Knowledge-enhanced Neural Grammar Induction

Bowen Li

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2021

Abstract

Natural language is usually presented as a word sequence, but the inherent structure of language is not necessarily sequential. Automatic grammar induction for natural language is a long-standing research topic in the field of computational linguistics and still remains an open problem today. From the perspective of cognitive science, the goal of a grammar induction system is to mimic children: learning a grammar that can generalize to infinitely many utterances by only consuming finite data. With regard to computational linguistics, an automatic grammar induction system could be beneficial for a wide variety of natural language processing (NLP) applications: providing syntactic analysis explicitly for a pipeline or a joint learning system; injecting structural bias implicitly into an end-to-end model.

Typically, approaches to grammar induction only have access to raw text. Due to the huge search space of trees as well as data sparsity and ambiguity issues, grammar induction is a difficult problem. Thanks to the rapid development of neural networks and their capacity of over-parameterization and continuous representation learning, neural models have been recently introduced to grammar induction. Given its large capacity, introducing external knowledge into a neural system is an effective approach in practice, especially for an unsupervised problem. This thesis explores how to incorporate external knowledge into neural grammar induction models. We develop several approaches to combine different types of knowledge with neural grammar induction models on two grammar formalisms — constituency and dependency grammar.

We first investigate how to inject symbolic knowledge, universal linguistic rules, into unsupervised dependency parsing. In contrast to previous state-of-the-art models that utilize time-consuming global inference, we propose a neural transition-based parser using variational inference. Our parser is able to employ rich features and supports inference in linear time for both training and testing. The core component in our parser is posterior regularization, where the posterior distribution of the dependency trees is constrained by the universal linguistic rules. The resulting parser outperforms previous unsupervised transition-based dependency parsers and achieves performance comparable to global inference-based models. Our parser also substantially increases parsing speed over global inference-based models.

Recently, tree structures have been considered as latent variables that are learned through downstream NLP tasks, such as language modeling and natural language inference. More specifically, auxiliary syntax-aware components are embedded into the

neural networks and are trained end-to-end on the downstream tasks. However, such latent tree models either struggle to produce linguistically plausible tree structures, or require an external biased parser to obtain good parsing performance. In the second part of this thesis, we focus on constituency structure and propose to use imitation learning to couple two heterogeneous latent tree models: we transfer the knowledge learned from a continuous latent tree model trained using language modeling to a discrete one, and further fine-tune the discrete model using a natural language inference objective. Through this two-stage training scheme, the discrete latent tree model achieves state-of-the-art unsupervised parsing performance.

The transformer is a newly proposed neural model for NLP. Transformer-based pre-trained language models (PLMs) like BERT have achieved remarkable success on various NLP tasks by training on an enormous corpus using word prediction tasks. Recent studies show that PLMs can learn considerable syntactical knowledge in a syntax-agnostic manner. In the third part of this thesis, we leverage PLMs as a source of external knowledge. We propose a parameter-free approach to select syntax-sensitive self-attention heads from PLMs and perform chart-based unsupervised constituency parsing. In contrast to previous approaches, our head-selection approach only relies on raw text without any annotated development data. Experimental results on both English and eight other languages show that our approach achieves competitive performance.

Acknowledgements

It has been graceful as a winding stream, passionate as fire to live and study in Edinburgh. I would like to express appreciation to everyone who has been with me and offered help to me during this unforgettable journey.

Foremost, I would like to express my sincere appreciation to my principal supervisor Prof. Frank Keller for the ongoing support of my PhD research, for his patience, motivation, enthusiasm, and in-depth knowledge. Frank has given me the freedom to choose research topics and explore possibilities. The feedback and guidance he has provided has been greatly helpful in narrowing down research directions, formulating research questions, developing models, and drafting research papers. Moreover, Dr. Shay Cohen, my second supervisor, also played an essential role in supporting my research with his knowledgeable advice. It was an honour, a pleasure and a fortune for me to work with Frank and Shay.

In addition, I would like to thank my thesis committee members Ivan Titov and Joakim Nivre for their efforts to examine my thesis and viva, and for all of the insightful comments. Special thanks to my collaborators, Yang Liu, Jianpeng Cheng, Lili Mou, Hao Zheng, Taeuk Kim and Reinald Kim Amplayo, for their brilliant research ideas and excellent team spirit. I am grateful to Yun Chen, Meng Zhang at Huawei Noah's Ark Lab for their valuable mentoring during that wonderful internship. Furthermore, I would like to recognize the support provided by the China Scholarship Council with the PhD fellowship.

I consider myself privileged and fortunate to be a part of ILCC in Edinburgh. I would like to thank all the group members as well as all the faculty who provided feedback on my PhD projects, presentations and research papers. Special thanks to Yang Liu, Li Dong and Jianpeng Cheng for their valuable advice on NLP research. I want to also thank all the Edinburgh Informatics members, Akash, Bailin, Biao, Cole, Chao, Chaoyun, Chunchuan, Hao, He, Jiangming, Kai, Kunkun, Matt, Shashi, Sicong, Spandana, Xinchi, Yanpeng, Yumo, etc. I want to specially thank my office mates, Yang Liu, Kai Xu, Yumo Xu, Joachim Fainberg, Marco Damonte, Sam Ribeiro and Parag Jain. We created a productive and pleasant working environment together, and those daily discussions inspired me a lot.

Research isn't the only part of doctoral life. I would like to thank He Zhang, Kai Xu, Yumo Xu, Yang Liu, Cole Hurwitz for all the fun times. Many thanks to Yingxuan Cui, Yuanjun Laili, Pingchuan Ma, Shiwen Ma, Huiyuan Xie, Yifan Mai, Zhiqiang Zhang, for making my four-year time enjoyable.

Lastly, I want to express my deepest gratitude to all of my family members. I would like to thank my parents for their unconditional love and support throughout my studies. Special thanks to my grandpa who has supported me since I was a young child, and my cousin, who has always been there for me.

Will there be a day when time will rewind?
Rewinding back to the good old days we can never return.
Ultimately, there will be a day when we will become yesterday.
It's you who accompanied me in my one and only life.

—— *Mayday*

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Bowen Li*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Natural language demonstrates a simple sequential format, but what is the underlying structure that governs the surface utterances? This question has fascinated people for over two thousand years. The earliest known attempts to describe a language in a systematic way originated in ancient India, commonly dated to the 5th century BCE, where the desire for a faithful transmission of the sacred scriptures known as the Vedas brought about the need to describe Sanskrit (McGee and Warms, 2013). Since then, grammars of natural languages have been studied and complied for the purposes of education and cultural transmission.

Noam Chomsky (Chomsky, 1965) defines a grammar as:

> *A fully adequate grammar must assign to each of an infinite range of sentences a structural description indicating how this sentence is understood by the ideal speaker-hearer.*

From this description, we can deduce two prominent characteristics of a grammar:

1. Being abstractive: a grammar regulates the inherent structures of sentences from a natural language;

2. Being assistive: a grammar helps the understanding of a natural language.

Formally speaking, in linguistics, a grammar is a finite set of structural rules managing the composition of clauses, phrases and words in a natural language. At the same time, a grammar should be able to generalize to the infinite set of sentences from the language. *Grammar induction*, which is learning formal grammars automatically from

finite language data, is a long-standing and important question in cognitive science and psycholinguistics. A learned grammar is able to provide syntactical analysis on unseen sentences; in this sense, grammar induction can serve as the basis for *unsupervised parsing*. On the one hand, from a psycholinguistic point of view, a grammar supports the precise transmission of information in the context of speaker-hearer communication. On the other hand, from the perspective of computational linguistics, syntactical analysis, the product of grammar modeling, assists machines to automatically process natural language data.

Throughout the long history of natural language processing (NLP), syntactical analysis has played an essential role. In the early days, syntactical analysis was used as a low-level component to help build high quality pipeline systems for complex downstream NLP tasks, such as semantic role labelling (Johansson and Nugues, 2008), question answering (Hovy et al., 2000), machine translation (Yamada and Knight, 2001). However, pipeline based NLP systems easily cause problems like inconsistent annotations and error propagation. These problems have led to joint systems, where syntactical parsing models are investigated to be jointly trained with other components. Recently, neural networks based models have developed rapidly and gradually become the dominant approaches in the NLP research. To bring syntax oriented inductive bias to the learning process of neural NLP models, syntactical analysis has been employed in various NLP tasks (Socher et al., 2013; Aharoni and Goldberg, 2017).

Normally, tree structures are either provided together with the dataset or produced by an existing parser that is trained with supervision. However, an annotated dataset is extremely expensive to obtain because it requires specialized expertise and years of laborious efforts. Regarding supervised parsers, they also suffer from drawbacks: for most low resource languages, no annotated data or even no annotation scheme exists; supervised parsers are often trained on corpora from limited domains, so they face an out-of-domain issue. Consequently, deriving tree structures directly from raw text is highly desirable. Besides, from a psycholinguistic point of view, success in grammar induction or unsupervised parsing also provides empirical evidence against innate grammar arguments such as “poverty of the stimulus” and universal grammar (White and White, 2003).

We note that grammar induction and unsupervised parsing are not synonymous. More details will be discussed in Section 2.1.2. In this thesis, from a utilitarian point of view, we focus more on the problem of unsupervised parsing for both constituency and dependency representations. Additionally, we induce probabilistic formal grammars

for constituency representations to investigate their linguistic properties.

1.2 Challenges

Typically, grammar induction or unsupervised parsing systems, only have access to raw text. Although some systems will take advantage of word cluster information such as part-of-speech (POS) annotations, the information available is still severely limited, so that grammar induction is considered a difficult problem. Normally, generative grammars are used for grammar induction models, which model the joint probability of both parse trees and sentences. They are trained by directly optimizing the marginal log likelihood of sentences with the expectation maximisation (EM) algorithm. Following this research line, early attempts were largely unsuccessful. The reasons for the discouraging results are manifold: the ill-posed optimization landscape, overly strict independence assumptions, the fragile optimization algorithm (EM is sensitive to initialization and easy to be stuck in local optima). Given the difficulties, follow-up approaches sought to make improvements by using auxiliary targets (Klein and Manning, 2004), Bayesian priors (Johnson et al., 2007), manually engineered features (Headden III et al., 2009), external knowledge (Mareček and Straka, 2013), posterior regularization (Naseem et al., 2010). Recent progress in the development of neural networks has triggered interests in designing neural models for grammar induction. Thanks to their capability of continuous representation learning and over-parameterizaiton, neural approaches have indeed brought improvement to this problem (Jiang et al., 2016; Kim et al., 2019a).

Given the fruitful prior research of incorporating constraints and knowledge into traditional grammar induction systems, it is reasonable to conjecture that it could be effective to do so for neural systems. More specifically, the over-parameterization of neural models eases the optimization (Arora et al., 2018) and distributed representations smooth the probabilities of correlated elements in the grammar (Jiang et al., 2016). External knowledge and constraints provide complementary regularization on the learned grammar, which is potentially compatible with neural approaches. However, only limited work has explored how to incorporate external knowledge and constraints into neural systems.

Although learning a generative grammar via optimizing the marginal log likelihood of sentences has been a main stream approach so far, researchers have explored alternative objectives for grammar induction, such as contrastive estimation (Smith and

Eisner, 2005), search-based structure prediction (Daumé III, 2009), convex formulation (Grave and Elhadad, 2015), and so no. Recently, unsupervised parsing has been formalized as a latent variable learning problem, where structure-sensitive components are embedded into neural networks and are trained in an end-to-end fashion. Under this formalism, the exploration of alternative objectives for unsupervised structure learning has been broadened to a wide range of downstream NLP tasks, such as language modeling (Shen et al., 2018b) and natural language understanding tasks like sentiment analysis and natural language inference (Maillard et al., 2017; Choi et al., 2018). In this way, external knowledge can be acquired from a variety of downstream tasks for unsupervised parsing. These *latent tree* models either model the tree structures explicitly where gradients are backpropagated through discrete structures via exact marginalization or gradient approximation, or model them implicitly where syntactical features are employed and the entire model is differentiable. Although these models bring benefits to downstream tasks, the learned tree structures do not always resemble human annotated trees and are not consistent across random restarts (Williams et al., 2018a). An algorithm coupling heterogeneous models and different learning objectives could have a chance to learn more linguistically plausible and more steady parsing strategies.

Transformer-based pre-trained language models (PLMs) like BERT (Devlin et al., 2019) have achieved the state of the art in many NLP tasks. They changed the paradigm in the research of natural language understanding: pre-training and fine-tuning has become the dominant approach. Thanks to the Transformer's superior capability of parallelism, PLMs can be efficiently trained on enormous raw text (e.g., common crawl of the internet) containing billions of tokens. Surprisingly, although PLMs contain no syntax-aware components, studies show that PLMs rediscover the classic NLP pipeline (Tenney et al., 2019a) and learn considerable syntactical knowledge (Goldberg, 2019; Liu et al., 2019a). Interestingly, PLMs show that it is possible to learn syntactical knowledge by training on massive data in a structure-agnostic manner. Therefore, it is feasible to employ PLMs as a source of external knowledge for syntactical structure learning. A challenging problem is how to design an effective and reliable algorithm to extract syntax related features from PLMs and perform unsupervised parsing and grammar induction.

1.3 Thesis Overview

In this thesis, we investigate existing problems in grammar induction and unsupervised parsing, and propose novel approaches by taking advantage of neural networks and external knowledge to address the challenges described in the previous section. This thesis will study both constituency and dependency structures utilizing knowledge from symbolic rules, auxiliary objectives and pre-trained language models (PLMs).

Chapter 2

We present background knowledge on grammar induction and neural networks in Chapter 2. We first introduce the motivation, problem formulation, related work and common experimental setup on grammar induction and unsupervised parsing for two formalisms, constituency and dependency. We then review neural networks typically relevant to our work in this thesis, which include recurrent neural networks, recursive neural networks, the Transformers and Transformer-based pre-trained language models.

Chapter 3

In Chapter 3, we study the problem of unsupervised dependency parsing. Previous state-of-the-art models, including generative and discriminative ones, all rely on global exact inference, which is implemented by dynamic programming with $O(n^3)$ run time. For the generative models, probabilistic dependency models like dependency models with valence (DMV; Klein and Manning 2004) are always used as the backbone model (Jiang et al., 2016; Han et al., 2019). While for the discriminative models, Cai et al. (2017) used a conditional random field (CRF) parser. Besides, transition-based models enable faster inference with $O(n)$ run time for both training and test. Although transition-based models have been shown to perform well in supervised parsing (Kiperwasser and Goldberg, 2016), their performance on unsupervised parsing still lags behind their global inference-based counterparts.

In this chapter, we use an autoencoder to integrate discriminative and generative transition-based parsers, dependency variants of recurrent neural network grammars (RNNGs; Dyer et al. 2016), yielding a reconstruction process with parse trees as latent variables. To further introduce regularization, we augment the model with posterior regularization (Ganchev et al., 2010), which allows us to seamlessly integrate linguistic

knowledge in the shape of symbolic linguistic rules and still maintain the efficiency of transition-based systems. Furthermore, we propose a novel variance reduction method to stabilize neural variational inference with discrete latent variables. This leads to better parsing performance on English and eight other languages for transition-based systems while also achieving superior parsing speed.

Chapter 4

In Chapter 4, we study the latent tree models for unsupervised constituency parsing. Recent work has explored the idea of leveraging feasible downstream NLP tasks to collect clues for structure learning. Models that follow this research line have investigated downstream NLP tasks like natural language understanding (NLU) tasks and language modeling. Generally, they harness neural networks and treat the tree structures as latent variables. Regarding the latent tree structures, two formalisms are often assumed. The first one is *hard* as it explicitly models the discrete tree structures in the neural networks mainly for NLU tasks, where dynamic programming based exact marginalization or gradient approximation is used for backpropagation. While the second one is *soft* as it implicitly employs the syntactical features in the neural networks mainly for language modeling, where tree structures are extracted via an *external* parsing method at a post-processing phase.

In practice, discrete hard models trained on NLU tasks perform poorly on parsing and show low self-agreement with random initialization (Williams et al., 2018a). Continuous soft models trained on language modeling succeed to produce syntactically plausible structures (Htut et al., 2018), but it has been pointed out that the employed external parser is incomplete in theory and biased to English (Dyer et al., 2019). To mitigate this problem, we propose an imitation learning approach that combines a continuous soft model (i.e., PRPN; Shen et al. 2018b) with a discrete hard model (i.e., Tree-LSTM), both trained without access to gold standard parse trees. We exploit the advantages of the PRPN (supports backpropagation) by transferring its knowledge to a discrete parser, which explicitly models tree-building operations. We accomplish the knowledge transfer by training the discrete parser to imitate the behavior of the PRPN. Then the discrete parser refines its policy by solely trained on a natural language inference task. Our approach effectively improves the parsing performance of the discrete parser and makes the learning reliable, as shown by the improved self-agreement.

