%	67.69%	71.51%
Overall	**76.78%**	**70.75%**	**73.65%**	72.87%	69.48%	71.13%

TABLE 7.3: Named Entity Recognition Results on SemEval 2010 dataset with both dependency and named entity information annotated.

7.2 Dependency as Graphical Models for NER

As different languages have different word order (Ahmad et al., 2019), we believe that the classic linear-chain structures might not always be the perfect solution for NER in every language. To this end, we propose to use the universal dependencies as guidance for building the graphical model. Our goal is to build a graphical model that follows the universal dependency trees but enjoys the linear-time complexity as in the linear-chain CRF model.

7.3 Universal Hybrid Tree Framework for Semantic Parsing

As mentioned in Chapter 2, we have many types of meaning representations such as lambda-calculus expression, FunQL, AMR, and SQL. However, researchers proposed different models for different types of meaning representations. We argue that it is not practical to have many different models in this scenario. In order to achieve a semantic parser that can easily adapt to different meaning representation, we propose a prototype of a dependency-based hybrid tree framework. As the hybrid tree framework only requires hybrid tree grammar for us to build a general parser. We propose a series of general hybrid tree grammar that potentially apply to all types of meaning representation.

Chapter 8

Conclusions

In this thesis, we leverage the potential of dependency trees in structured prediction tasks. In particular, we focus on the semantically meaningful tasks: named entity recognition and semantic parsing.

We first present our observations from the datasets that have both named entity and dependency parse tree annotations. Our observations suggest that most of the entities (about 99%) in the datasets, including datasets in other languages, form contiguous subtrees in the dependency trees (Jie, Muis, and Lu, 2017). On the other hand, we also statistically show that certain entity types have strong correlations with some specific dependency relations (Jie and Lu, 2019), which could be strong indicators for the existence of named entities.

Inspired by the first observation, we build an efficient dependency-guided model (Jie, Muis, and Lu, 2017) based on the semi-Markov CRF (Sarawagi and Cohen, 2004) for named entity recognition. Making use of the "*subtree*" properties, we can define certain constraints to eliminate the search space for entity candidates. Specifically, we can disallow some *invalid* spans to become entities. Theoretically, we prove that our dependency-guided model achieves a linear-time complexity on average while having comparable or even better performance compared with the semi-Markov CRF model on the OntoNotes Broadcast News dataset.

As our first approach did not fully make use of the complete dependency structures and dependency relations, we further investigate a deeper relationship between the dependency trees and named entities. First, the *long-distance* interactions within the entity boundary are able to help us identify those long entities because such interactions *shorten* the path from the entity left boundary to the right boundary. Secondly, the *long-distance* interactions cross the entity boundary are able to help us identify the direct connection between the entity head word and the word outside entity. Thirdly, certain entity types have strong correlations with some specific dependency relations. According to these properties, we propose a dependency-guided LSTM-CRF model (Jie and Lu, 2019) to capture these properties. Empirically, the dependency-guided LSTM-CRF model achieves state-of-the-art performance on the datasets with four languages. Through the detailed analysis, we demonstrate that the proposed model indeed captures the properties above.

We further present a scenario where annotations are incomplete for named entity recognition and discuss the possibility to solve the problem with a dependency-based solution. We first argue that the current incomplete annotation scenarios are not realistic and the O labels should not be assumed to be available. This practical scenario makes the commonly used approaches such as marginal CRF (Greenberg et al., 2018)

suffer from "*low precision and high recall*" problem. We then propose an iterative training approach to address this problem and such an approach achieves much better performance compared with the strong baselines. We also discuss the possibility of using dependency trees to help us identify those non-entity words based on the relationships mentioned above.

Finally, we attempt to solve the semantic parsing task without explicit dependency annotations but regarding the dependency trees as a latent variable. Motivated by previous approaches (Reddy et al., 2016; Wang, Berant, and Liang, 2015) which transform the dependency trees into meaning representations (*e.g.*, lambda-calculus expression and AMR), we presented a dependency-based hybrid tree representation to jointly encode tree-structured FunQL and the natural language words. Such a representation is flexible in terms of word order. Empirically, the experimental results show the proposed hybrid tree model achieves state-of-the-art performance on 7 out of 8 languages and the improvement is significant. Our further analysis found that our model consistently has promising results because the proposed dependency-based representation can capture the difference of word order in different languages.

We pose some research challenges in Chapter 7 for our future research purpose. The challenges including a joint task for dependency parsing and NER, a task that using dependency as guidance to design a graphical model, and a universal dependency-based hybrid tree framework for different types of meaning representation in semantic parsing.

Bibliography

Ahmad, Wasi et al. (2019). "On Difficulties of Cross-Lingual Transfer with Order Differences: A Case Study on Dependency Parsing". In: *Proceedings of NAACL*. URL: https://www.aclweb.org/anthology/N19-1253.pdf.