Chapter 5

In Chapter 5, we study the pre-trained language models (PLMs) for constituency grammar induction. Recent progress on Transformer-based PLMs like BERT (Devlin et al., 2019) shows that PLMs trained on large-scale corpus, containing billions of tokens, can bring significant improvements on downstream NLP tasks. Surprisingly, although PLMs neither are trained on syntax-oriented objectives nor contain syntax-aware components, studies show that PLMs have learned considerable syntactical knowledge (Goldberg, 2019; Liu et al., 2019a). This finding intrigues an interesting problem: how much syntactical knowledge PLMs have learned and is it possible to leverage PLMs to do grammar induction?

Existing syntactical analysis on PLMs are limited: hand-crafted test suites (Goldberg, 2019) require the laborious compilation of language- and construction-specific suites of sentences; general probes (also known as diagnostic classifiers; Belinkov and Glass 2019) need to carefully set the extraction experiments to adequately reflect differences in representations (Hewitt and Liang, 2019; Voita and Titov, 2020); specifically designed structural probes (Hewitt and Manning, 2019) lack justification in terms of the evaluation metric (Hall Maudslay et al., 2020). It is therefore natural to use an unsupervised parsing task to test whether PLMs have learned syntactical knowledge. Previous studies (Kim et al., 2020a,b) have tried top-down and chart-based parsing algorithms, but they crucially rely on an annotated development set for feature selection. In this chapter, we propose a novel approach to build a PLM-based unsupervised parser without requiring an annotated development set: we rank Transformer heads based on their inherent properties, such as how likely tokens are to be grouped in a hierarchical structure. We then ensemble the top-K heads to produce constituency trees. On English and eight other languages, our approach yields competitive parsing performance. Moreover, we learn neural probabilistic context-free grammars (PCFGs) from the trees induced from PLMs using our approach.

Chapter 6

Chapter 6 concludes the thesis and discusses directions for future work.

Contributions

The main contributions of this thesis are:

1. A new approach to unsupervised dependency parsing induction that combines generative and discriminative transition-based dependency parsers using variational inference, posterior regularization and variance reduction techniques.

2. A novel imitation learning approach to couple both continuous and discrete neural latent tree models through knowledge transfer and further policy refinement.

3. A ranking-based approach to build a PLM-based unsupervised parser without requiring an annotated development set, which fulfills both constituency grammar induction and interpretability study on PLMs.

1.4 Published Work

The contributions presented in this thesis are published in the following papers.

Chapter 3 was presented as:

> Li, B., Cheng, J., Liu, Y., and Keller, F. 2019. Dependency grammar induction with a neural variational transition-based parser. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Chapter 4 was presented as:

> Li, B., Mou, L., and Keller, F. 2019. An imitation learning approach to unsupervised parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Chapter 5 was presented as:

> Li, B., Kim, T., Amplayo, R. K., and Keller, F. 2020. Heads-up! Unsupervised constituency parsing via self-attention heads. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*.

Part of Chapter 5 was presented as an unpublished work:

> Kim, T., Li, B., and Lee, S. 2020. Chart-based zero-shot constituency parsing on multiple languages. arXiv:2004.13805v2.

Chapter 2

Background

2.1 Grammar Induction

2.1.1 Motivation

How children learn their first language by merely listening to the speech in the environment has been a long-standing and important question in the fields of cognitive science and psycholinguistics. One famous argument from Noam Chomsky (Chomsky et al., 2006), the *Poverty of the Stimulus* argument, suggests that children are not exposed to rich enough language data in their surroundings to acquire every feature of their language. The claim is that the utterances children hear during learning do not contain enough information to develop a thorough understanding of the grammar of a language, which is considered contrary to the empiricist idea that language is learned solely through experience. It is closely related to *Plato's Problem*, that knowledge of geometry concepts was unearthed from a slave who was never explicitly taught them. However, the idea that the Poverty of the Stimulus supports the innateness hypothesis remains controversial. Cowie et al. (1999) argues that the Poverty of the Stimulus fails on both empirical and conceptual grounds to support innateness.

Stemming from the the poverty of the stimulus argument and the existence of some universal properties of human languages, *Universal Grammar* (UG; White and White 2003; Chomsky 2018) has been developed, usually credited to Noam Chomsky. The core notion of UG is that a certain set of structural rules are innate to humans, independent of sensory experience. However, some linguists have argued that languages are so diverse that such universality is rare (Evans and Levinson, 2009). UG has also been refuted by abundant variation at all levels of linguistic organization (Hinzen, 2012). In

addition to that, this criticism on the Poverty of the Stimulus argument as well as UG is further supported by the success of automatic grammar induction system in learning hierarchical structures from finite language data.

In early NLP research, syntactical analysis always functions as an essential low-level component to help build high quality systems for complex downstream NLP tasks. Within such pipeline-based systems, features or parse trees produced by a syntactical parsing system are fed into the subsequent components for a variety of NLP tasks, such as speech disfluency correction (Johnson and Charniak, 2004), recognizing textual entailment (Finkel et al., 2006), semantic role labelling (Johansson and Nugues, 2008), question answering (Hovy et al., 2000), machine translation (Yamada and Knight, 2001). However, pipeline based NLP systems run several processors over the data, which easily causes problems such as inconsistent annotations and error propagation. To this end, syntactical parsing has been investigated to be jointly trained with tasks like language modeling (Chen, 1995), name entity recognition (Finkel and Manning, 2009), semantic role labelling (Li et al., 2010). In recent years, neural networks based models have developed rapidly and become the dominant backbone approaches in the NLP research. To bring syntax oriented inductive bias and regularities to the learning process of neural NLP models, syntactical analysis has been employed to various NLP tasks, such as sentiment analysis (Socher et al., 2013), text entailment (Bowman et al., 2016), neural machine translation (Aharoni and Goldberg, 2017), and so on.

Typically, such studies assume that tree structures are either provided together with the dataset or produced by an off-the-shelf syntactic parser. However, an annotated dataset is extremely expensive to obtain as it requires syntactical expertise and years to annotate a good-sized dataset. On the other hand, supervised parsers are limited for several reasons: for most low resource languages, no annotated data is available; supervised parsers are often trained on a newswire corpus, so they face an out-of-domain issue. As a natural solution, deriving tree structures directly from raw text data is therefore highly motivated. Recently, researchers have started to explore how to induce syntactical tree structures via neural network-based models from downstream NLP tasks such as language modeling (Shen et al., 2018b, 2019b; Wang et al., 2019), and sentence understanding tasks like sentiment analysis and natural language inference (Yogatama et al., 2017; Maillard et al., 2017; Choi et al., 2018).

2.1.2 Problem Formulation

The long and substantial exploration by researchers in studying the syntax of natural language indicates that natural language sentences can be analyzed in the form of tree-like structures. The two most popular structures are constituency and dependency structures. They capture different aspects of the syntax of a language and therefore each has validity in its own terms.

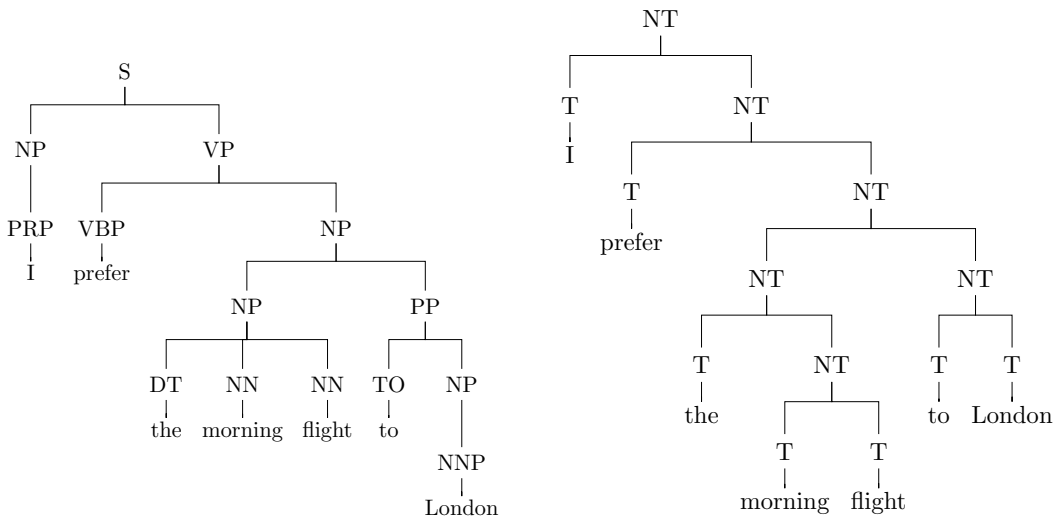
Constituency Grammar

The core notion of the constituency grammar is abstraction, groups of words behaving as single units or constituents, such as noun phrases (NPs) and verb phrases (VPs). The constituency tree identifies the constituent phrases in a given sentence and encodes the order in which the tree is derived. Figure 2.1a shows the constituency tree of a sample sentence. Detailed annotations will be discussed later.

Identifying phrase-structured grammars from surface text is a classical problem in computational linguistics, which can be traced back to 1950s. In Chomsky (1956), Noam Chomsky found that no finite-state Markov process that produces symbols with state transition can serve as an English grammar and furthermore formalized the notion of *phrase structure*. Based on this concept, a set of grammatical transformations are specified to rewrite sentences with phrase structure into new sentences with derived phrase structure so that all sentences are constructed by repeated transformations. This laid the foundation of the well-known *context-free grammars* (CFGs) for natural language analysis. Lamb (1961) used the *distributional analysis* of Harris (1951) and Hockett (1958), where the syntax is completely described by a list of distribution classes of items and a list of constructions. Namely, a construction is characterized by (1) the distribution classes which enter into it and their relative order, (2) the distribution class membership of the constitutes. Solomonoff (1964) employed the context-free grammar to deal with the extrapolation of sets of strings, in which the constraints among the symbols are like those that exist among the words of sentences in European languages. Horning (1969) introduced *probabilistic* grammars and demonstrated that it is possible to learn such grammars from positive examples.

Next, we introduce the formal definition of a context-free grammar. Typically, A context-free grammar G is defined by the 4-tuple $G = (V, \Sigma, R, S)$ (Sipser, 1996; Hopcroft et al., 2001):

- V is a finite set. Each element $v \in V$ is called a non-terminal or a variable.



(a) A fully annotated constituency tree.

(b) A constituency tree w/o tag labels.

Figure 2.1: Constituency tree examples. (a): a fully annotated constituency tree, where nonterminal and preterminal (part-of-speech) tags are labeled. (b): a constituency tree, where the parse tree is binarized using right-branching and nonterminal (NT) and preterminal (T) tags are left anonymized.

Each variable represents a different type of phrase or clause in the sentence to distinguish syntactical categories.

- Σ is a finite set of terminals (i.e., surface words), different from V, which make up the actual content of the sentence. Σ is actually the alphabet of the language defined by the grammar G.

- S is the start variable (or start symbol), used to represent the whole sentence. It is an element of V.

- R is a finite relation in $V \times (V \cup \Sigma)^*$, where the asterisk indicates the Kleene star operation. The members of R are production (or rewrite) rules of the grammar. Specifically, if G is in Chomsky normal form (Chomsky, 1959), R takes the following form:

$$\begin{aligned} S &\rightarrow A, & A &\in V \\ A &\rightarrow B\,C, & A, B, C &\in V \\ T &\rightarrow w, & T &\in V, \quad w \in \Sigma. \end{aligned}$$

In one specific formalism where the set V is further split into N(non-terminals)

and P (pre-terminals, e.g., POS; Part-of-Speech tags), R takes the form:

$$\begin{aligned} &S \rightarrow A, && A \in N \\ &A \rightarrow B\,C, && A \in N, \quad B,C \in N \cup P \\ &T \rightarrow w, && T \in P, \quad w \in \Sigma. \end{aligned}$$

In a probabilistic CFG (PCFG), each production rule will be assigned a probability.

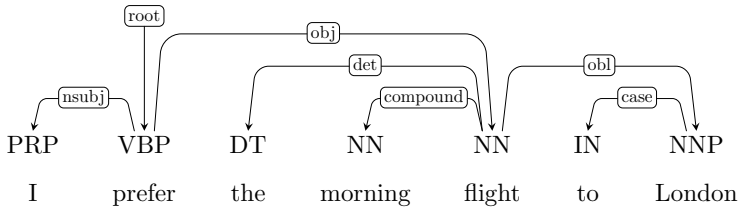
Figure 2.1a is the parsed constituency structure of a sample sentence. The production rules applied in the derivation are $(S \rightarrow NP,VP)$, $(VP \rightarrow VBP,NP)$, $(NP \rightarrow DT,NN,NN)$, $(NP \rightarrow PRP)$, $(NP \rightarrow NNP)$, etc. In this example, preterminals (POS tags) are distinguished from the nonterminals. Note that, the CFG underlying this constituency tree does not take the Chomsky normal form given the production rule $(NP \rightarrow DT,NN,NN)$. But this grammar can be converted to an equivalent Chomsky normal form by utilizing some rules like binarization.

Normally, the goal of grammar induction for constituency grammars is to identify the CFG solely from raw text. The learned grammar can then be employed to parse a given sentence through a parsing algorithm and yield the corresponding constituency tree structure. Since there is no supervision provided for nonterminal and preterminal tags, the learned tags will be anonymized. For convenience, the target CFG can be specified to take the Chomsky normal form. An example is presented in Figure 2.1b.

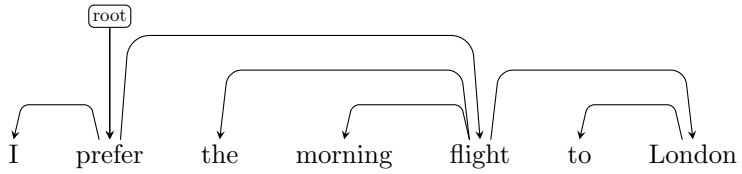
Dependency Grammar

As opposed to the constituency relation of phrase structure, dependency grammar is a class of grammatical theories based on the dependency relation. Dependency is the notion that the syntactic structure of a sentence consists of binary asymmetrical relations between the words of the sentence. Concretely, basic linguistic units (i.e., words in a sentence), are connected to each other by directed links. The (finite) verb is taken to be the center of clause structure. All other units are directly or indirectly connected to the verb through the directed links, which are called dependencies. A dependency structure (or representation) is determined by the relation between a word (i.e., a head) and its dependents. In the phrase structure, for every word in a sentence, there is one or more nodes in the constituency tree structure that correspond to that word. Regarding the phrasal constituents, although dependency representations acknowledge phrases, they lack phrase nodes in the tree structure. Even though dependency representations can model an *ordered* tree structures, they often abstract away from linear word order

and just concentrate on the hierarchical order, which means they do not encode actual word order. As a result, dependency representations are well suited for the analysis of languages with free word order, such as Czech. Furthermore, given their flexibility, dependency representations have the potential to achieve cross-linguistically consistency and facilitate multilingual language processing. For example, Universal Dependencies (UD; Nivre et al. 2016, 2020) is a open source project aiming to develop cross-linguistically consistent treebank annotations for many languages to capture similarities as well as characteristics among topologically different languages.



(a) A dependency tree w/ labeled relations.



(b) A dependency tree w/o labeled relations.

Figure 2.2: Dependency tree examples. (a): a fully annotated dependency tree, where POS tags and dependency relations are labeled. (b): a dependency tree without any further annotations.

Normally, in a dependency treebank, dependency relations are also labeled in the dependency tree structures as presented in Figure 2.2a. For instance, `nsubj` indicates nominal subject and `obj` indicates object. [1] Such dependency relation categories can be learned in a supervised learning setting. Regarding unsupervised dependency parsing or dependency grammar induction, the aim is to identify the unlabeled pair-wise relations among words in a given sentence, which means relation labels are omitted (illustrated in Figure 2.2b).

The seminal work of Tesnière (Tesnière, 1959) is usually considered as the starting point of the modern theoretical tradition of dependency grammar. This tradition comprises a diverse family of grammatical theories and formalisms (Nivre, 2005). Be-

[1] More details can be found in https://universaldependencies.org/u/dep/index.html

sides the theory of structural syntax developed by Tesnière (1959), there are also Word Grammar (Hudson, 1984, 2007, 2010), Functional Generative Description (Sgall et al., 1986), Meaning-Text Theory (Mel'cuk et al., 1988), Constraint Dependency Grammar (Maruyama, 1990; Harper and Helzerman, 1995; Menzel and Schroder, 1998), Constraint Grammar (Karlsson, 1990; Karlsson et al., 1995), and so on. These dependency grammar formalisms share common core of assumptions, centered upon the notion of dependency (see above), while they also diverge on some points such as the issue of projective and non-projective representations.

The earliest work on parsing with dependency representations was tied to formalizations of dependency grammar that were very close to context-free grammar, as described in Hays (1964) and Gaifman (1965). An argument has been developed that dependency grammar is only a restricted variant of context-free grammar (Jarvinen and Tapanainen, 1998). However, this argument is erroneous because the results only concern the specific version of dependency grammar formalized by Hays (1964) and Gaifman (1965), which for one thing is restricted to projective dependency structures. The close relation to context-free grammar in the formalization of dependency grammar by Hays (1964) and Gaifman (1965) indicates that essentially the same parsing methods can be employed for both types of system. For instance, the parsing algorithm outlined in Hays (1964) is a bottom-up dynamic programming algorithm resembles the CKY algorithm (Cocke, 1969; Kasami, 1966; Younger, 1967) proposed for context-free constituency parsing. [2] A common characteristic of all frameworks that implement dependency parsing as a form of lexicalized context-free parsing is that they are restricted to the derivation of projective dependency representations, although some of them allow post-processing to incorporate non-projective representations (Sleator and Temperley, 1993).

Grammar Induction vs. Unsupervised Parsing

Parsing is the computational implementation of syntactic analysis based on different syntactic representations (e.g., constituency or dependency). From the machine learning perspective, *unsupervised parsing* particularly denotes learning a parsing model in an unsupervised manner. On the other hand, *grammar induction* refers to learning a formal grammar, a set of production rules (and associated probabilities in the context of a probabilistic grammar) that gives a systematic account of how natural language

[2] More examples can be found in the survey (Nivre, 2005).

is generated. Once learned, such a grammar can be evaluated on *unsupervised parsing*. More specifically, parsing conceptually consists of the derivation of all analyses that are permissible according to the learned grammar and the selection of the most probable analysis according to the probabilistic model. Then the predicted parse tree is compared against the expert-annotated one. In this sense, it is worth noting that grammar induction and unsupervised parsing are not synonymous.

Regarding the constituency structures, inducing a probabilistic context-free grammar is a classic research problem and has been investigated for over thirty years. There is also substantial research work only focusing on the unsupervised parsing without considering the underlying formal grammar (Klein and Manning, 2002; Seginer, 2007; Shen et al., 2018b, 2019a; Wang et al., 2019; Drozdov et al., 2019, 2020). In this thesis, we consider unsupervised constituency parsing in Chapter 4, both unsupervised constituency parsing and grammar induction in Chapter 5.

Unlike theories and parsers based on constituency analysis, theoretical frameworks and parsers are often rather less connected for dependency-based analysis. Early attempts on unsupervised dependency parsing were grammar-driven where a formal dependency grammar was relied on and a probabilistic model was induced (e.g., Carroll and Charniak 1992). On the contrary, modern unsupervised dependency parsing methods, especially those based on the DMV model (Klein and Manning, 2004), are purely generative models without involving a formal grammar. In this thesis, we focus on the problem of unsupervised dependency parsing while consider no underlying formal grammars in Chapter 3.

2.1.3 Related Work

In this section, we briefly review the early attempts that have been made to address the problem of grammar induction as well as unsupervised parsing for both constituency and dependency grammars. More recent work, including neural network-based approaches, will be reviewed comprehensively in the related work sections of the following three chapters.

Constituency Structures

Unfortunately, initial results on constituency grammar induction were mostly negative. Gold (1967) investigated *language learnability* via learning to identify an unknown language by accessing positive examples alone. In this work, the learning algorithm

is constrained such that there is some finite time after which the predictions will all be the same and correct (*identification in the limit*). Under this constraint, the author showed that it is not possible to learn even regular grammars. The *constructive* procedure is one grammar induction procedure (Solomonoff, 1959; Feldman, 1967; Feldman et al., 1969), which is the systematic use of sample strings to construct the rules of the grammar. But constructive methods are incomplete in theory and only support smaller grammar classes in practice. On the contrary, *enumerative* methods (Gold, 1967; Feldman et al., 1969; Horning, 1969) are advantageous on these problems; they enumerate the class of grammars under consideration, examine each grammar in turn, and select the first grammar which is appropriate for the sample of strings from the unidentified language. Horning (1969) showed that it is possible to learn probabilistic grammars from positive examples alone using enumerative methods. However, enumerative methods typically require unacceptably large amounts of computation.

Regarding probabilistic context-free grammars (PCFGs), it is generally required to specify a probabilistic grammar (e.g., formalism) and fit its parameters through optimization. Lari and Young (1990) first empirically showed the possibility of statistical induction of PCFGs using the EM algorithm, especially with the inside-outside algorithm (Baker, 1979). However, prior work found it hard to induce plausible grammars from the natural language data, such as directly optimizing the log likelihood with the EM algorithm (Carroll and Charniak, 1992; Charniak, 1996). Two major reasons for the failure are the ill-behaved optimization landscape and the strict independence assumptions of PCFGs. Therefore, follow-up methods to grammar induction have resorted to Variational Bayesian Inference (Kurihara and Sato, 2006; Wang and Blunsom, 2013), Markov Chain Monte Carlo (Johnson et al., 2007), non-parametric Bayesian model (Liang et al., 2007) and hand-crafted features (Huang et al., 2012; Golland et al., 2012) to encourage the desired constituency structures to emerge. Recent success on constituency grammar induction is attributed to neural parameterization (Kim et al., 2019a; Zhu et al., 2020) and detailed discussions can be found in Section 5.1.1.

There is also prior work on learning an unsupervised parser without considering an underlying grammar. Klein and Manning (2002) presented a simple generative model combining the benefits of EM-based parameter search and distributional clustering methods. Bod (2006) proposed an unsupervised data-oriented-parsing model which assigns all possible binary trees to a set of sentences and then use all subtrees from these binary trees to predict the most probable parse trees. Seginer (2007) adopted an incremental setting for unsupervised parsing by utilizing a representation for syntactic

structure similar to dependency links. More recently, researchers have explored the induction of constituency structures from neural network-based models via various tasks, such as language modeling, natural language inference, constituency tests (Shen et al., 2018b, 2019a; Williams et al., 2018a; Wang et al., 2019; Cao et al., 2020). A comprehensive review can be found in Section 5.1.

Dependency Structures

Similar to constituency grammars, early efforts on dependency grammar induction have been discouraging. Carroll and Charniak (1992) presented a set of experiments trying to induce probabilistic dependency grammar based on the inside-outside algorithm (Baker, 1979; Lari and Young, 1990). They began with all the rules that were applicable for the sentences in the corpus and iteratively filtered them based on the re-estimated probabilities. However, the proposed method was tested on small artificial languages and only worked when the grammar was fairly restricted. DMV (Klein and Manning, 2004) was the first model that outperformed the trivial right-branching baseline. The EM and the inside-outside algorithms are generally used to learn the DMV model. For inference, dynamic programming (DP) is employed to find the optimal dependency structure.

Since DMV's first breakthrough, there has been a lot of follow-up work. Smith and Eisner (2005) investigated using contrastive estimation to estimate the DMV. In the Bayesian framework, researchers have studied Bayesian priors (Headden III et al., 2009; Cohen and Smith, 2009), Bayesian non-parametric models (Blunsom and Cohn, 2010) and posterior regularization (Naseem et al., 2010). Mareček and Straka (2013) exploited prior knowledge of STOP-probabilities obtained from a large raw corpus. Spitkovsky et al. (2013) proposed to switch between different objectives to break out of local optima. To enrich the expressiveness of the DMV, researchers introduced parameter tying (Cohen and Smith, 2009; Headden III et al., 2009), tree substitution grammars (Blunsom and Cohn, 2010), lexicalization and rich context features (Headden III et al., 2009).

There are also alternative approaches to unsupervised dependency parsing that are not based on the DMV. Daumé III (2009) proposed a stochastic search based method to do unsupervised transition-based parsing. Rasooli and Faili (2012) proposed a transition-based unsupervised dependency parsing model together with *baby-step* training (Spitkovsky et al., 2010) to improve parsing accuracy. Le and Zuidema (2015) presented a self-training approach that started with trees generated by an unsu-

pervised parser and iteratively improved these trees using the richer probability models used in supervised parsing. Inspired by discriminative clustering, Grave and Elhadad (2015) formulated the unsupervised dependency parsing problem as convex optimization of both the model parameters and the parses of training sentences.

More recently, researchers have introduced neural network-based approaches to unsupervised dependency parsing on both discriminative and generative modeling (Jiang et al., 2017; Nivre et al., 2016; Han et al., 2019). A comprehensive review can be found in Section 3.1.

2.1.4 Experimental Setup

Corpora

Most research work in grammar induction has focused on inducing grammars for English. ATIS and WSJ are the two datasets typically used in the experiments for evaluating the performance of grammar induction approaches. ATIS is a corpus from the air traffic information system containing short sentences concerning the same domain and includes topics like reservation; ATIS is less used recently in the evaluation of grammar induction. WSJ (Wall Street Journal) is a corpus pertaining to news domain about business and politics and is the most employed benchmark to date. Both ATIS and WSJ have phrase structure annotations as part of the Penn Treebank (PTB; Marcus et al. 1993) [3] so that the annotations can be directly compared against the induced ones for evaluation. Regarding the dependency structures, the phrase-structure annotations need to be converted to dependency structures using head-selection techniques (Collins, 2003; Yamada and Matsumoto, 2003; Johansson and Nugues, 2007; Surdeanu et al., 2008). [4] For an unannotated corpus, an off-the-shelf supervised parser can be utilized, such as Stanford PCFG parser (Klein and Manning, 2003) and Stanford neural-network dependency parser (Chen and Manning, 2014); the produced silver parse trees are then used for evaluation.

Apart from English, multilingual corpora have been developed to evaluate grammar induction systems on other languages. SPMRL (Seddah et al., 2013, 2014) is a multilingual corpus for a shared task on statistical parsing of morphologically rich languages. It features data sets from 9 languages, each available both in constituency

[3] In the context of grammar induction, PTB and WSJ are sometimes synonymous in some recent studies.

[4] E.g., an open source treebank converter: LTH conversion tool `http://nlp.cs.lth.se/software/treebank-converter/`

and dependency annotation, and additional unannotated data. The PASCAL dataset (Gelling et al., 2012) is for a competition specially on dependency grammar induction, which makes use of a 10 different treebanks annotated in a range of different linguistic formalisms and covering 9 languages. In recent years, many researchers have contributed to the Universal Dependencies project (UD; Nivre et al. 2016, 2020), a collection of treebanks for many languages (183 treebanks representing 104 languages in its current version 2.7) [5], where the morphological and dependency annotation styles are unified across the languages.

All forementioned corpora are equipped with annotations of POS tags. Normally, raw text and the annotated POS tags are available for grammar induction systems. Previous studies employ either of them or both of them. Some approaches also induce POS tags (word classes) using some word clustering tools so that they only require the raw text without any further annotations.

Evaluation Metrics

For constituency grammar induction and unsupervised parsing, the ParsEval metrics (Black et al., 1991) are evaluation metrics for phrase-structure parse trees. Despite various drawbacks, they are the de-facto standard for system comparison on phrase-structure parsing. Assume G and H are phrase-structure gold and hypothesized trees respectively. Each of them is represented by a set of tuples (i, j) where i and j are starting and ending indices of a constituent span. In a supervised parsing setting, the tuple includes an additional constituent label. The ParsEval scores are defined as the accuracy of the hypothesis in terms of the normalized size of the intersection of the constituent sets as follows

$$\begin{aligned} precision &= \frac{G \cap H}{|H|} \\ recall &= \frac{G \cap H}{|G|} \\ F_1 &= \frac{2 \times P \times R}{P + R}. \end{aligned} \tag{2.1}$$

ParsEval scores and F_1 scores are used synonymously in some studies.

For dependency grammar induction, unlabeled attachment score (UAS), also known as directed dependency accuracy (DDA), is a standard and the most popular metric for measuring unsupervised dependency parsing quality. More specifically, it is the percentage of words that are correctly attached to their parents. UAS has been shown to be

[5] https://universaldependencies.org/

sensitive to annotation variations (Kübler et al., 2009; Schwartz et al., 2011; Tsarfaty et al., 2011). To address this issue, undirected UAS (UUAS) and neutral edge direction (NED; Schwartz et al. 2011) have been proposed by researchers. UUAS discards the direction of dependency edges and is therefore less biased towards such conventions. NED is defined as: traverse over the tokens and mark a correct attachment if the token's induced parent is either (1) its gold parent (2) its gold child or (3) its gold grandparent. This metric is even more tolerant in assessing parsing errors than UUAS. In this thesis, for the convenience of comparing with previous studies, we adopt UAS (or DDA) as our evaluation metric.

2.2 Neural Networks

In this section, we briefly review the popular neural networks for NLP tasks including sequential models and tree structured models. Formally speaking, modern neural NLP models often involve the encoder and decoder, where the former is specialized for language understanding and the latter for language generation. To have a better connection with our work that will be presented in the next chapters, we only focus on the encoder modeling in this section.

2.2.1 Recurrent and Recursive Neural Networks

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of deep neural networks that are specially designed to process temporal sequences. Derived from feed-forward neural networks, RNNs maintain an internal state (or memory) and update it at each time step. This enables RNNs to handle input sequences of variable lengths. More specifically, given a sequence of input vectors $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$, at each time step t, the RNN takes the input $\mathbf{x}_t$, updates the hidden state $\mathbf{h}_t$ and outputs $\mathbf{y}_t$. A typical implementation, the Elman network (Elman, 1990), is defined as follows

$$\begin{aligned} \mathbf{h}_t &= \sigma_h(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{b}_h) \\ \mathbf{y}_t &= \sigma_y(\mathbf{W}_y\mathbf{h}_t + \mathbf{b}_y). \end{aligned} \tag{2.2}$$

In the equations, $\mathbf{x}_t \in \mathbb{R}^d$, $\mathbf{h}_t \in \mathbb{R}^d$, $\mathbf{y}_t \in \mathbb{R}^{d'}$; $\mathbf{W}_h \in \mathbb{R}^{d\times d}$, $\mathbf{U}_h \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_y \in \mathbb{R}^{d'\times d}$ are weight matrices; $\mathbf{b}_h \in \mathbb{R}^d$ and $\mathbf{b}_y \in \mathbb{R}^{d'}$ are the bias, $\sigma_h(\cdot)$ and $\sigma_y(\cdot)$ are the non-linear activation functions. Normally, RNNs are trained by backpropagation through

time (BPTT). A major issue with gradient descent for standard RNNs is vanishing gradient, where the error gradient vanishes exponentially with the size of time lag. As one example of the problem cause, some activation functions (e.g., hyperbolic tangent) have gradients with magnitudes less than 1. When other activation functions are used whose gradients can take larger values, one risks encountering the related exploding gradient problem.

Long Short-Term Memory Networks

Long Short-Term Memory Networks (LSTMs; Hochreiter and Schmidhuber 1997) are specialized RNNs that solve the vanishing and exploding gradient problem. A common LSTM unit is composed of an internal memory cell, an input gate, a forget gate and an output gate. The memory cell in the LSTM keeps track of the dependencies among different elements in the input sequence. At each time step t, the input gate $\mathbf{i}_t$ controls how much information from a new input flows into the cell; the forget gate $\mathbf{f}_t$ controls how much history memory is kept in the cell; the output gate $\mathbf{o}_t$ controls how much information in the current cell is used to compute the output. The forward pass of a LSTM unit is given by

$$\begin{bmatrix} \mathbf{f}_t \\ \mathbf{i}_t \\ \mathbf{o}_t \\ \tilde{\mathbf{c}}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} (\mathbf{W} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}) \tag{2.3}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{2.4}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{2.5}$$

where $\tilde{\mathbf{c}}_t$ is cell candidate for the current input, $\mathbf{h}_t$ is the hidden state also known as output vector of the LSTM unit, $\mathbf{W} \in \mathbb{R}^{4d \times 2d}$ is the weight matrix and $\mathbf{b} \in \mathbb{R}^{4d}$ is the bias. σ and tanh are element-wise sigmoid and hyperbolic tangent functions respectively. $\odot$ is element-wise multiplication. Figure 2.3 shows an illustration of the update of the LSTM unit.

Recursive Neural Networks

Recursive Neural Networks (RvNNs) are a class of deep neural networks crafted by applying the same set of weights recursively over a structured input. Unlike the standard RNNs that are specified to process the input in a sequential order, RvNNs can

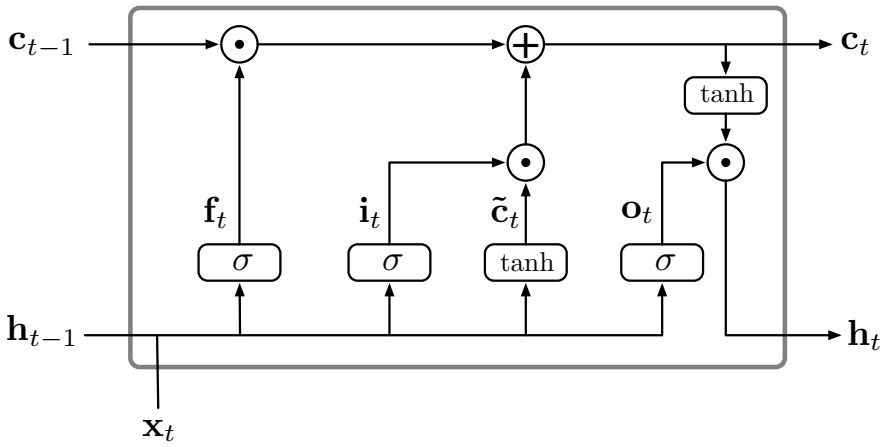


Figure 2.3: Illustration of the LSTM unit. $\mathbf{x}_t$, $\mathbf{h}_t$, $\mathbf{c}_t$ and $\tilde{\mathbf{c}}_t$ are the input, hidden state, memory cell and memory cell candidate at time step t. $\mathbf{h}_{t-1}$, $\mathbf{c}_{t-1}$ are for the previous time step. $\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$ are the forget, input and output gates. $\odot$ and $\oplus$ indicate element-wise multiplication and summation operators.

be applied to a wide range of structures. Given a structure, RvNNs process the elements by traversing the structure in its topological order. This property makes RvNNs favourable in many NLP applications (e.g., sentence representation learning), where trees are considered to be the underlying structures of natural language.