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). "Contextual String Embeddings for Sequence Labeling". In: *Proceedings of COLING*. URL: https://aclweb.org/anthology/C18-1139.

Al-Rfou, Rami, Bryan Perozzi, and Steven Skiena (2013). "Polyglot: Distributed Word Representations for Multilingual NLP". In: *Proceedings of CoNLL*. URL: http://www.aclweb.org/anthology/W13-3520.

Artzi, Yoav, Kenton Lee, and Luke Zettlemoyer (2015). "Broad-coverage CCG semantic parsing with AMR". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D15-1198.pdf.

Baker, James K (1979). "Trainable grammars for speech recognition". In: *The Journal of the Acoustical Society of America* 65.S1, S132–S132.

Banarescu, Laura et al. (2013). "Abstract meaning representation for sembanking". In: *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*. URL: https://www.aclweb.org/anthology/W13-2322.pdf.

Bédaride, Paul and Claire Gardent (2011). "Deep semantics for dependency structures". In: *Proceedings of CICLing*.

Bellare, Kedar and Andrew McCallum (2007). "Learning extractors from unlabeled text using relevant databases". In: *Proceedings of International Workshop on Information Integration on the Web*. URL: https://www.aaai.org/Papers/Workshops/2007/WS-07-14/WS07-14-002.pdf.

Bengio, Yoshua, Nicholas Léonard, and Aaron Courville (2013). "Estimating or propagating gradients through stochastic neurons for conditional computation". In: *arXiv preprint arXiv:1308.3432*.

Berant, Jonathan and Percy Liang (2014). "Semantic parsing via paraphrasing". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P14-1133.pdf.

Berant, Jonathan et al. (2013). "Semantic parsing on freebase from question-answer pairs". In: *Proceedings of EMNLP*.

Berg-Kirkpatrick, Taylor, David Burkett, and Dan Klein (2012). "An Empirical Investigation of Statistical Significance in NLP". In: *Proceedings of EMNLP-CoNLL*. URL: https://www.aclweb.org/anthology/D12-1091.

Byrd, Richard H et al. (1995). "A Limited memory algorithm for bound constrained optimization". In: *SIAM Journal on Scientific Computing* 16.5, pp. 1190–1208.

Cai, Qingqing and Alexander Yates (2013). "Large-scale semantic parsing via schema matching and lexicon extension". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P13-1042.pdf.

Carlson, Andrew, Scott Gaffney, and Flavian Vasile (2009). "Learning a Named Entity Tagger from Gazetteers with the Partial Perceptron". In: *Proceedings of AAAI Spring Symposium: Learning by Reading and Learning to Read*. URL: https://www.aaai.org/Papers/Symposia/Spring/2009/SS-09-07/SS09-07-003.pdf.

Che, Wanxiang et al. (2018). "Towards Better UD Parsing: Deep Contextualized Word Embeddings, Ensemble, and Treebank Concatenation". In: *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. URL: http://www.aclweb.org/anthology/K18-2005.

Chen, Danqi, Jason Bolton, and Christopher D Manning (2016). "A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task". In: *Proceedings of ACL*.

Chen, Danqi and Christopher D Manning (2014). "A Fast and Accurate Dependency Parser using Neural Networks." In: *Proceedings of EMNLP*.

Cheng, Jianpeng et al. (2017). "Learning Structured Natural Language Representations for Semantic Parsing". In: *Proceedings of ACL*.

Chiu, Jason P. C. and Eric Nichols (2016). "Named Entity Recognition with Bidirectional LSTM-CNNs". In: *Transactions of the Association of Computational Linguistics* 4, pp. 357–370. URL: https://aclweb.org/anthology/Q16-1026.

Chomsky, Noam (1956). "Three models for the description of language". In: *IRE Transactions on information theory* 2.3, pp. 113–124.

— (2002). *Syntactic structures*. Walter de Gruyter.

Cimiano, Philipp (2009). "Flexible semantic composition with DUDES". In: *Proceedings of ICCS*.

Cohn, Trevor and Philip Blunsom (2005). "Semantic role labelling with tree conditional random fields". In: *Proceedings of CoNLL*.

Collins, Michael (2002). "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms". In: *Proceedings of EMNLP*. URL: http://www.aclweb.org/anthology/W02-1001.

Collobert, Ronan, Koray Kavukcuoglu, and Clément Farabet (2011). "Torch7: A matlab-like environment for machine learning". In: *Proceedings of BigLearn, NIPS workshop*.

Corro, Caio and Ivan Titov (2019). "Learning Latent Trees with Stochastic Perturbations and Differentiable Dynamic Programming". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P19-1551.pdf.

Cortes, Corinna and Vladimir Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297. URL: https://link.springer.com/content/pdf/10.1007/BF00994018.pdf.

Cotterell, Ryan and Kevin Duh (2017). "Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields". In: *Proceedings of IJCNLP*. URL: https://www.aclweb.org/anthology/I17-2016.