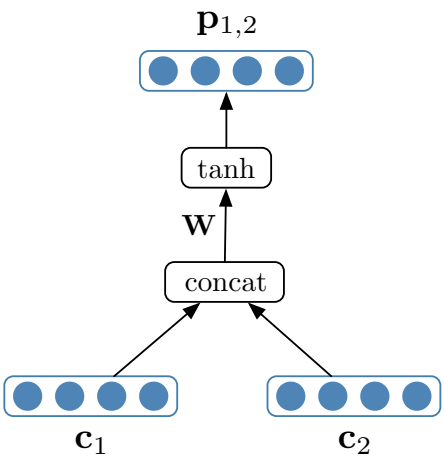


Figure 2.4: Illustration of a simple RvNN architecture. $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{p}_{1,2}$ are two children vectors and one parent vector respectively. $\mathbf{p}_{1,2} = \tanh(\mathbf{W}[\mathbf{c}_1;\mathbf{c}_2])$, where $\mathbf{W}$ is the weight matrix and $[\cdot;\cdot]$ indicates the vector concatenation.

In a basic RvNN, children nodes are combined into parents using a weight matrix that is shared across the whole network, together with a non-linearity such as the hyperbolic tangent function (shown in Figure 2.4). $\mathbf{c}_1$ and $\mathbf{c}_1$ are the representation

of two children nodes in a given structure. The representation for the parent can be computed as

$$\mathbf{p}_{1,2} = \tanh(\mathbf{W}\begin{bmatrix}\mathbf{c}_1\\ \mathbf{c}_2\end{bmatrix}), \tag{2.6}$$

where $\mathbf{c}_1 \in \mathbb{R}^d$, $\mathbf{c}_2 \in \mathbb{R}^d$, $\mathbf{p}_{1,2} \in \mathbb{R}^d$ and $\mathbf{W} \in \mathbb{R}^{d\times 2d}$ is the weight matrix. The subscript in $\mathbf{p}_{1,2}$ indicates the node's children indices. More complicated parameterization for RvNNs include Matrix-Vector RvNNs (Socher et al., 2012), Recursive Neural Tensor Networks (Socher et al., 2013), and so on.

Tree-LSTMs

Tree-LSTMs (Tai et al., 2015) are popular parameterization of RvNNs for NLP tasks, which are a generalization of LSTMs to tree-structured network topologies. The core idea is to inject syntactic knowledge to sentence modeling by extending the chain-structured LSTMs to the tree-structured LSTMs. Both dependency trees and constituency trees can be leveraged to obtain the tree structures.

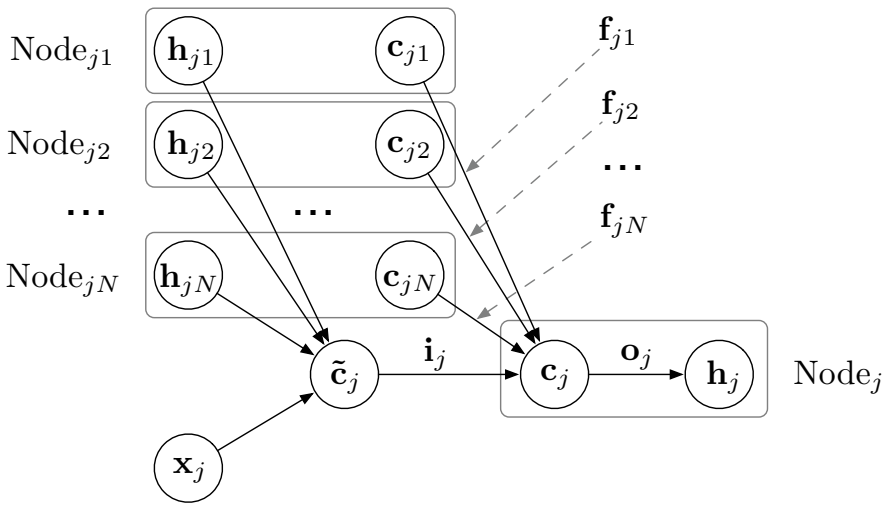


Figure 2.5: Illustration of a N-ary Tree-LSTM unit. Node_{jk} is the kth child node of Node_j ($1 \leq k \leq N$). $\mathbf{h}_{jk}$ and $\mathbf{c}_{jk}$ are the hidden state and memory cell for the child Node_{jk}. $\mathbf{h}_j$, $\mathbf{c}_j$, $\tilde{\mathbf{c}}_j$ and $\mathbf{x}_j$ are the hidden state, memory cell, memory cell candidate and input for the parent Node_j. $\mathbf{f}_{jk}$ is the corresponding forget gate for the child Node_{jk}. $\mathbf{i}_j$ and $\mathbf{o}_j$ are the input and output gate for the parent Node_j.

A typical implementation for a N-ary tree structured Tree-LSTM is shown in Figure 2.5. Distinct from the standard LSTM, at a given node j, the input gate $\mathbf{i}_j$ and output gate $\mathbf{o}_j$ are computed based on the hidden states from all its children nodes; separate

forget gates $\mathbf{f}_{jk}$ are computed based on the corresponding children node that is indexed by k. Then the forward pass for the node j is

$$\begin{bmatrix} \mathbf{i}_j \\ \mathbf{o}_j \\ \tilde{\mathbf{c}}_j \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \tanh \end{bmatrix} (\mathbf{W} \begin{bmatrix} \mathbf{h}_{j1} \\ \dots \\ \mathbf{h}_{jN} \\ \mathbf{x}_j \end{bmatrix} + \mathbf{b}) \tag{2.7}$$

$$\mathbf{f}_{jk} = \sigma(\mathbf{W}_f \begin{bmatrix} \mathbf{h}_{jk} \\ \mathbf{x}_j \end{bmatrix} + \mathbf{b}_f) \tag{2.8}$$

$$\mathbf{c}_j = \mathbf{i}_j \odot \tilde{\mathbf{c}}_j + \sum_{k=1}^{N} \mathbf{f}_{jk} \odot \mathbf{c}_{jk} \tag{2.9}$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{c}_j), \tag{2.10}$$

where $\mathbf{x}_j \in \mathbb{R}^d$ represents the input of node j. For the kth child node ($1 \leq k \leq N$), $\mathbf{f}_{jk}$ is the corresponding forget gate, $\mathbf{h}_{jk} \in \mathbb{R}^d$ and $\mathbf{c}_{jk} \in \mathbb{R}^d$ are the hidden state and cell state. $\mathbf{W} \in \mathbb{R}^{3d \times (N+1)d}$ and $\mathbf{W}_f \in \mathbb{R}^{d \times 2d}$ are the weight matrices. $\mathbf{b} \in \mathbb{R}^{3d}$ and $\mathbf{b}_f \in \mathbb{R}^d$ are the bias. The N-ary Tree-LSTMs are often used on tree structures where the branching factor is at most N and children are ordered. Therefore, N-ary Tree-LSTMs are well-suited for constituency trees, such as a commonly used simplified form, binary constituency trees. In practice, only the leaf node takes the corresponding word vectors as input on the constituency tree. For dependency trees where branching factor is high and children are unordered, Child-Sum Tree-LSTMs (Tai et al., 2015) are a better choice. We refer readers to the original paper for more details.

2.2.2 Transformers

Recurrent models typically factor computation along the element positions of the input in the sequential order while recursive models factors it in the topological order. Along the positions of different time steps, they produce a sequence of hidden states. Each hidden state is conditioned on the previous hidden states as well as the input. This inherent sequential nature hinders the parallelization at training time, which causes bottleneck for longer training examples and larger data volume.

The Transformer (Vaswani et al., 2017) is a novel attention mechanism-based architecture that is proposed to address this problem. Attention mechanisms have become an essential part of compelling sequence models in various tasks, allowing modeling of dependencies regardless of their distance in the input sequence. By utilizing

the causal masks, the Transformer can be fully parallelized for sequential modeling. More specifically, given a sequence of input vectors $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, where $\mathbf{x}_t \in \mathbb{R}^d$ $(1 \leq t \leq n)$, the Transformer is constructed by stacked multi-head self-attention and point-wise fully connected layers (shown in Figure 2.6):

$$A^l = \text{LayerNorm}\ (H^{l-1} + \text{MultiHeadAtt}(H^{l-1})) \tag{2.11}$$

$$H^l = \text{LayerNorm}\ (A^l + \text{FFN}(A^l)), \tag{2.12}$$

where $H^l = [\mathbf{h}_1^l; \mathbf{h}_2^l; \cdots; \mathbf{h}_n^l]$, $\mathbf{h}_t^l$ is the representation vector at l^{th} layer at time step t, A^l is the output of the lth self-attention layer. LayerNorm, MultiHeadAtt and FFN are discussed below.

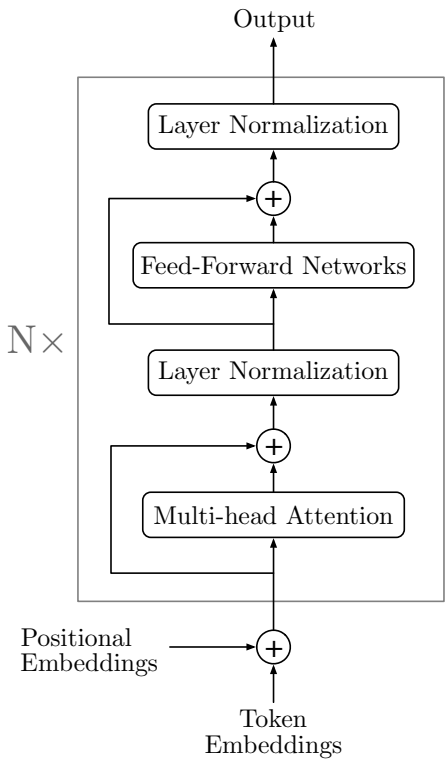


Figure 2.6: Illustration of a N layer Transformer model.

Multi-head Attention

An attention function can be depicted as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is the weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query and the corresponding key. Instead of performing a single attention function with d-dimensional keys, values and queries, the Transformer

linearly project the queries, keys and values K times with different, learnable linear projections to d_k dimension, where $d_k = \frac{d}{K}$. For example, for the representation at time step t with the kth head, the attention function is computed as

$$\begin{bmatrix} \mathbf{Q} \\ \mathbf{K} \\ \mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_q \\ \mathbf{W}_k \\ \mathbf{W}_v \end{bmatrix} H \tag{2.13}$$

$$\mathbf{head}_k = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{2.14}$$

where $H \in \mathbb{R}^{d\times n}$ is the representation from the previous layer, $\mathbf{W}_q \in \mathbb{R}^{d_k\times d}$, $\mathbf{W}_k \in \mathbb{R}^{d_k\times d}$, $\mathbf{W}_v \in \mathbb{R}^{d_k\times d}$ are weight matrices for queries $\mathbf{Q} \in \mathbb{R}^{d_k\times n}$, keys $\mathbf{K} \in \mathbb{R}^{d_k\times n}$ and values $\mathbf{V} \in \mathbb{R}^{d_k\times n}$, $\mathbf{head}_k$ is the output values with the kth head.

The attention function is performed in parallel on each head and all the yielding output values are concatenated and once again projected, resulting in the final values, which is calculated as follows

$$\text{MultiHeadAttn}(H) = \mathbf{W}_o[\,\mathbf{head}_1;\,\mathbf{head}_2;\cdots;\,\mathbf{head}_K], \tag{2.15}$$

where $\mathbf{W}_o \in \mathbb{R}^{d\times d}$ is the output matrix.

Layer Normalization and Feed-Forward Networks

Layer Normalization (Ba et al., 2016) is a simple normalization method to speed up the training for neural networks. It computes the mean and variance used for normalization from all the summed inputs to the neurons in a layer on a single training case. Each neuron is given its own bias and gain which are applied after the normalization. Given a vector $\mathbf{h} \in \mathbb{R}^d$, the forward pass of Layer Normalization is

$$a = \frac{1}{d}\sum_{i=1}^{d}\mathbf{h}[i] \tag{2.16}$$

$$b = \frac{1}{d}\sum_{i=1}^{d}(\mathbf{h}[i]-a)^2 \tag{2.17}$$

$$\text{LayerNorm}(\mathbf{h}) = \mathbf{W}_L\frac{\mathbf{h}-a}{\sqrt{b+\varepsilon}} + \mathbf{b}_L\ , \tag{2.18}$$

where $\mathbf{h}[i]$ is the ith element in $\mathbf{h}$, $\mathbf{W}_L$ is the weight matrix, $\mathbf{b}_L$ is the bias and ε is a small scalar to prevent division by zero.

The feed-forward networks (FFN) in the Transformer consists of two linear transformations and ReLU activation function in between. Given the vector $\mathbf{h} \in \mathbb{R}^d$,

$$\text{FFN}(\mathbf{h}) = \mathbf{W}_2\max(0, \mathbf{W}_1\mathbf{h}+\mathbf{b}_1)+\mathbf{b}_2, \tag{2.19}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{ff} \times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d \times d_{ff}}$ are the weight matrices, $\mathbf{b}_1 \in \mathbb{R}^{d_{ff}}$ and $\mathbf{b}_2 \in \mathbb{R}^d$ are the bias and $\max(\cdot)$ is the element-wise maximum operation.

Positional Embeddings

Since the Transformer contains no recurrence or convolution to model the positional dependencies among elements, information about the positions of the tokens in the sequence is injected to the Transformer via positional embeddings. At position t of the input sequence, the input embedding $\mathbf{h}_t^0$ is give by

$$\mathbf{h}_t^0 = \mathbf{x}_t + \mathbf{PE}_t, \tag{2.20}$$

where $\mathbf{x}_t \in \mathbb{R}^d$ is the word embedding and $\mathbf{PE}_t \in \mathbb{R}^d$ is the positional embedding at position t in the input sequence. In the original Transformer model, $\mathbf{PE}_t$ is defined by sine and cosine functions of different frequencies:

$$\mathbf{PE}_t[i] = \sin(t/10000^{2i/d}) \tag{2.21}$$

$$\mathbf{PE}_t[2i+1] = \cos(t/10000^{2i/d}), \tag{2.22}$$

where $\mathbf{PE}_t[i]$ indicates the ith element in the vector $\mathbf{PE}_t$. In this way, each dimension of the positional embedding corresponds to a sinusoid. For any fixed offset δ, $\mathbf{PE}_{t+\delta}$ can be represented as a linear function of $\mathbf{PE}_t$, so that the model can easily learn to attend by relative positions. Follow-up Transformer model variants also treat positional embeddings as learnable parameters and learn them end-to-end on the various NLP tasks.

2.2.3 Pre-trained Language Models

In recent years, substantial work has shown that pre-trained language models (PLMs) are able to learn universal language representations. Such models are pre-trained on large corpus with unsupervised objectives and are beneficial for downstream NLP tasks when they are further fine-tuned on annotated data. Surprisingly, this fine-tuned models can outperform those trained from scratch, usually by large margins. Thanks to their superior ability to scale up, the Transformer models are always selected as the backbones of PLMs.

Earlier work on pre-trained models focuses on learning good word embeddings, such as Skip-Gram (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). Due to the weak computational power in early days and the fact that these models are no

longer used for downstream tasks (only learned embeddings are used), they are usually shallow for efficient training. As a result, although the learned word embeddings can capture semantic meanings of words, they are context-free and struggle to capture high-level linguistic properties. On the contrast, recent PLMs aim at learning contextual word embeddings, such as CoVe (McCann et al., 2017), ELMo (Peters et al., 2018a), GPT (Radford et al., 2018) and BERT (Devlin et al., 2019). The learned encoders are employed to compute context-sensitive word representations for downstream tasks. Among them, the Transformer-based PLMs are particularly effective as the models are deeper and larger than RNN-based alternatives.

BERT

Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019) is a representative Transformer-based PLM that achieves remarkable success by presenting state-of-the-art results in a wide variety of NLP tasks. As opposed to uni-directional language models, which read the text sequentially (left-to-right, right-to-left or combined together), the BERT encoder reads the entire input sequence at once. It means, each token in the BERT encoder has access to the tokens from both sides simultaneously. Consequently, conventional uni-directional language modeling is no more a suitable objective for the BERT encoder. Inspired by the Cloze task (Taylor, 1953), a novel *masked language modeling* (MLM) is used for BERT training. The masked language model randomly masks out some tokens from the original input sequence, and the objective is to predict the masked tokens based on its bi-directional context. In addition to the masked language modeling, BERT also introduces a *next sentence prediction* task that jointly pre-trains text-pair representations. After pre-training on a huge corpus including the Wikipedia and books, the BERT encoder can be fine-tuned with an additional output layer to achieve state of the art on a wide range of NLP tasks such as question answering and natural language inference.