Cucchiarelli, Alessandro and Paola Velardi (2001). "Unsupervised named entity recognition using syntactic and semantic contextual evidence". In: *Computational Linguistics* 27.1, pp. 123–131.

Damonte, Marco, Shay B Cohen, and Giorgio Satta (2017). "An Incremental Parser for Abstract Meaning Representation". In: *Proceedings of EACL*. URL: https://www.aclweb.org/anthology/E17-1051.pdf.

De Marneffe, Marie-Catherine, Bill MacCartney, Christopher D Manning, et al. (2006). "Generating typed dependency parses from phrase structure parses." In: *Lrec*. Vol. 6, pp. 449–454.

De Marneffe, Marie-Catherine and Christopher D Manning (2008). *Stanford typed dependencies manual*. Tech. rep. Stanford University. URL: https://nlp.stanford.edu/software/dependencies_manual.pdf.

Debusmann, Ralph et al. (2004). "A relational syntax-semantics interface based on dependency grammar". In: *Proceedings of COLING*.

Devlin, Jacob et al. (2019). "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of NAACL*. URL: https://arxiv.org/abs/1810.04805.

Do, Trinh-Minh-Tri and Thierry Artieres (2010). "Neural conditional random fields". In: *Proceedings of AISTAT*.

Dong, Li and Mirella Lapata (2016). "Language to Logical Form with Neural Attention". In: *Proceedings of ACL*.

— (2018). "Coarse-to-Fine Decoding for Neural Semantic Parsing". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P18-1068.

Dozat, Timothy and Christopher D Manning (2017). "Deep biaffine attention for neural dependency parsing". In: *Proceedings of ICLR*. URL: https://openreview.net/pdf?id=Hk95PK9le.

Durrett, Greg and Dan Klein (2015). "Neural CRF Parsing". In: *Proceedings of ACL-IJCNLP*.

Eisenstein, Jacob (2018). *Natural language processing*.

Eisner, Jason (2000). "Bilexical grammars and their cubic-time parsing algorithms". In: *Advances in probabilistic and other parsing technologies*. Springer, pp. 29–61.

Eisner, Jason M (1996). "Three new probabilistic models for dependency parsing: An exploration". In: *Proceedings of COLING*.

Fernandes, Eraldo R and Ulf Brefeld (2011). "Learning from partially annotated sequences". In: *Proceedings of ECML-KDD*. URL: https://link.springer.com/content/pdf/10.1007%2F978-3-642-23780-5_36.pdf.

Finegan-Dollak, Catherine et al. (2018). "Improving Text-to-SQL Evaluation Methodology". In: *Proceedings of ACL*. URL: http://aclweb.org/anthology/P18-1033.

Finkel, Jenny et al. (2005). "Exploring the boundaries: gene and protein identification in biomedical text". In: *BMC bioinformatics* 6.1, p. 1.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning (2005). "Incorporating non-local information into information extraction systems by gibbs sampling". In: *Proceedings of ACL*.

Finkel, Jenny Rose and Christopher D Manning (2009). "Joint parsing and named entity recognition". In: *Proceedings of NAACL*.

— (2010). "Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data". In: *Proceedings of ACL*.

Flanigan, Jeffrey et al. (2014). "A discriminative graph-based parser for the abstract meaning representation". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P14-1134.pdf.

Gardner, Matt and Jayant Krishnamurthy (2017). "Open-Vocabulary Semantic Parsing with both Distributional Statistics and Formal Knowledge." In: *Proceedings of AAAI*.

Gardner, Matt et al. (2017). "AllenNLP: A Deep Semantic Natural Language Processing Platform". In: *Proceedings of Workshop for NLP-OSS*. URL: https://arxiv.org/pdf/1803.07640.pdf.

Ghaddar, Abbas and Phillippe Langlais (2018). "Robust Lexical Features for Improved Neural Network Named-Entity Recognition". In: *Proceedings of COLING*. URL: https://aclweb.org/anthology/C18-1161.

Gormley, Matthew R (2015). "Graphical Models with Structured Factors, Neural Factors, and Approximation-Aware Training". PhD thesis.

Grave, Edouard et al. (2018). "Learning Word Vectors for 157 Languages". In: *Proceedings of LREC*. URL: https://www.aclweb.org/anthology/L18-1550.

Greenberg, Joseph (1963). "Some universals of grammar with particular reference to the order of meaningful elements". In: *In J. Greenberg, ed., Universals of Language. 73-113. Cambridge, MA.*

Greenberg, Nathan et al. (2018). "Marginal Likelihood Training of BiLSTM-CRF for Biomedical Named Entity Recognition from Disjoint Label Sets". In: *Proceedings of EMNLP*. URL: http://aclweb.org/anthology/D18-1306.

Grishman, Ralph and Beth M Sundheim (1996). "Message understanding conference-6: A brief history". In: *Proceedings of COLING*. URL: https://www.aclweb.org/anthology/C96-1079.pdf.