For a given token, the input representation in BERT is the sum of the corresponding token, segment and position embeddings. More specifically, token embeddings take the sub-word as the basic unit using WordPiece (Wu et al., 2016); positional embeddings are learned to support sequences with length up to 512; the first token of every sequence is set to be a special classification token (`[CLS]`); sentence pairs are packed into a single sequence that are distinguished by a special token (`[SEP]`) and segment embeddings. Figure 2.7 illustrates the training process of the BERT encoder on a given training sample.

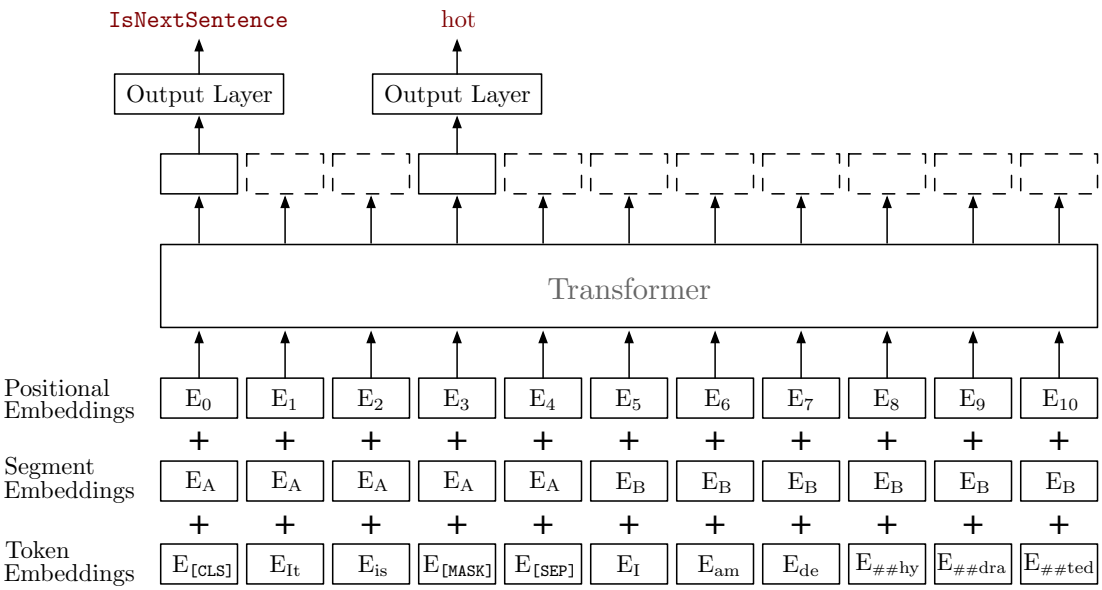


Figure 2.7: Illustration of the BERT encoder trained on the masked language modeling and next sentence prediction objectives. In this example, two adjacent sentences "It is hot" and "I am dehydrated" are packed into a single input sequence, where "dehydrated" is shattered into multiple sub-words ("de", "##hy", "##dra", "##ted") and "hot" is masked out for masked language modeling training.

Other PLMs

Inspired by the success of BERT, researchers have put great effort into the Transformer-based PLMs. RoBERTa (Liu et al., 2019c) is an enhanced version of BERT by modifying BERT: removing the next sentence prediction objective; training the model longer, over more data, with bigger batches and longer sequences; dynamically changing the masking pattern applied to the training data. XLNet (Yang et al., 2019) is a generalized auto-regressive pre-trained model that still enables learning bidirectional contexts. Rather than the above models functioning as encoders and focusing on the natural language understanding tasks, GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020) are Transformer-based uni-directional language models that are pre-trained like Transformer decoders and have the ability of generation.

Transformer-based models have also been pre-trained in the encoder-decoder settings for both natural language understanding and generation. UniLM (Dong et al., 2019; Bao et al., 2020) extends the mask prediction on three types of language modeling tasks: uni-directional, bi-directional, and sequence-to-sequence (Seq2Seq) prediction. BART (Lewis et al., 2020) constructs a denoising autoencoder for pre-training

sequence-to-sequence models by corrupting texts with arbitrary noising functions and learning to reconstruct them. T5 (Raffel et al., 2020) reframes all NLP tasks into a unified text-to-text-format and is pre-trained on a new cleaned common crawl-based corpus.

Beyond English, multilingual PLMs have been investigated to learn text representations across different languages for cross-lingual NLP tasks. Multilingual BERT (mBERT; Devlin et al. 2019) is pre-trained with the shared vocabulary on Wikipedia text from over 100 languages. XLM (Conneau and Lample, 2019) improves mBERT by incorporating a cross-lingual task, translation language modeling, which performs MLM on the packed parallel bilingual sentence pairs. XLM-RoBERTa (XLM-R; Conneau et al. 2020) is a scaled multilingual encoder pre-trained on a significantly increased amount of data with the monolingual masked language modeling objective. MASS (Song et al., 2019) pre-trains a Seq2Seq model with monolingual Seq2Seq MLM on multiple languages and achieves significant improvement for unsupervised neural machine translation. XNLG (Chi et al., 2020) performs a two-stage pre-training scheme for cross-lingual natural language generation. A multilingual extension of BART (mBART; Liu et al. 2020), pre-trains the Transformer jointly with a Seq2Seq denoising autoencoder task on large-scale monolingual corpora across 25 languages and brings improvement to machine translation tasks.

Although multilingual PLMs perform well across many languages, some research work points out that PLMs trained on a single language can outperform the multilingual counterparts (Martin et al., 2019; Virtanen et al., 2019). Researchers have also developed language specific PLMs for Chinese (Cui et al., 2019; Sun et al., 2019b; Zhang et al., 2019; Sun et al., 2020), French (Martin et al., 2019; Le et al., 2019), Russian (Yu and Arkhipov, 2019), Finnish (Virtanen et al., 2019), German [6], Korean [7], and so on.

[6] bert-base-german (https://deepset.ai/german-bert) and bertbase-dbmdz (https://github.com/dbmdz/berts).

[7] KoBERTbase (https://github.com/SKTBrain/KoBERT).

Chapter 3

Transition-based Unsupervised Dependency Parsing

Grammar induction is the task of deriving plausible syntactic structures from raw text, without the use of annotated training data. In the case of dependency parsing, the syntactic structure takes the form of a tree whose nodes are the words of the sentence, whose arcs are directed and denote head-dependent relationships between words. Inducing such a tree without annotated training data is challenging because of data sparseness and ambiguity, and because the search space of potential trees is huge, making optimization difficult. State-of-the-art models, including both the generative and the discriminative models, all rely on global inference, which is implemented by dynamic programming with $O(n^3)$ run time. For the generative models, probabilistic dependency grammars are always used. While for the discriminative models, graph-based models are popular choices. On the other hand, transition-based models enable faster inference with $O(n)$ run time, but the performance still lags behind their global inference-based counterparts. In this chapter, we propose a neural transition-based parser for dependency grammar induction with $O(n)$ run time, whose inference procedure utilizes rich neural features. We train the parser with integration of variational inference, posterior regularization and variance reduction techniques. The resulting framework outperforms previous unsupervised transition-based dependency parsers and achieves performance comparable to global inference-based models, both on the English Penn Treebank and eight languages on the Universal Dependency Treebank. In an empirical comparison, we show that our approach substantially improves parsing speed over global inference-based models.

Most existing approaches to dependency grammar induction, including both the

generative and the discriminative models, have used exact global inference. For the generative models, probabilistic dependency grammars are always used such as the Dependency Model with Valence (DMV; Klein and Manning 2004). While for the discriminative models, graph-based models are popular choices. Among them, state-of-the-art representatives include LC-DMV (Noji et al., 2016), NDMV (Jiang et al., 2016), L-NDMV (Han et al., 2017) and D-NDMV (Han et al., 2019). Though such models achieve impressive results, their inference procedure requires $O(n^3)$ run time. Meanwhile, features in these models must be decomposable over substructures to enable dynamic programming. In contrast, transition-based models allow faster inference in linear time and are compatible with richer feature sets. Although relying on local inference, transition-based models have been shown to perform well in supervised parsing (Dyer et al., 2015; Kiperwasser and Goldberg, 2016). However, few works have studied unsupervised transition-based parsers. One exception is the work of Rasooli and Faili (2012), in which search-based structure prediction (Daumé III, 2009) is used with a simple feature set. In general, for transition-based approaches, there is still a significant performance gap compared to global inference-based ones.

In this chapter, we make a departure from the existing literature in dependency grammar induction, by proposing a novel unsupervised transition-based parser. We borrow the idea from the recurrent neural network grammar (RNNG, Dyer et al. 2016) to build our backbone parser. RNNG is a probabilistic transition-based model for constituency parsing. It can be used either in a generative way as a language model or in a discriminative way as a parser. Cheng et al. (2017) used an autoencoder to integrate discriminative and generative RNNGs, yielding a reconstruction process with parse trees as latent variables and enabling the two components to be trained jointly on a language modeling objective. However, their work uses observed trees for training and does not study unsupervised learning. Inspired by Cheng et al. (2017), we propose an unsupervised neural variational transition-based parser. Concretely, we first modify the transition actions in the original RNNGs into a set of arc-standard actions for projective dependency parsing and then build a dependency variant of the model of Cheng et al. (2017). Although this approach performs well for supervised parsing, when applied in an unsupervised setting, it fails dramatically. We hypothesize that this is because the parser is relatively unconstrained: no conditional independence assumptions are made; no prior linguistic knowledge is injected. Therefore, we augment the model with posterior regularization, allowing us to seamlessly integrate linguistic knowledge in the shape of a small number of universal linguistic rules and still maintain the efficiency

of transition-based models. In addition, we propose a novel variance reduction method for stabilizing neural variational inference with discrete latent variables. This yields the first known model that makes it possible to use posterior regularization for neural variational inference with discrete latent variables.

In the experiments on the English Penn Treebank and on eight languages from the Universal Dependency (UD) Treebank, we find that our model with posterior regularization outperforms the previous best unsupervised transition-based dependency parser (Rasooli and Faili, 2012), and approaches the performance of global inference-based models. We also show how a weak form of supervision can be integrated into our framework in the form of rule expectations. Furthermore, we present empirical evidence for the complexity advantage of transition-based models: our model attains a large speed-up compared to a representative global inference-based model. [1]

3.1 Related Work

In this section, we start by reviewing the two model variants of the global inference-based models, i.e., generative models and discriminative models. Then we discuss the transition-based models and other techniques for unsupervised dependency parsing.

3.1.1 Generative Models

Generative approaches model the joint probability of the sentence as well as the corresponding dependency parse tree. Traditional models mostly use probabilistic grammars. Conditional independence assumptions (e.g., the context-free assumption) are always made by such models to enable efficient inference. In the dynamic-programming based inference, the joint probability is decomposed into a product of independent component probabilities or scores. However, such assumptions also lead to unavailability of useful information (e.g., context and generation history) in the generation process.

Different generative models specify different generation processes of the sentence and parse tree under their respective independence assumptions. Paskin (2002) first uniformly sampled a dependency tree skeleton and then populated the nodes with tokens conditioned on the dependency tree in a recursive root-to-leaf manner. On the contrary, the Dependency Model with Valence (DMV; Klein and Manning 2004) gen-

[1] Our code is available at https://github.com/libowen2121/VI-dependency-syntax

erates the sentence and the parse tree simultaneously. In the generation process of DMV, a decision is first sampled to decide whether to generate a child token or terminate conditioned on the head token and the dependency direction. Then a child token is sampled additionally conditioned on the valence, defined as the number of the child tokens already generated from the head token. Empirically, DMV is the first model that outperforms a simple but strong baseline, right-branching (left-headed), where every word is attached to the previous word. Based on the vanilla DMV, Headden III et al. (2009) introduced the valence to the condition of decision sampling. Spitkovsky et al. (2012) additionally considered sibling words, sentence completeness, and punctuation context in decision sampling and child token generation.

By default, log marginal likelihood is used to optimize the generative models. Besides, priors and regularization terms are often added to the objective function to incorporate various inductive biases. Smith and Eisner (2006) introduced penalty terms into the objective to control dependency lengths and the root number of the parse tree. Naseem et al. (2010) proposed to use Posterior Regularization (Ganchev et al., 2010) to bring prior knowledge into the model as constraints. Tu and Honavar (2012) introduced an entropy term to constrain the ambiguity. Noji et al. (2016) proposed to inject a hard constraint to the objective that limits the degree of center-embedding of the parse tree. Mareček and Žabokrtský (2012) and Mareček and Straka (2013) proposed the reducibility principle to model the head-dependant relations in the generation process.

Moreover, efforts have also been put to improve the Expectation Maximization (EM) learning algorithm used in the generative models. Smith and Eisner (2005) introduced the contrastive estimation to the learning algorithm. Spitkovsky et al. (2013) proposed to switch between different objectives to break out of local optima.

3.1.2 Discriminative Models

Different from the generative counterparts, discriminative approaches model the conditional probability or score of the dependency parse tree given the sentence. By conditioning on the whole input sentence, discriminative approaches are able to utilize not only local features (i.e., features related to the current dependency) but also global features (i.e., contextual features from the whole sentence).

Autoencoder-based approaches have recently been popular in the development of discriminative models. They map a sentence into an intermediate representation with

an encoder and then reconstruct the observed sentence from the intermediate representation with a decoder. One common method is to treat the dependency structure as the intermediate representation. Cai et al. (2017) proposed CRF-AE to use a first-order graph-based conditional random field (CRF) parser to map the input sentence into the dependency parse tree. Then the decoder independently generates each token of the reconstructed sentence conditioned on the head of the token specified by the dependency tree generated by the CRF parser. For marginalization, dynamic programming-based exact inference is conducted.

Recently, Han et al. (2019) proposed to use a continuous sentence embedding as the intermediate representation to capture global sentence context, where a LSTM is utilized as an encoder. The decoder is a neural DMV (Jiang et al., 2016) that is conditioned on the sentence vector generated by the LSTM. In their model, marginalization over the continuous sentence vector is computationally intractable. Accordingly, variational inference is employed where a variational autoencoder is formalized. Similarly, we also utlize variational inference to resolve the marginalization problem. In our work, we propose a dependency-based recurrent neural network grammar (RNNG, Dyer et al. 2016) for unsupervised dependency parsing and use the discriminative RNNG and generative RNNG as the encoder and decoder respectively. In our case, rich neural feature utilization as well as the linear run time requirement block the exact inference. So we formulate a variational autoencoder framework to address this problem. Details of our model will be presented in the following sections. Corro and Titov (2019a) also proposed a variational autoencoder-based model where the encoder is a CRF parser and the decoder is a graph convolutional network (GCN) whose structure is specified by the parse tree generated from the CRF parser.

Aside from the (variational) autoencoder based models, there are also other discriminative models for unsupervised dependency parsing. Le and Zuidema (2015) designed a complicated reranking-based system. Grave and Elhadad (2015) proposed the Convex-MST to employ a first-order graph-based discriminative parser, where they searched for the parses of all the training sentences and learned the parser simultaneously. Daumé III (2009) introduced a stochastic search-based method to do unsupervised parsing. Jiang et al. (2017) combined the models of Grave and Elhadad (2015) and Noji et al. (2016) with dual decomposition inference algorithm.

3.1.3 Transition-based Models

Other than the models we reviewed above, another model family is transition-based. Many transition-based systems exist in the literature, especially for the supervised dependency parsing. We adopt the arc-standard (Nivre, 2003) system for our model in this chapter (details will be presented in the following section). Compared to models with global inference, transition-based models are capable of fully utilizing the parsing history and supports linear parsing run time if greedy decoding is performed. Note that when the feature set is properly constrained (e.g., a minimal set of bidirectional LSTM features), practical dynamic programming-based global inference is also available for transition-based models (Cross and Huang, 2016; Shi et al., 2017; Gómez-Rodríguez et al., 2018).

Transition-based models are also explored as an alternative for unsupervised dependency parsing. Daumé III (2009) proposed a stochastic search-based method to do unsupervised transition-based parsing, which is a discriminative model. Rasooli and Faili (2012) proposed a generative unsupervised parsing model together with "baby-step" training (Spitkovsky et al., 2010) to improve parsing accuracy. In general, transition-based models are less studied for unsupervised dependency parsing, where a large performance gap exists compared to models with global inference. In this chapter, we manage to propose a transition-based model that performs competitively compared to models with global inference and still keeps the linear run time.

3.1.4 Other Techniques

In this section, we briefly review other techniques proposed by researchers to further improve the unsupervised dependency parsing performance.