Hacioglu, Kadri (2004). "Semantic role labeling using dependency trees". In: *Proceedings of COLING*. URL: https://www.aclweb.org/anthology/C04-1186.pdf.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780. URL: https://www.mitpressjournals.org/doi/pdfplus/10.1162/neco.1997.9.8.1735.

Honnibal, Matthew and Ines Montani (2017). "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing". To appear.

Jia, Robin and Percy Liang (2016). "Data Recombination for Neural Semantic Parsing". In: *Proceedings of ACL*.

Jie, Zhanming and Wei Lu (2014). "Multilingual semantic parsing: Parsing multiple languages into semantic representations". In: *Proceedings of COLING*.

— (2018). "Dependency-based Hybrid Trees for Semantic Parsing". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D18-1265.pdf.

— (2019). "Dependency-Guided LSTM-CRF for Named Entity Recognition". In: *Proceedings of EMNLP-IJCNLP*. URL: https://www.aclweb.org/anthology/D19-1399.pdf.

Jie, Zhanming, Aldrian Obaja Muis, and Wei Lu (2017). "Efficient dependency-guided named entity recognition". In: *Proceedings of AAAI*. URL: https://arxiv.org/abs/1810.08436.

Jie, Zhanming et al. (2019). "Better modeling of incomplete annotations for named entity recognition". In: *Proceedings of NAACL*. URL: https://www.aclweb.org/anthology/N19-1079.pdf.

Johansson, Richard and Pierre Nugues (2007). "Extended Constituent-to-dependency Conversion for English". In: *Proceedings of NODALIDA 2007*. Tartu, Estonia, pp. 105–112.

Johansson, Richard and Pierre Nugues (2008). "Dependency-based semantic role labeling of PropBank". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D08-1008.pdf.

Johnson, Mark (1998). "PCFG models of linguistic tree representations". In: *Computational Linguistics* 24.4, pp. 613–632.

Jones, Bevan Keeley, Mark Johnson, and Sharon Goldwater (2012). "Semantic parsing with bayesian tree transducers". In: *Proceedings of ACL*.

Jurafsky, Dan (2000). *Speech & language processing*. Pearson Education India. URL: https://web.stanford.edu/~jurafsky/slp3/.

Kamath, Aishwarya and Rajarshi Das (2018). "A Survey on Semantic Parsing". In: *Proceedings of AKBC*. URL: https://arxiv.org/pdf/1812.00978.pdf.

Kate, Rohit J and Raymond J Mooney (2006). "Using string-kernels for learning semantic parsers". In: *Proceedings of COLING/ACL*.

Kate, Rohit J, Yuk Wah Wong, and Raymond J Mooney (2005). "Learning to transform natural to formal languages". In: *Proceedings of AAAI*.

Kazama, Jun'ichi and Kentaro Torisawa (2008). "Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations". In: *Proceedings of ACL*.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*. URL: https://arxiv.org/pdf/1308.3432.pdf.

Kingsbury, Paul and Martha Palmer (2002). "From TreeBank to PropBank". In: *Proceedings of LREC*. URL: http://www.lrec-conf.org/proceedings/lrec2002/pdf/283.pdf.

Kipf, Thomas N and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of ICLR*. URL: https://openreview.net/pdf?id=SJU4ayYgl.

Klein, Dan and Christopher D Manning (2004). "Parsing and hypergraphs". In: *New developments in parsing technology*. Springer, pp. 351–372.

Koehn, Philipp (2004). "Statistical Significance Tests for Machine Translation Evaluation." In: *Proceedings of EMNLP*.

Konstas, Ioannis et al. (2017). "Neural AMR: Sequence-to-Sequence Models for Parsing and Generation". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P17-1014.pdf.

Koo, Terry and Michael Collins (2010). "Efficient third-order dependency parsers". In: *Proceedings of ACL*.

Krishnamurthy, Jayant (2016). "Probabilistic models for learning a semantic parser lexicon". In: *Proceedings of NAACL*. URL: https://www.aclweb.org/anthology/N16-1074.pdf.

Krishnamurthy, Jayant, Pradeep Dasigi, and Matt Gardner (2017). "Neural semantic parsing with type constraints for semi-structured tables". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D17-1160.pdf.

Krishnamurthy, Jayant and Tom M Mitchell (2015). "Learning a compositional semantics for Freebase with an open predicate vocabulary". In: *Transactions of the Association for Computational Linguistics* 3, pp. 257–270. URL: https://www.aclweb.org/anthology/Q15-1019.pdf.

Kübler, Sandra, Ryan McDonald, and Joakim Nivre (2009). "Dependency parsing". In: *Synthesis lectures on human language technologies* 1.1, pp. 1–127.

Kwiatkowski, Tom et al. (2010). "Inducing probabilistic CCG grammars from logical form with higher-order unification". In: *Proceedings of EMNLP*.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: *Proceedings of ICML*.