In earlier days, Cohen et al. (2009) and Cohen and Smith (2009) leveraged logistic-normal prior distributions to encourage correlations between POS tags in DMV. More recently, thanks to the capability of parameter-sharing and over-parameterization, neural networks have been brought into the research of unsupervised dependency parsing. To encode correlation between POS tags and smooth the probabilities of grammar rules, Jiang et al. (2016) for the first time, introduced neural networks into DMV. Han et al. (2019) extended the generative approach in Jiang et al. (2016) to a discriminative approach by further utilizing sentence context by a neural network. Our work in this chapter and Corro and Titov (2019a) use RNNGs and GCNs respectively to score dependencies in the discriminative models.

In the most common setting of unsupervised dependency parsing, the parser is unlexicalized, where the POS tags are either human annotated or induced from the training corpus (Spitkovsky et al., 2011; He et al., 2018). However, different words sharing the same POS tag may have different syntactic behaviour. Researchers have introduced lexical information into unsupervised parsers (Headden III et al., 2009; Blunsom and Cohn, 2010; Spitkovsky et al., 2013; Pate and Johnson, 2016; Han et al., 2017). Our work in this chapter experiments with both unlexical and lexical settings.

3.2 Problem Formulation

To build our dependency grammar induction model, we follow Cheng et al. (2017) and propose a dependency-based, encoder-decoder RNNG. This model includes (1) a discriminative RNNG as the *encoder* (Section 3.2.2) to map the input sentence into a discrete latent variable, which is a sequence of parse actions to build a dependency tree; (2) a generative RNNG as the *decoder* (Section 3.2.2) to reconstruct the input sentence based on the latent parse actions. The training objective is the marginal likelihood of the observed input sentence, which is reformulated as an evidence lower bound (ELBO) and solved with neural variational inference. The REINFORCE algorithm (Williams, 1992) is utilized to handle discrete latent variables in optimization (Section 3.2.3). Overall, the encoder and decoder are jointly trained, inducing latent parse trees or actions from only unlabeled text data. To further regularize the space of parse trees with a linguistic prior, we introduce posterior regularization into the basic framework (Section 3.2.4). Finally, we propose a novel variance reduction technique to train our posterior regularized framework more effectively (Section 3.2.5).

3.2.1 Background

RNNG (Dyer et al., 2016) is a top-down transition-based system originally proposed for constituency parsing. Basically, there are two variants: the discriminative RNNG and the generative RNNG. The discriminative RNNG takes a sentence as input, and predicts the probability of a corresponding parse tree conditioned on the sentence. The model uses a buffer to store unprocessed terminal words and a stack to store partially completed syntactic constituents. It then follows top-down transition actions to shift words from the buffer to the stack to construct syntactic constituents incrementally.

The discriminative RNNG can be modified slightly to formulate the generative

RNNG, an algorithm for incrementally producing trees and sentences in a generative fashion. In the generative RNNG, there is no buffer of unprocessed words, but there is an output buffer to store words that have been generated. Top-down actions are specified to generate terminals (words) and non-terminals in pre-order. Though not able to parse on its own, a generative RNNG can be used for language modeling as long as parse trees are sampled from a known distribution.

We modify the transition actions in the original RNNG into a set of arc-standard actions for projective dependency parsing. In the discriminative modeling case, the action space includes:

- SHIFT fetches the first word in the input buffer and pushes it onto the top of the stack;
- LEFT-REDUCE adds a left arc in between the top two words of the stack and merges them into a single construct;
- RIGHT-REDUCE adds a right arc in between the top two words of the stack and merges them into a single construct.

In the generative modeling case, the SHIFT operation is replaced by a GEN operation:

- GEN generates a word and adds it to the stack and the output buffer.

Running examples are given in Table 3.1 and 3.2 to illustrate discriminative and generative dependency RNNGs give an input sentence "I saw a girl".

3.2.2 Model Configuration

Encoder We formulate the encoder as a discriminative dependency RNNG that computes the conditional probability $p(\mathbf{a}|\mathbf{x})$ of the transition action sequence $\mathbf{a}$ given the observed sentence $\mathbf{x}$. The conditional probability is factorized over time steps, and parameterized by a transitional state embedding $\mathbf{v}$:

$$p(\mathbf{a}|\mathbf{x}) = \prod_{t=1}^{|\mathbf{a}|} p(\mathbf{a}_t|\mathbf{v}_t), \tag{3.1}$$

where $\mathbf{v}_t$ is the transitional state embedding of the encoder at time step t. Specifically, we use the following features for $\mathbf{v}_t$: (1) the stack embedding $\mathbf{e}_t$ obtained with a stack-LSTM (Dyer et al., 2015, 2016) that encodes the stack of the encoder; (2) the input buffer embedding $\mathbf{i}_t$, where we use a bidirectional LSTM to compose the input buffer

#	Stack	Input Buffer	Discriminative Action	Parse Tree
0	ϕ	*I saw a girl*	SHIFT	ϕ
1	*I*	*saw a girl*	SHIFT	*I*
2	*I saw*	*a girl*	LEFT-REDUCE	*I saw*
3	*saw*	*a girl*	SHIFT	*I saw*
4	*saw a*	*girl*	SHIFT	*I saw a*
5	*saw a girl*	ϕ	LEFT-REDUCE	*I saw a girl*
6	*saw girl*	ϕ	RIGHT-REDUCE	*I saw a girl*
7	*saw*	ϕ	-	*I saw a girl*

Table 3.1: The parsing process of the discriminative dependency RNNG give an input sentence “I saw a girl”.

and represent each word as a concatenation of forward and backward LSTM states. Finally, $\mathbf{v}_t$ is computed as:

$$\mathbf{v}_t = \mathbf{W}_2 \tanh(\mathbf{W}_1[\mathbf{e}_t, \mathbf{i}_t] + \mathbf{b}_e), \tag{3.2}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$ are weight parameters and $\mathbf{b}_e$ the bias. We note that the encoder is the actual component for **parsing** at run time.

Decoder The decoder is a generative dependency RNNG that models the joint probability $p(\mathbf{x}, \mathbf{a})$ of a latent transition action sequence $\mathbf{a}$ and an observed sentence $\mathbf{x}$. This joint distribution can be factorized into a sequence of action and word (emitted by GEN) probabilities, which are parameterized by a transitional state embedding $\mathbf{u}$:

$$\begin{aligned} p(\mathbf{x}, \mathbf{a}) &= p(\mathbf{a}) p(\mathbf{x}|\mathbf{a}) \\ &= \prod_{t=1}^{|\mathbf{a}|} p(\mathbf{a}_t|\mathbf{u}_t) p(\mathbf{x}_t|\mathbf{u}_t)^{I(\mathbf{a}_t = \text{GEN})}, \end{aligned} \tag{3.3}$$

where I is an indicator function and $\mathbf{u}_t$ is the state embedding at time step t. Specifically, we use the following features: (1) the stack embedding $\mathbf{d}_t$ which encodes the

#	Stack	Output Buffer	Generative Action	Parse Tree
0	ϕ	ϕ	GEN	ϕ
1	*I*	*I*	GEN	*I*
2	*I saw*	*I saw*	LEFT-REDUCE	*I* *saw*
3	*saw*	*I saw*	GEN	*I* *saw*
4	*saw a*	*I saw a*	GEN	*I* *saw* *a*
5	*saw a girl*	*I saw a girl*	LEFT-REDUCE	*I* *saw* *a* *girl*
6	*saw girl*	*I saw a girl*	RIGHT-REDUCE	*I* *saw* *a* *girl*
7	*saw*	*I saw a girl*	-	*I* *saw* *a* *girl*

Table 3.2: The generation process of the generative dependency RNNG give an input sentence “I saw a girl”.

stack of the decoder and is obtained with a stack-LSTM; (2) the output buffer embedding $\mathbf{o}_t$, where we use a standard LSTM to compose the output buffer and $\mathbf{o}_t$ is represented as the most recent state of the LSTM. Finally, $\mathbf{u}_t$ is computed as:

$$\mathbf{u}_t = \mathbf{W}_4 \tanh(\mathbf{W}_3[\mathbf{d}_t, \mathbf{o}_t] + \mathbf{b}_d), \tag{3.4}$$

where $\mathbf{W}_3$, $\mathbf{W}_4$ are weight parameters and $\mathbf{b}_d$ the bias.

The model configuration above is borrowed from Cheng et al. (2017). The differences are (1) we use neither the parent non-terminal embedding nor the action history embedding for both the decoder and encoder; (2) we do not use the adaptive buffer embedding for the encoder. The reason is that the expressive power of the model for unsupervised parsing should be fairly constrained to avoid overfitting.

3.2.3 Training Objective

Consider a latent variable model in which the encoder infers the latent transition actions (i.e., the dependency structure) and the decoder reconstructs the sentence from these actions. The maximum likelihood estimate of the model parameters is deter-

mined by the log marginal likelihood of the sentence:

$$\log p(\mathbf{x}) = \log \sum_{\mathbf{a}} p(\mathbf{x}, \mathbf{a}). \tag{3.5}$$

Since the form of the log likelihood is intractable in our case, we optimize the ELBO by Jensen's Inequality as follows:

$$\begin{aligned} \log p(\mathbf{x}) &\geqslant \log p(\mathbf{x}) - KL[q(\mathbf{a})||p(\mathbf{a}|\mathbf{x})] \\ &= \mathbb{E}_{q(\mathbf{a})}[\log \frac{p(\mathbf{x}, \mathbf{a})}{q(\mathbf{a})}] = \mathcal{L}_x, \end{aligned} \tag{3.6}$$

where $KL[\cdot||\cdot]$ is the Kullback-Leibler divergence and $q(\mathbf{a})$ is the variational approximation of the true posterior $p(\mathbf{a}|\mathbf{x})$. This training objective is optimized with the EM algorithm. In the E-step, the variational distribution $q(\mathbf{a}|\mathbf{x})$ is estimated based on the encoder and the observation $\mathbf{x}$ — $q(\mathbf{a})$ is parameterized as $q_\omega(\mathbf{a}|\mathbf{x})$, where ω represents the parameters of the encoder. Similarly, the joint probability $p(\mathbf{x}, \mathbf{a})$ is parameterized by the decoder as $p_\theta(\mathbf{x}, \mathbf{a})$, where θ represents the parameters of the decoder.

In the M-step, the decoder parameters θ can be directly updated by gradient descent via Monte Carlo simulation:

$$\begin{aligned} \frac{\partial \mathcal{L}_x}{\partial \theta} &= \mathbb{E}_{q_\omega(\mathbf{a}|\mathbf{x})}[\frac{\partial \log p_\theta(\mathbf{x}, \mathbf{a})}{\partial \theta}] \\ &\approx \frac{1}{M} \sum_m \frac{\partial \log p_\theta(\mathbf{x}, \mathbf{a}^{(m)})}{\partial \theta} \end{aligned} \tag{3.7}$$

where M samples $\mathbf{a}^{(m)} \sim q_\omega(\mathbf{a}|\mathbf{x})$ are drawn independently to compute the stochastic gradient.

For the encoder parameters ω, since the sampling operation is not differentiable, we approximate the gradients using the policy gradient method:

$$\begin{aligned} \frac{\partial \mathcal{L}_x}{\partial \omega} &= \mathbb{E}_{q_\omega(\mathbf{a}|\mathbf{x})}[l(\mathbf{x}, \mathbf{a}) \frac{\partial \log q_\omega(\mathbf{a}|\mathbf{x})}{\partial \omega}] \\ &\approx \frac{1}{M} \sum_m l(\mathbf{x}, \mathbf{a}^{(m)}) \frac{\partial \log q_\omega(\mathbf{a}^{(m)}|\mathbf{x})}{\partial \omega}, \end{aligned} \tag{3.8}$$

where

$$l(\mathbf{x}, \mathbf{a}) = \log \frac{p_\theta(\mathbf{x}, \mathbf{a})}{q_\omega(\mathbf{a}|\mathbf{x})}. \tag{3.9}$$

Note that, technically we are not using the score function gradient estimator, which is also known as the REINFORCE algorithm (Williams, 1992; Glynn, 1987; Mohamed et al., 2020). In our formulation, we consider l as the reward. This gives us some room to *tweak* the reward, which will be clear when we incorporate the posterior regularization in the next section.

3.2.4 Posterior Regularization

As shown in the following section (Section 3.3.3), the basic model discussed previously performs poorly when directly applied to unsupervised parsing, barely outperforming the left-branching baseline for English. We hypothesize the reason is that the basic model is fairly unconstrained. Without any constraints to regularize the latent space, the induced parses will be arbitrary, since the model is only trained to maximize sentence likelihood (Naseem et al., 2010; Noji et al., 2016), especially given the fact that the dependency RNNGs have strong expressive power.

We therefore introduce posterior regularization (PR; Ganchev et al. 2010) to encourage the neural network to generate linguistically plausible trees. Via posterior regularization, we can give the model access to a small amount of linguistic prior in the form of syntactic rules, which are universal for all languages. These rules effectively function as features, which impose soft constraints on the neural parameters in the form of expectations.

To integrate PR constraints into the model, a set Q of allowed posterior distributions over the hidden variable $\mathbf{a}$ can be defined as:

$$Q = \{q(\mathbf{a}) : \exists \xi,\ \mathbb{E}_q[\phi(\mathbf{x},\mathbf{a})] - \mathbf{b} \leqslant \xi;\ ||\xi||_\beta \leqslant \varepsilon\}, \tag{3.10}$$

where $\phi(\mathbf{x},\mathbf{a})$ is a vector of feature functions, $\mathbf{b}$ is a vector of given negative expectations, ξ is a vector of slack variables, ε is a predefined small value and $||\cdot||_\beta$ denotes some norm. The PR algorithm only works if Q is non-empty.

In dependency grammar induction, $\phi_k(\mathbf{x},\mathbf{a})$ (the kth element in $\phi(\mathbf{x},\mathbf{a})$) can be set to be the negative number of times a given rule (dependency arcs, e.g., *Root* $\rightarrow$ *Verb*, *Verb* $\rightarrow$ *Noun*) occurs in a sentence. We hope to bias the learning so that each sentence is parsed to contain such kinds of arcs more than a threshold in the expectation. The posterior regularized likelihood is then:

$$\begin{aligned} \mathcal{L}_Q &= \max_{q \in Q} \mathcal{L}_x \\ &= \log p(\mathbf{x}) - \min_{q \in Q} KL[q(\mathbf{a}) \ ||\ p(\mathbf{a}|\mathbf{x})]. \end{aligned} \tag{3.11}$$

Equation (3.11) indicates that, in the posterior regularized framework, $q(\mathbf{a})$ not only approximates the true posterior $p(\mathbf{a}|\mathbf{x})$ (estimated by the encoder network $q_\omega(\mathbf{a}|\mathbf{x})$) but also belongs to the constrained set Q. To optimize $\mathcal{L}_Q$ via the EM algorithm, we get

the revised E′-step as:

$$\begin{aligned} q(\mathbf{a}) &= \operatorname*{argmax}_{q \in Q} \mathcal{L}_Q \\ &= \operatorname*{argmin}_{q \in Q} KL[q(\mathbf{a}) \mid\mid q_\omega(\mathbf{a}|\mathbf{x})]. \end{aligned} \tag{3.12}$$

Formally, the optimization problem in the E′-step can be described as:

$$\begin{aligned} \min_{q,\xi} \quad & KL[q(\mathbf{a}) \mid\mid q_\omega(\mathbf{a}|\mathbf{x})]. \\ s.t. \quad & \mathbb{E}_q[\phi(\mathbf{x},\mathbf{a})] - \mathbf{b} \leqslant \xi; \;\; ||\xi||_\beta \leqslant \varepsilon \end{aligned} \tag{3.13}$$

Following Ganchev et al. (2010), we can solve the optimization problem in (3.13) in its Lagrangian dual form. Thanks to the fact that our transition-based encoder satisfies the decomposition property, the conditional probability $q_\omega(\mathbf{a}|\mathbf{x})$ can be factored as $\prod_{t=1}^{|\mathbf{a}|} q_\omega(\mathbf{a}_t|\mathbf{v}_t)$ in (3.1). Thus, the factored primal solution can be written as:

$$q(\mathbf{a}) = \frac{q_\omega(\mathbf{a}|\mathbf{x})}{Z(\lambda^*)} \exp(-\lambda^{*T}\phi(\mathbf{x},\mathbf{a})), \tag{3.14}$$

where λ is the Lagrangian multiplier whose solution is given as $\lambda^* = \operatorname{argmax}_{\lambda \geqslant 0} -\mathbf{b}^T\lambda - \log Z(\lambda) - \varepsilon||\lambda||_{\beta^*}$[2] and $Z(\lambda)$ is given as:

$$Z(\lambda) = \sum_{\mathbf{a}} q_\omega(\mathbf{a}|\mathbf{x}) \exp(-\lambda^T\phi(\mathbf{x},\mathbf{a})). \tag{3.15}$$

We also define the multiplier introduced by PR as:

$$\gamma(\mathbf{a},\mathbf{x}) = \frac{1}{Z(\lambda)} \exp(-\lambda^T\phi(\mathbf{x},\mathbf{a})). \tag{3.16}$$

Computing the partition function $Z(\lambda)$ is also intractable in our case. To address this problem, we view $Z(\lambda)$ as an expectation and estimate it by Monte Carlo simulation as:

$$\begin{aligned} Z(\lambda) &= \mathbb{E}_{q_\omega(\mathbf{a}|\mathbf{x})}[\exp(-\lambda^T\phi(\mathbf{x},\mathbf{a}))] \\ &\approx \frac{1}{M}\sum_m \exp(-\lambda^T\phi(\mathbf{x},\mathbf{a}^{(m)})). \end{aligned} \tag{3.17}$$

The Monte Carlo estimate of the partition function $Z(\lambda)$ will introduce bias and implicate the estimated gradient with respect to $\lambda = 0$.