Lample, Guillaume et al. (2016). "Neural Architectures for Named Entity Recognition". In: *Proceedings of NAACL-HLT*. URL: https://www.aclweb.org/anthology/N16-1030.

Lee, Kenton et al. (2017). "End-to-end Neural Coreference Resolution". In: *Proceedings of EMNLP*. URL: https://aclweb.org/anthology/D17-1018.

Li, Jing et al. (2018). "A survey on deep learning for named entity recognition". In: *arXiv preprint arXiv:1812.09449*. URL: https://arxiv.org/pdf/1812.09449.pdf.

Li, Peng-Hsuan et al. (2017). "Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks". In: *Proceedings of EMNLP*. URL: https://aclweb.org/anthology/D17-1282.

Liang, Percy (2013). "Lambda dependency-based compositional semantics". In: *Technical Report arXiv*.

Liang, Percy, Michael I Jordan, and Dan Klein (2011). "Learning Dependency-Based Compositional Semantics". In: *Proceedings of NNACL*.

Lin, Bill Yuchen and Wei Lu (2018). "Neural Adaptation Layers for Cross-domain Named Entity Recognition". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D18-1226.pdf.

Lin, Dekang and Xiaoyun Wu (2009). "Phrase clustering for discriminative learning". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P09-1116.pdf.

Lin, Kevin et al. (2019). "Grammar-based neural text-to-sql generation". In: *arXiv preprint arXiv:1905.13326*. URL: https://arxiv.org/pdf/1905.13326.pdf.

Ling, Xiao and Daniel S Weld (2012). "Fine-Grained Entity Recognition". In: *Proceedings of AAAI*. URL: http://dl.acm.org/citation.cfm?id=2900728.2900742.

Liu, Dong C and Jorge Nocedal (1989). "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1, pp. 503–528.

Liu, Jingchen, Minlie Huang, and Xiaoyan Zhu (2010). "Recognizing biomedical named entities using skip-chain conditional random fields". In: *Proceedings of the Workshop on BioNLP*.

Liu, Tianyu, Jin-Ge Yao, and Chin-Yew Lin (2019). "Towards improving neural named entity recognition with gazetteers". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P19-1524.pdf.

Liu, Yijia et al. (2014). "Domain Adaptation for CRF-based Chinese Word Segmentation using Free Annotations". In: *Proceedings of EMNLP*. URL: http://www.aclweb.org/anthology/D14-1093.

Lou, Xinghua, Fred A Hamprecht, and IWR HCI (2012). "Structured Learning from Partial Annotations". In: *Proceedings of ICML*. URL: https://icml.cc/2012/papers/753.pdf.

Lu, Wei (2014). "Semantic parsing with relaxed hybrid trees". In: *Proceedings of EMNLP*.

Lu, Wei (2015). "Constrained semantic forests for improved discriminative semantic parsing". In: *Proceedings of ACL*.

— (2017). "A Unified Framework for Structured Prediction: From Theory to Practice". In: *Proceedings of EMNLP (Tutorial)*.

Lu, Wei and Hwee Tou Ng (2011). "A probabilistic forest-to-string model for language generation from typed lambda calculus expressions". In: *Proceedings of EMNLP*.

Lu, Wei and Dan Roth (2015). "Joint mention extraction and classification with mention hypergraphs". In: *Proceedings of EMNLP*.

Lu, Wei et al. (2008). "A generative model for parsing natural language to meaning representations". In: *Proceedings of EMNLP*.

Lukovnikov, Denis et al. (2018). "Translating Natural Language to SQL using Pointer-Generator Networks and How Decoding Order Matters". In: *arXiv preprint arXiv:1811.05303*.

Lyu, Chunchuan and Ivan Titov (2018). "AMR Parsing as Graph Prediction with Latent Alignment". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P18-1037.pdf.

Ma, Xuezhe and Eduard Hovy (2016). "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *Proceedings of ACL*. URL: http://www.aclweb.org/anthology/P16-1101.

Manning, Christopher et al. (2014). "The Stanford CoreNLP natural language processing toolkit". In: *Proceedings of ACL: System Demonstrations*. URL: https://www.aclweb.org/anthology/P14-5010.

Marcheggiani, Diego and Ivan Titov (2017). "Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D17-1159.

Martins, André FT et al. (2019). "Latent Structure Models for Natural Language Processing". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P19-4001.pdf.

Mayhew, Stephen et al. (2019). "Named Entity Recognition with Partially Annotated Training Data". In: *Proceedings of CoNLL*. URL: https://www.aclweb.org/anthology/K19-1060.pdf.

McCallum, Andrew, Dayne Freitag, and Fernando CN Pereira (2000). "Maximum Entropy Markov Models for Information Extraction and Segmentation." In: *Proceedings of ICML*.