[2] $||\cdot||_{\beta^*}$ is the dual norm of $||\cdot||_\beta$. Here we use ℓ_2 norm for both primal norm $||\cdot||_\beta$ and dual norm $||\cdot||_{\beta^*}$.

The gradients for the encoder and decoder in the M-step is be defined as follows:

$$\begin{aligned}\frac{\partial \mathcal{L}_x}{\partial \theta} &= \frac{1}{M}\sum_m \frac{\partial \log p_\theta(\mathbf{x},\mathbf{a}^{(m)})}{\partial \theta} \\ \frac{\partial \mathcal{L}_x}{\partial \omega} &= \frac{1}{M}\sum_m l(\mathbf{x},\mathbf{a}^{(m)})\frac{\partial \log q(\mathbf{a}^{(m)})}{\partial \omega}.\end{aligned} \tag{3.18}$$

Note that the Monte Carlo simulation should be performed over distribution $q(\mathbf{a})$. In practice, we perform the importance sampling that we use $q_\omega(\mathbf{a}|\mathbf{x})$ as the proposal distribution.

The original reward without a baseline should be defined as:

$$\log \frac{p_\theta(\mathbf{x},\mathbf{a})}{q(\mathbf{a})} = \log \frac{p_\theta(\mathbf{x},\mathbf{a})}{q_\omega(\mathbf{a}|\mathbf{x})\gamma(\mathbf{a},\mathbf{x})} \tag{3.19}$$

But in practical training, we observe that learning with the reward in Equation (3.19) is not stable. So we simply use the reward in Equation (3.9). We show how this modification affects the loss function in Appendix A.1.

As shown in later sections (Section 3.2.5 and 3.3.3), pre-training is empirically effective, we use $\gamma(\mathbf{x},\mathbf{a})$ as an extra learning signal and add it to the gradients of both encoder and decoder. Intuitively, we also would like the decoder also get the reward from l so that the encoder and decoder can couple better during training. Finally, the gradients of the encoder and decoder are defined as

$$\begin{aligned}\frac{\partial \mathcal{L}_x}{\partial \theta} &= \frac{1}{M}\sum_m \gamma(\mathbf{x},\mathbf{a}^{(m)})l(\mathbf{x},\mathbf{a}^{(m)})\frac{\partial \log p_\theta(\mathbf{x},\mathbf{a}^{(m)})}{\partial \theta} \\ \frac{\partial \mathcal{L}_x}{\partial \omega} &= \frac{1}{M}\sum_m \gamma(\mathbf{x},\mathbf{a}^{(m)})l(\mathbf{x},\mathbf{a}^{(m)})\frac{\partial \log q(\mathbf{a}^{(m)})}{\partial \omega}\end{aligned} \tag{3.20}$$

where l is computed as Equation (3.9).

3.2.5 Variance Reduction in the M-step

Training a neural variational inference framework with discrete latent variables is known to be a challenging problem (Mnih and Gregor, 2014; Miao and Blunsom, 2016; Miao et al., 2016). This is mainly caused by the sampling step of discrete latent variables which results in high variance, especially at the early stage of training when both encoder and decoder parameters are far from optimal. Intuitively, $l(\mathbf{x},\mathbf{a})$ weighs the gradient for each latent sample $\mathbf{a}$, and its variance plays a crucial role in updating the parameters in the M-step.

To reduce the variance and stabilize the learning process, previous studies (Mnih and Gregor, 2014; Miao and Blunsom, 2016; Miao et al., 2016) used the *baseline method* (RL-BL), re-defining l as:

$$l_{\text{RL-BL}}(\mathbf{x},\mathbf{a}) = l(\mathbf{x},\mathbf{a}) - b(\mathbf{x}) - b \tag{3.21}$$

where $b(\mathbf{x})$ is a parameterized, input-dependent baseline (e.g., a neural language model in our case) and b is the bias. For the baseline $(b(\mathbf{x})+b)$ in RL-BL, we pre-train a LSTM language model. During training the RL-BL, we fix the LSTM language model and rescale and shift the output $\log p(\mathbf{x})$ to fit the ELBO of the given sentence as

$$b(\mathbf{x}) + b = \alpha \log p(\mathbf{x}) + \tau$$

The baseline method is able to reduce the variance to some extent, but also introduces extra model parameters that complicate optimization. In the following we propose an alternative generic method for reducing the variance of the gradient estimator in the M-step, as well as another task-specific method which results in further improvement.

Generic Method The intuition behind the generic method is as follows: the algorithm takes M latent samples for each input $\mathbf{x}$ and a score $l(\mathbf{x},\mathbf{a}^{(m)})$ is computed for each sample $\mathbf{a}^{(m)}$, hence the variance can be reduced by normalization within the group of samples. This motivates the following $l_{\text{RL-SN}}(\mathbf{x},\mathbf{a})$:

$$l_{\text{RL-SN}}(\mathbf{x},\mathbf{a}) = \frac{l(\mathbf{x},\mathbf{a}) - \bar{l}(\mathbf{x},\mathbf{a})}{\max(1, \sqrt{\text{Var}[l(\mathbf{x},\mathbf{a})]})} \tag{3.22}$$

Task-Specific Method Besides the generic variance reduction method which applies to discrete neural variational inference in general, we further propose to enhance the quality of $l_{\text{RL-SN}}(\mathbf{x},\mathbf{a})$ for the specific dependency grammar induction task. Intuitively, $l(\mathbf{x},\mathbf{a})$ in (3.20) weights the gradient of a given sample $\mathbf{a}$ by a positive or negative value, while $\gamma(\mathbf{x},\mathbf{a})$ only weights the gradient by a positive value. As a result, $l(\mathbf{x},\mathbf{a})$ plays a crucial role in determining the optimization direction. Therefore, we propose to correct the polarity of our $l_{\text{RL-SN}}(\mathbf{x},\mathbf{a})$ with the number of rules $s(\mathbf{x},\mathbf{a}) = -\text{SUM}[\phi(\mathbf{x},\mathbf{a})]$ that occur in the induced dependency structure, where $\text{SUM}[]$ returns the sum of vector elements. The refined $l(\mathbf{x},\mathbf{a})$ is:

$$l_{\text{RL-PC}}(\mathbf{x},\mathbf{a}) = \begin{cases} |l_{\text{RL-SN}}(\mathbf{x},\mathbf{a})| & \hat{s}(\mathbf{x},\mathbf{a}) \geqslant 0 \\ -|l_{\text{RL-SN}}(\mathbf{x},\mathbf{a})| & \hat{s}(\mathbf{x},\mathbf{a}) < 0 \end{cases} \tag{3.23}$$

where $\hat{s}(\mathbf{x},\mathbf{a}) = \frac{s(\mathbf{x},\mathbf{a})-\bar{s}(\mathbf{x},\mathbf{a})}{\sqrt{\mathrm{Var}[s]}}$. Since $\hat{s}(\mathbf{x},\mathbf{a})$ provides a natural corrective, we can obtain a simpler variant of (3.23) by directly using $\hat{s}(\mathbf{x},\mathbf{a})$ as the reward:

$$l_{\text{RL-C}}(\mathbf{x},\mathbf{a}) = \hat{s}(\mathbf{x},\mathbf{a}) \tag{3.24}$$

We will experimentally compare the different variance reduction techniques (or rewards) of the reinforcement learning objective.

Pre-training Unsupervised models in general face a *cold-start* problem since no gold annotations exist to *warm up* the model parameters quickly. This can be observed in (3.20): the gradient updates of the model are dependent on the reward l, which in return relies on the model parameters. At the beginning of training we cannot obtain an accurately approximated l to update model parameters. To alleviate this problem, one approach is to ignore it in the gradient update at the early stage. In this case, both the encoder and decoder are trained with the direct reward from PR (shown in Algorithm 1).

Algorithm 1: Pre-training for Neural Variational Inference Dependency Parser.

Parameters: $\omega, \theta, \lambda, \varepsilon, ||\cdot||_\beta, M$
Constrained Feature Functions: $\phi(\mathbf{x},\mathbf{a})$
Initialization;
while *not converged* **do**
  Sample $\mathbf{a}^{(m)} \sim q_\omega(\mathbf{a}|\mathbf{x}), 1 \leqslant m \leqslant M$;
  PR Computation:
  $Z(\lambda) \approx \frac{1}{M}\sum_m \exp(-\lambda^T\phi(\mathbf{x},\mathbf{a}^{(m)}))$, $\gamma(\mathbf{x},\mathbf{a}^{(m)}) = \frac{1}{Z(\lambda)}\exp(-\lambda^T\phi(\mathbf{x},\mathbf{a}^{(m)}))$;
  Update parameters in mini-batch:
  Update θ w.r.t. its gradient $\frac{1}{M}\sum_m \gamma(\mathbf{x},\mathbf{a}^{(m)})\frac{\partial \log p_\theta(\mathbf{x},\mathbf{a}^{(m)})}{\partial\theta}$;
  Update ω w.r.t. its gradient $\frac{1}{M}\sum_m \gamma(\mathbf{x},\mathbf{a}^{(m)})\frac{\partial \log q_\omega(\mathbf{a}|\mathbf{x})}{\partial\omega}$;
  Update λ to optimize $\max_{\lambda\geqslant 0} \quad -\mathbf{b}^T\lambda - \log Z(\lambda) - \varepsilon||\lambda||_{\beta^*}$ (with projected gradient descent algorithm).
end

3.2.6 Limitations

We admit that the approaches we proposed in this section have their heuristic nature. Different from the standard REINFORCE algorithm, or the score function estimator,

we optimize the model using a more general approach, policy gradient. The variance reduction methods, including both the generic method and task-specific methods, can be considered as changes of the objective, modifications of the ELBO. Since we have *tweaked* the learning objective, unlike typical unbiased variance reduction techniques, our methods are biased. On the other hand, the universal rules are not perfect; they can also be biased.

Given the fact that parsing speed is our first concern, we are using parsers with linear runtime, namely neural transition-based parsers, in our model. We resort to universal linguistic rules to regularize the latent space and incorporate complicated approximation into the learning process for better empirical results. Suppose we did not have such runtime constraints, we would have tried a neural CRF parser (Cai et al., 2017) alternatively as the encoder in our model. In this way, we could naturally inject the context-free assumption and regularize the latent space by maximising the entropy. Kim et al. (2019a) has investigated this idea and successfully employed it to unsupervised constituency parsing.

3.3 Experiments

3.3.1 Datasets

English Penn Treebank We use the Wall Street Journal (WSJ) section of the English Penn Treebank (Marcus et al., 1993). To be in line with previous work, the dataset is preprocessed to strip off punctuation. We train our model on sections 2–21, tune the hyperparameters on section 22, and evaluate on section 23. Sentences of length ≤ 10 are used for training. We report directed dependency accuracy (DDA) on test sentences of length ≤ 10 (WSJ-10) and on all sentences (WSJ).

Universal Dependency Treebank We select eight languages from the Universal Dependency Treebank 1.4 (Nivre et al., 2016). We train our model on training sentences of length ≤ 10 and report DDA on test sentences of length ≤ 15 and ≤ 40. We find that training on short sentences generally increase performance compared to training on longer sentences (e.g., length ≤ 15).

Projectivity In English, projective trees are sufficient to analyze most sentence types. In fact, the dependency English Penn Treebank is automatically generated from the

original English Penn Treebank and is by convention exclusively projective. [3] In languages with more flexible word order than English, non-projective trees are more frequent. The eight languages we select from the UD treebank are all projective. We note that transition-based parsers can only produce projective trees. It's reasonable to expect that our approach will perform less well on non-projective treebanks, such as Czech.

3.3.2 Settings

We employ the universal linguistic rules from Naseem et al. (2010) and Noji et al. (2016) for WSJ and the Universal Dependency Treebank, respectively (shown in Table 3.3 and 3.4). For WSJ, we expand the coarse rules defined in Naseem et al. (2010) with the Penn Treebank fine-grained part-of-speech tags. For example, *Verb* is expanded as *VB*, *VBD*, *VBG*, *VBN*, *VBP* and *VBZ*.

Since the universal linguistic rules are defined on POS tags, it's essential to use gold POS tags in our approach. Actually, for the task of POS tagging, modern neural network-based models (Bohnet et al., 2018) can reach 97.96 accuracy on the English Penn Treebank and 93.40 accuracy on over fifty languages (95.20 accuracy on our selected eight languages). It will be interesting to evaluate how our approach performs using predicted POS tags. We leave it for future work.

Root → Auxiliary	Noun → Adjective
Root → Verb	Noun → Article
Verb → Noun	Noun → Noun
Verb → Pronoun	Noun → Numeral
Verb → Adverb	Preposition → Noun
Verb → Verb	Adjective → Adverb
Auxiliary → Verb	

Table 3.3: Universal dependency rules for WSJ (Naseem et al., 2010).

We use AdaGrad (Duchi et al., 2011) to optimize the parameters of the encoder and decoder, as well as the projected gradient descent algorithm (Bertsekas, 1999) to optimize the parameters of posterior regularization. During learning, the posterior reg-

[3] Actually, like we discussed in Section 2.1.4, there exist different ways to convert the original English Penn Treebank. Some of them may produce non-projective trees. In this section, we use the data in Jiang et al. (2016).

ROOT → VERB	NOUN → ADJ
ROOT → NOUN	NOUN → DET
VERB → NOUN	NOUN → NOUN
VERB → ADV	NOUN → NUM
VERB → VERB	NOUN → CONJ
VERB → AUX	NOUN → ADP
ADJ → ADV	

Table 3.4: Universal dependency rules for the Universal Dependency Treebank (Noji et al., 2016).

ularization is incorporated on sentence level. [4] We use GloVe embeddings (Pennington et al., 2014) to initialize English word vectors and FastText embeddings (Bojanowski et al., 2016) for the other languages. Across all experiments, we test both unlexicalized and lexicalized variants of our model. The unlexicalized variant uses gold POS tags as model inputs, while the lexicalized variant additionally use word tokens. Following previous work (Buys and Blunsom, 2015), we use Brown clustering (Brown et al., 1992) to obtain additional features in the lexicalized variant. We report average DDA and best DDA over five runs for our main parsing results.

3.3.3 Exploration of Model Variants

Posterior Regularization To study the effectiveness of posterior regularization in our neural grammar induction model, we first implement a fully unsupervised model without posterior regularization. This model is trained with variational inference, using the standard REINFORCE objective with a *baseline* (Mnih and Gregor, 2014; Miao and Blunsom, 2016; Miao et al., 2016) and employing no posterior regularization. Table 3.5 shows the results for the unsupervised model, together with the random and left- and right-branching baselines. We observe that the unsupervised model (both unlexicalized and lexicalized) fails to beat the left-branching baseline. These results

[4]Cheng et al. (2017) optimize their neural variational inference-based model at the sentence level. For the posterior regularization, in the original paper (Ganchev et al., 2010), the constraints are enforced at the instance (sentence) level in practice, although the constraints are defined corpus-wide. Thus, introducing the posterior regularization at the batch level is a better approximation. However, it's not trivial to implement mini-batching in this case. The deep learning library we used in this chapter, PyTorch (Paszke et al., 2019), does not support autobatching. Some other libraries do provide the autobatching functionality, such as DyNet (Neubig et al., 2017) and JAX (Bradbury et al., 2018). It's also viable to construct mini-batching manually like Corro and Titov (2019b) did in the latent dependency tree learning.

Model	WSJ-10	WSJ
Random	19.1	16.4
Right branching	20.1	20.6
Left branching	36.2	30.2
UNSUPERVISED	33.3 (39.0)	29.0 (30.5)
L-UNSUPERVISED	34.9 (36.4)	28.0 (30.2)

Table 3.5: Evaluation of the fully unsupervised model (without posterior regularization) on the English Penn Treebank. We report average DDA and the best DDA (in brackets) over five runs. "L-" denotes the lexicalized variant.

	WSJ-10	WSJ
No Pre-training	47.5 (59.8)	36.7 (46.3)
Pre-training	64.8 (67.1)	42.0 (43.7)

Table 3.6: Evaluation of the posterior-regularized model with and without pre-training on the WSJ. We report average DDA and best DDA (in brackets) over five runs.

suggest that without any prior linguistic knowledge, the trained model is fairly unconstrained. A comparison with posterior-regularized results in Table 3.6 (to be discussed next) reveals the effectiveness of posterior regularization in incorporating additional linguistic knowledge.