McCallum, Andrew and Wei Li (2003). "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons". In: *Proceedings of NAACL*.

McCallum, Andrew Kachites et al. (2000). "Automating the construction of internet portals with machine learning". In: *Information Retrieval* 3.2, pp. 127–163. URL: https://doi.org/10.1023/A:1009953814988.

McCann, Bryan et al. (2018). "The natural language decathlon: Multitask learning as question answering". In: *arXiv preprint arXiv:1806.08730*.

McDonald, Ryan, Koby Crammer, and Fernando Pereira (2005). "Online large-margin training of dependency parsers". In: *Proceedings of ACL*.

Mikolov, Tomas et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Proceedings of NeurIPS*. URL: https://papers.nips.

cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Miller, Philip H (1999). *Strong generative capacity: The semantics of linguistic formalism*. CSLI publications.

Misra, Dipendra and Yoav Artzi (2016). "Neural shift-reduce ccg semantic parsing". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D16-1183.pdf.

Miwa, Makoto and Mohit Bansal (2016). "End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P16-1105.

Muis, Aldrian Obaja and Wei Lu (2016a). "Learning to Recognize Discontiguous Entities". In: *Proceedings of EMNLP*.

— (2016b). "Weak Semi-Markov CRFs for Noun Phrase Chunking in Informal Text". In: *Proceedings of NAACL*.

Nivre, Joakim, Johan Hall, and Jens Nilsson (2006). "Maltparser: A data-driven parser-generator for dependency parsing". In: *Proceedings of LREC*.

Nivre, Joakim and Ryan McDonald (2008). "Integrating graph-based and transition-based dependency parsers". In: *Proceedings of ACL*. URL: http://www.aclweb.org/anthology/P/P08/P08-1108.

Nivre, Joakim and Mario Scholz (2004). "Deterministic dependency parsing of English text". In: *Proceedings of COLING*.

Nivre, Joakim et al. (2016). "Universal dependencies v1: A multilingual treebank collection". In: *Proceedings of LREC*. URL: https://www.aclweb.org/anthology/L16-1262.pdf.

Okanohara, Daisuke et al. (2006). "Improving the scalability of semi-markov conditional random fields for named entity recognition". In: *Proceedings of COLING-ACL*.

Pan, Sinno Jialin and Qiang Yang (2009). "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359. URL: https://www.cse.ust.hk/~qyang/Docs/2009/tkde_transfer_learning.pdf.

Passos, Alexandre, Vineet Kumar, and Andrew McCallum (2014). "Lexicon Infused Phrase Embeddings for Named Entity Resolution". In: *Proceedings of CoNLL*. URL: https://www.aclweb.org/anthology/W14-1609.pdf.

Pasupat, Panupong and Percy Liang (2015). "Compositional semantic parsing on semi-structured tables". In: *Proceedings of the ACL*.

Paszke, Adam et al. (2017). "Automatic differentiation in PyTorch". In: URL: https://openreview.net/pdf?id=BJJsrmfCZ.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "Glove: Global vectors for word representation". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D14-1162.

Peters, Matthew, Sebastian Ruder, and Noah A Smith (2019). "To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks". In: *arXiv preprint arXiv:1903.05987*. URL: https://arxiv.org/pdf/1903.05987.pdf.

Peters, Matthew et al. (2018a). "Deep Contextualized Word Representations". In: *Proceedings of NAACL*. URL: https://aclweb.org/anthology/N18-1202.

Peters, Matthew et al. (2018b). "Dissecting Contextual Word Embeddings: Architecture and Representation". In: *Proceedings of EMNLP*. URL: https://aclweb.org/anthology/D18-1179.

Poon, Hoifung (2013). "Grounded unsupervised semantic parsing". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P13-1092.pdf.

Poon, Hoifung and Pedro Domingos (2009). "Unsupervised semantic parsing". In: *Proceedings of EMNLP*.

Pradhan, Sameer et al. (2012). "CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes". In: *Proceedings of Joint EMNLP-CoNLL: Shared Task*. URL: https://www.aclweb.org/anthology/W12-4501.

Pradhan, Sameer et al. (2013). "Towards robust linguistic analysis using OntoNotes". In: *Proceedings of CoNLL*. URL: https://www.aclweb.org/anthology/W13-3516.

Quattoni, Ariadna, Michael Collins, and Trevor Darrell (2005). "Conditional random fields for object recognition". In: *Proceedings of NIPS*.

Ramshaw, Lance A and Mitchell P Marcus (1999). "Text chunking using transformation-based learning". In: *Proceedings of Third Workshop on Very Large Corpora*. URL: https://www.aclweb.org/anthology/W95-0107.

Ratinov, Lev and Dan Roth (2009). "Design challenges and misconceptions in named entity recognition". In: *Proceedings of CoNLL*.

Recasens, Marta et al. (2010). "Semeval-2010 task 1: Coreference resolution in multiple languages". In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. URL: https://www.aclweb.org/anthology/S10-1001.

Reddy, Siva et al. (2016). "Transforming dependency structures to logical forms for semantic parsing". In: *Transactions of the Association for Computational Linguistics* 4, pp. 127–140. URL: https://www.aclweb.org/anthology/Q16-1010.pdf.

Reddy, Siva et al. (2017). "Universal Semantic Parsing". In: *Proceedings of EMNLP*.

Reimers, Nils and Iryna Gurevych (2017). "Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging". In: *Proceedings of EMNLP*. URL: http://aclweb.org/anthology/D17-1035.

Sachan, Mrinmaya and Eric Xing (2016). "Machine comprehension using rich semantic representations". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P16-2079.pdf.

Sarawagi, Sunita and William W Cohen (2004). "Semi-markov conditional random fields for information extraction". In: *Proceedings of NeurIPS*.

Sasano, Ryohei and Sadao Kurohashi (2008). "Japanese Named Entity Recognition Using Structural Natural Language Processing". In: *Proceedings of IJCNLP*. URL: https://www.aclweb.org/anthology/I08-2080.

Settles, Burr, Mark Craven, and Lewis Friedland (2008). "Active learning with real annotation costs". In: *Proceedings of NIPS Workshop on Cost-Sensitive Learning*. URL: http://burrsettles.com/pub/settles.nips08ws.pdf.

Shi, Tianze, Liang Huang, and Lillian Lee (2017). "Fast (er) Exact Decoding and Global Training for Transition-Based Dependency Parsing via a Minimal Feature Set". In: *Proceedings of EMNLP*.

Smith, Noah A (2019). "Contextual word representations: A contextual introduction". In: *arXiv preprint arXiv:1902.06006*.

Snow, Rion et al. (2008). "Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks". In: *Proceedings of EMNLP*. URL: http://www.aclweb.org/anthology/D08-1027.

Socher, Richard et al. (2013). "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of EMNLP*.

Song, Linfeng et al. (2016). "AMR-to-text generation as a Traveling Salesman Problem". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D16-1224.pdf.

Stanovsky, Gabriel et al. (2016). "Getting more out of syntax with PROPS". In: *arXiv preprint arXiv:1603.01648*.

Steedman, Mark (1996). "Surface structure and interpretation". In:

Strubell, Emma et al. (2017). "Fast and Accurate Entity Recognition with Iterated Dilated Convolutions". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D17-1283.

Surdeanu, Mihai, Ramesh Nallapati, and Christopher Manning (2010). "Legal claim identification: Information extraction with hierarchically labeled data". In: *Proceedings of LREC*. URL: http://www.lrec-conf.org/proceedings/lrec2010/workshops/W23.pdf#page=27.

Susanto, Raymond Hendy and Wei Lu (2017a). "Neural Architectures for Multilingual Semantic Parsing". In: *Proceedings of ACL*.

— (2017b). "Semantic Parsing with Neural Hybrid Trees." In: *Proceedings of AAAI*.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to sequence learning with neural networks". In: *Proceedings of NIPS*.

Sutton, Charles and Andrew McCallum (2004). "Collective Segmentation and Labeling of Distant Entities in Information Extraction". In: *Proceedings of the ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*. URL: https://www.research.ed.ac.uk/portal/files/14900035/tr_04_49.pdf.

Titov, Ivan and Alexandre Klementiev (2011). "A Bayesian model for unsupervised semantic parsing". In: *Proceedings of ACL*.

Tjong Kim Sang, Erik F (2002). "Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition". In: *Proceedings of CoNLL*. URL: http://aclweb.org/anthology/W02-2024.

Tjong Kim Sang, Erik F and Fien De Meulder (2003). "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition". In: *Proceedings of CoNLL*. URL: http://aclweb.org/anthology/W03-0419.

Tsochantaridis, Ioannis et al. (2005). "Large margin methods for structured and interdependent output variables". In: *Journal of machine learning research* 6, pp. 1453–1484. URL: http://www.jmlr.org/papers/volume6/tsochantaridis05a/tsochantaridis05a.pdf.

Tsuboi, Yuta et al. (2008). "Training Conditional Random Fields Using Incomplete Annotations". In: *Proceedings of COLING*. URL: http://www.aclweb.org/anthology/C08-1113.

Vilares, David and Carlos Gómez-Rodríguez (2018). "A Transition-Based Algorithm for Unrestricted AMR Parsing". In: *Proceedings of NAACL*. URL: https://www.aclweb.org/anthology/N18-2023.pdf.

Wang, Chenglong, Marc Brockschmidt, and Rishabh Singh (2018). "Pointing Out SQL Queries From Text". In: *Technical report*.

Wang, Chuan, Nianwen Xue, and Sameer Pradhan (2015a). "A transition-based algorithm for AMR parsing". In: *Proceedings of NAACL*. URL: https://www.aclweb.org/anthology/N15-1040.pdf.

— (2015b). "Boosting transition-based AMR parsing with refined actions and auxiliary analyzers". In: *Proceedings of ACL-IJCNLP*. URL: https://www.aclweb.org/anthology/P15-2141.pdf.

Wang, Chuan et al. (2016). "CAMR at semeval-2016 task 8: An extended transition-based AMR parser". In: *Proceedings of SemEval*. URL: https://www.aclweb.org/anthology/S16-1181.pdf.

Wang, Yushi, Jonathan Berant, and Percy Liang (2015). "Building a semantic parser overnight". In: *Proceedings of ACL-IJCNLP*, pp. 1332–1342.

Weischedel, Ralph et al. (2013). "Ontonotes release 5.0 ldc2013t19". In: *Linguistic Data Consortium*. URL: https://catalog.ldc.upenn.edu/docs/LDC2013T19/OntoNotes-Release-5.0.pdf.

Weiss, Karl, Taghi M Khoshgoftaar, and DingDing Wang (2016). "A survey of transfer learning". In: *Journal of Big data* 3.1, p. 9.

Wong, Yuk Wah and Raymond J Mooney (2006). "Learning for semantic parsing with statistical machine translation". In: *Proceedings of NAACL*.

Xu, Xiaojun, Chang Liu, and Dawn Song (2017). "Sqlnet: Generating structured queries from natural language without reinforcement learning". In: *arXiv preprint arXiv:1711.04436*.

Xu, Yan et al. (2016). "Improved relation classification by deep recurrent neural networks with data augmentation". In: *Proceedings of COLING*. URL: https://www.aclweb.org/anthology/C16-1138.pdf.

Yadav, Vikas and Steven Bethard (2018). "A Survey on Recent Advances in Named Entity Recognition from Deep Learning models". In: *Proceedings of COLING*. URL: https://www.aclweb.org/anthology/C18-1182.pdf.

Yang, Bishan and Claire Cardie (2012). "Extracting opinion expressions with semi-markov conditional random fields". In: *Proceedings of EMNLP-CoNLL*.

Yang, Fan and Paul Vozila (2014). "Semi-supervised Chinese word segmentation using partial-label learning with conditional random fields". In: *Proceedings of EMNLP*. URL: http://aclweb.org/anthology/D14-1010.

Yang, Yaosheng et al. (2018). "Distantly Supervised NER with Partial Annotation Learning and Reinforcement Learning". In: *Proceedings of COLING*. URL: http://aclweb.org/anthology/C18-1183.

Yang, Zhilin, Ruslan Salakhutdinov, and William W Cohen (2017). "Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks". In: *Proceedings of ICLR*. URL: https://openreview.net/pdf?id=ByxpMd9lx.

Ye, Zhixiu and Zhen-Hua Ling (2018). "Hybrid semi-Markov CRF for Neural Sequence Labeling". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P18-2038.pdf.

Yih, Scott Wen-tau et al. (2015). "Semantic parsing via staged query graph generation: Question answering with knowledge base". In: *Proceedings of ACL-IJCNLP*.

Yin, Pengcheng et al. (2016). "Neural enquirer: Learning to query tables with natural language". In: *Proceedings of NAACL Workshop on Human-Computer Question Answering*.

Yu, Chun-Nam John and Thorsten Joachims (2009). "Learning structural svms with latent variables". In: *Proceedings of ICML*. URL: http://doi.acm.org/10.1145/1553374.1553523.

Yu, Tao et al. (2018a). "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task". In: *Proceedings of EMNLP*. URL: https://www.aclweb.org/anthology/D18-1425.pdf.

Yu, Tao et al. (2018b). "TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation". In: *Proceedings of NAACL*.

Zelle, John M and Raymond J Mooney (1996). "Learning to parse database queries using inductive logic programming". In: *Proceedings of AAAI*.

Zettlemoyer, Luke S and Michael Collins (2005). "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars". In: *Proceedings of UAI*.

Zhang, Yue and Jie Yang (2018). "Chinese NER Using Lattice LSTM". In: *Proceedings of ACL*. URL: https://www.aclweb.org/anthology/P18-1144.

Zhang, Yuhao, Peng Qi, and Christopher D Manning (2018). "Graph Convolution over Pruned Dependency Trees Improves Relation Extraction". In: *Proceedings of EMNLP*. URL: https://aclweb.org/anthology/D18-1244.

Zhong, Victor, Caiming Xiong, and Richard Socher (2017). "Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning". In: *arXiv preprint arXiv:1709.00103*.

Zhou, GuoDong and Jian Su (2002). "Named entity recognition using an HMM-based chunk tagger". In: *Proceedings of ACL*.

Zou, Yanyan and Wei Lu (2018). "Learning Cross-lingual Distributed Logical Representations for Semantic Parsing". In: *Proceedings of ACL*. URL: http://aclweb.org/anthology/P18-2107.