Pre-training Table 3.6 shows the results of a standard posterior-regularized model compared to one only with pre-training. Both models use the unlexicalized setup. We find that the posterior-regularized model benefits a lot from pre-training, which therefore is a useful way to avoid cold start.

Variance Reduction Previously, we described various variance reduction techniques, or modified rewards, for the reinforcement learning objective. These include the conventional *baseline* method (RL-BL), our sample normalization method (RL-SN), sample normalization with additional polarity correction (RL-PC), and a simplified version of the later (RL-C). We now compare these techniques; all experiments were conducted with pre-training and on the unlexicalized model. The experimental results in Table 3.7 show that RL-SN outperforms RL-BL on average DDA, which indicates that sample normalization is more effective in reducing the variance of the gradient

	Generic		Task-specific	
	RL-BL	RL-SN	RL-C	RL-PC
μ	58.7	60.8	64.4	66.7
σ	1.8	0.6	0.3	0.7

Table 3.7: Comparison of models with different variance reduction techniques (or rewards) on the WSJ-10 test set. We report the average DDA μ and its standard deviation σ over five runs. RL-BL: REINFORCE with baseline. RL-SN: REINFORCE with sample normalization in a group. RL-C: REINFORCE using polarity the correction criteria as rewards. RL-PC: REINFORCE with polarity correction.

estimator. We hypothesize that the gain comes from the fact that sample normalization does not introduce extra model parameters, whereas RL-BL does. Polarity correction further boosts performance. However, polarity correction uses the number of universal rules present in a induced dependency structure, i.e., it is a task-specific method for variance reduction. Also RL-C (the simplified version of RL-PC) achieves competitive performance.

Universal Rules In our PR scheme, the rule expectations can be uniformly initialized. This approach does not require any annotated training data; the parser is furnished only with a small set of universal linguistic rules. We call this setting UNIVERSALRULES. However, we can initialize the rule expectation non-uniformly, which allows us to introduce a gentle degree of supervision into the PR scheme. Here, we explore one way of doing this: we assume a training set that is annotated with dependency rules (e.g., the training portion of the WSJ), based on which we estimate expectations for the universal rules. In practice, such expectations can be provided by human experts. We call this setting WEAKLYSUPERVISED. The results of an experiment comparing these two settings is shown in Table 3.8. In both cases we use pre-training and the best performing reward RL-PC. Here we report results using both unlexicalized and lexicalized settings. It can be seen that the best performing UNIVERSALRULES model is the unlexicalized one, while the best WEAKLYSUPERVISED model is lexicalized. We conjecture that a more powerful model (the lexicalized variant) is capable of making better use of the additional supervision, i.e., the non-uniform rule expectations. Overall, WEAKLYSUPERVISED outperforms UNIVERSALRULES, which demonstrates that our posterior regularized parser is able to effectively use weak supervision in the form

Model	WSJ-10	WSJ
UNIVERSALRULES	**54.7** (58.2)	**37.8** (39.3)
L-UNIVERSALRULES	54.7 (56.3)	36.8 (38.1)
WEAKLYSUPERVISED	66.7 (67.6)	43.6 (45.0)
L-WEAKLYSUPERVISED	**68.2** (71.1)	**48.6** (50.2)

Table 3.8: Comparison of uniformly initialized (UNIVERSALRULES) and empirically estimated (WEAKLYSUPERVISED) rule expectation on the WSJ. We report average DDA and best DDA (in brackets) over five runs.

of an empirical initialization of the rule expectations.

3.3.4 Parsing Results

English Penn Treebank

Since English Penn Treebank is a standard benchmark for unsupervised dependency parsing, here we do a more comprehensive comparison of our model against other models including both global inference-based and transition-based models in Table 3.9. For the transition-based models, we compare our unsupervised UNIVERSALRULES model and its WEAKLYSUPERVISED variant with Daumé III (2009) as well as the state-of-the-art Rasooli and Faili (2012) denoted as RF. Since we use different preprocessing, we re-implement the RF model for a fair comparison. For global inference-based models, although they are not directly comparable to our approach, we still present them for reference. In particular, we present the results of recent work, including the state of the art, from the categories of both generative and discriminative models. Among them, Convex-MST (Grave and Elhadad, 2015) and HDP-DEP (Naseem et al., 2010) also utilize universal linguistic rules. Specifically, we use the exactly same rules as HDP-DEP (Naseem et al., 2010), while the rules used in Convex-MST (Grave and Elhadad, 2015) are slightly different. We refer readers to the original paper for more details.

Concretely, the parser of Rasooli and Faili (2012) is unlexicalized and count-based. To reach the best performance, the authors employ *baby steps* (i.e., they start training on short sentences and gradually add longer sentences (Spitkovsky et al., 2009)), as well as two heuristics called H1 and H2. H1 involves multiplying the probability of the last verb reduction in a sentence by 10^{-10}. H2 involves multiplying each *Noun* $\rightarrow$ *Verb*,

Model	WSJ-10	WSJ
Global inference-based models		
Generative models		
HDP-DEP (Naseem et al., 2010)†	71.9	–
NDMV (Jiang et al., 2016)	72.5	57.6
L-NDMV (Han et al., 2017)	75.1	59.5
Discriminative models		
Convex-MST (Grave and Elhadad, 2015)†	60.8	48.6
CRFAE (Cai et al., 2017)	71.7	55.7
D-NDMV (Han et al., 2019)	**75.6**	**61.4**
Transition-based models		
Generative models		
RF (Rasooli and Faili, 2012)	37.3 (40.7)	32.1 (33.1)
RF+H1+H2 (Rasooli and Faili, 2012)‡	51.0 (52.7)	37.2 (37.6)
Discriminative models		
Daumé III (2009)	45.4	-
UNIVERSALRULES† (ours)	54.7 (58.2)	37.8 (39.3)
L-WEAKLYSUPERVISED†† (ours)	**68.2** (71.1)	**48.6** (50.2)

Table 3.9: Comparison of our models (UNIVERSALRULES and L-WEAKLYSUPERVISED) with previous work on the English Penn Treebank. †: universal linguistic rules are used. ††: weak supervision is used where rule expectations are estimated from a labeled training set. ‡: H1 and H2 are two heuristics used in Rasooli and Faili (2012). We report average DDA and best DDA (in brackets) over five runs for both RF and our models.

Adjective → *Verb*, and *Adjective* → *Noun* rule by 0.1. These heuristics seem fairly ad-hoc; they presumably bias the probability estimates towards more linguistically plausible values.

As the results in Table 3.9 show, our UNIVERSALRULES model outperforms RF on both WSJ-10 and full WSJ, achieving a new state of the art for transition-based dependency grammar induction. The RF model does not use universal rules, but its linguistic heuristics play a similar role, which makes our comparison fair. Note that our L-WEAKLYSUPERVISED model achieves a further improvement over UNIVERSALRULES, making it comparable with global inference-based models, demonstrating the potential of the neural, transition-based dependency grammar induction approach.

Universal Dependency Treebank

Our multilingual experiments use the UD treebank. Here we evaluate the two models that perform the best on the WSJ: the unlexicalized UNIVERSALRULE model and lexicalized L-WEAKLYSUPERVISED model. We use the same hyperparameters as in the WSJ experiments. Again, we mainly compare our models with the transition-based model RF (with heuristics H1 and H2). We also include the global inference-based models for reference including the state of the art.

Table 3.10 shows the UD treebank results. It can be observed that both UNIVERSALRULES and L-WEAKLYSUPERVISED significantly outperform the RF on both short and long sentences. The improvement of average DDA is roughly 20% on sentences of length ≤ 40. This shows that although the heuristic approach employed by Rasooli and Faili (2012) is useful for English, it does not generalize well across languages, in contrast to our posterior-regularized neural networks with universal rules. It is interesting that UNIVERSALRULES even outperforms L-WEAKLYSUPERVISED on longer sentences on average. We conjecture that universal linguistic rules with uniform expectations are robust in the multilingual setting especially given the fact that we directly utilize the hyperparameters of the English experiment and do not further tune them for each language. It is notable that our models match the perfomance of the neural DMV (NDMV, Jiang et al. 2016) which is a strong baseline for this task.

Parsing Speed

To highlight the advantage of our linear time complexity parser, we compare both lexicalized and unlexicalized variants of our parser with a representative DMV-based model, LC-DMV (Noji et al., 2016), in terms of parsing speed. The results in Table 3.11 show that our unlexicalized parser results in a 1.8-fold speed-up for short sentences (length ≤ 15), and a speed-up of factor 16 for long sentences (full length). And our parser does not lose much parsing speed even in a lexicalized setting.

3.4 Summary

In this chapter, we proposed a neural variational transition-based model for dependency grammar induction. The model consists of a generative dependency RNNG for generation, and a discriminative dependency RNNG for parsing and inference. We trained the model on unlabeled corpora with integration of neural variational infer-

Model	Global inference-based				Transition-based		
	Generative		Discriminative		Generative	Discriminative (Ours)	
	NDMV	LC-DMV	Conv-MST†	D-NDMV	RF+H1+H2‡	L-WEAKLYSUP††	UNIVRULES†
Length ≤ 15							
Basque	48.3	47.9	**52.5**	42.7	49.0 (51.0)	**55.2** (56.0)	52.9 (55.1)
Dutch	**44.1**	35.5	43.4	43.0	26.6 (31.9)	38.7 (41.3)	**39.6** (40.2)
French	59.5	52.1	61.6	**61.7**	33.2 (37.5)	56.6 (57.2)	**59.9** (61.6)
German	56.2	51.9	54.4	**58.5**	40.5 (44.0)	**59.7** (59.9)	57.5 (59.4)
Italian	72.7	73.1	**73.2**	63.5	33.3 (38.9)	58.5 (59.8)	**59.7** (62.3)
Polish	72.7	66.2	66.7	**75.8**	46.8 (59.7)	**61.8** (63.4)	57.1 (59.3)
Portuguese	34.4	**70.5**	60.7	69.1	35.7 (43.7)	52.5 (54.1)	**52.7** (54.2)
Spanish	38.1	65.5	61.6	**66.6**	35.9 (38.3)	**55.8** (56.2)	55.6 (56.8)
Average	53.3	57.8	59.3	**60.1**	37.6 (43.1)	**54.9** (56.0)	54.4 (56.1)
Length ≤ 40							
Basque	47.8	45.4	**50.0**	42.4	45.4 (47.6)	**51.0** (51.7)	48.9 (51.5)
Dutch	35.6	34.1	**45.3**	43.7	23.1 (30.4)	42.2 (44.8)	**42.5** (44.3)
French	38.1	48.6	**62.0**	58.5	27.3 (30.7)	46.4 (47.5)	**55.4** (56.3)
German	50.4	50.5	51.4	**52.9**	32.5 (37.0)	**55.6** (56.3)	54.2 (56.3)
Italian	63.6	**71.1**	69.1	61.3	27.7 (33.0)	54.1 (55.6)	**55.7** (58.7)
Polish	62.8	63.7	63.4	**73.0**	43.3 (46.0)	**57.3** (59.4)	51.7 (52.8)
Portuguese	49.0	**67.2**	57.9	65.7	28.8 (35.9)	44.6 (48.6)	**45.3** (46.5)
Spanish	58.0	61.9	61.9	**64.4**	26.9 (28.8)	50.8 (54.0)	**52.4** (53.9)
Average	50.7	55.3	57.6	**57.7**	31.9 (36.2)	50.3 (52.2)	**50.8** (52.5)

Table 3.10: Evaluation on eight languages of the UD treebank with test sentences of length ≤ 15 and length ≤ 40. NDMV: Jiang et al. (2016). LC-DMV: Noji et al. (2016). Conv-MST: Grave and Elhadad (2015). D-NDMV: Han et al. (2019). †: universal linguistic rules are used. ††: weak supervision is used where rule expectations are estimated from a labeled training set. ‡: H1 and H2 are two heuristics used in Rasooli and Faili (2012). We report average DDA and best DDA (in brackets) over five runs for both RF and our models.

Sentence length	$\leq$15	$\leq$40	All
LC-DMV	663	193	74
Our Unlexicalized	1192	1194	1191
Our Lexicalized	939	938	983

Table 3.11: Parsing speed (tokens per second) on the French UD Treebank with test sentences of various lengths. All experiments are conduct on the same CPU platform.

ence, posterior regularization and variance reduction techniques. This allows us to use a small amount of universal linguistic rules as prior knowledge to regularize the discrete latent space. We also showed that it is straightforward to integrate weak supervision into our model in the form of rule expectations. Empirically on English and eight other languages, our parser obtained a new state of the art for unsupervised transition-based dependency parsing and significantly narrowed the performance gap between transition-based models and global inference-based models. With respect to parsing speed, our model keeps the linear runtime superiority over global inference-based models. It was also verified empirically that a speed-up of factor 16 on sentences of full length was achieved by our model when compared with a representative DMV-based model (i.e., LC-DMV).

It's a trade-off between the parsing speed and performance. In this chapter, parsing speed is our major concern, so we employ a transition-based encoder (i.e., discriminative dependency RNNG). Since the encoder has strong expressive power, complicated techniques have been used to regularize the model. We believe that the parsing performance can be further boosted if the encoder is replaced with a global inference-based model (e.g., a CRF parser).

Compared to dependency structures, constituency structures are more appealing to neural NLP architectures, especially to recurrent and recursive neural networks. Constituency structures are easier to be integrated to sequential models, which brings the modeling convenience. In the next chapter, we will focus on the problem of learning latent constituency structures from downstream NLP tasks.

Chapter 4

Imitation Learning based Unsupervised Constituency Parsing

Natural language usually exhibits a sequential surface format, but the underlying structure governing the sentence is best represented as tree structure, which is widely believed by cognitive science and computational linguists (Chomsky and Lightfoot, 2002; Sag et al., 2003). The tree structure, also known as *syntax*, depict how surface words are composed into components of the sentence in a hierarchical manner. This phenomenon intrigues researchers to explore the possibility of introducing tree structure to end-to-end neural NLP models. From a practical point of view, injecting tree structure into neural NLP models is beneficial in four ways: (1) to obtain a hierarchical representation with increasing levels of abstraction, exploiting a key characteristic of deep neural networks (LeCun et al., 2015; Schmidhuber, 2015); (2) to capture complicated linguistic properties, such as compositional effects (Socher et al., 2013); (3) to address the long-term dependency problem (Tai et al., 2015; Li et al., 2015) where tokens can be structurally close but far away in sequential order ; (4) to provide shortcut for gradient backpropagation (Chung et al., 2016) with fewer updates at the more abstractive high-level layers.

Tree-structured recursive neural networks (Socher et al., 2011; Tai et al., 2015) are a class of representative syntax-aware neural NLP models. In these models, a sentence representation is built by incrementally composing children nodes to compute the representation for the parent node following the corresponding tree structure. They have been proven to be effective at sentence understanding tasks such as sentiment analysis (Socher et al., 2013), textual entailment (Bowman et al., 2016) and machine translation (Eriguchi et al., 2016). A supervised constituency parser is a common way

to obtain tree structures, where trees produced by the parser are used to guide the composition process.

However, supervised parsers are limited for several reasons: (1) for most low resource languages, no annotated data is available, and often not even an annotation scheme exists; (2) annotated data often comes from a newswire corpus, so a supervised parser faces an out-of-domain issue; (3) language may change over time (mainly on the lexical side), so structured models should evolve accordingly and handle the changes properly. Learning the tree structure from the data in an unsupervised manner becomes appealing and promising to address the limitations. Specifically, there has been increasing interest in latent tree induction using neural networks. The aim is to induce a tree structure for a sentence without having access to labeled trees during training. This approach has been shown to be beneficial for downstream natural language understanding tasks, such as sentiment analysis and natural language inference (Yogatama et al., 2017; Maillard et al., 2017; Choi et al., 2018). Apart from understanding tasks, there is also some research work managing to bring tree structures to another classic NLP task, language modeling (Shen et al., 2018b, 2019b; Wang et al., 2019).

A natural question is whether such latent trees, which are the results of optimizing a training objective on a classification task or language modeling, correspond to standard syntactic trees as annotated in treebanks. For a classification training objective, the answer has so far been negative, and furthermore, latent tree induction shows low self-agreement when randomly initialized multiple times (Williams et al., 2018a). However, for a language modeling training objective, strong parsing performance have been achieved. Controversially, an external biased parsing approach is utilized to extract tree structures from the model. This approach has been challenged as it overestimates the parsing performance and is only effective for English (Dyer et al., 2019).

The parsing-reading-predict network (PRPN, Shen et al. 2018b) is a latent tree neural language model that achieves remarkable performance on the language modeling task. With respect to the unsupervised parsing performance, PRPN has been claimed to be the first latent tree induction model to successfully produce syntactically plausible structures (Htut et al., 2018). The model is based on a continuous notion of syntactic distance, which can be computed by differentiable structured attention. However, PRPN does not model the parsing action directly. Constituency trees need to be extracted externally, working together with a biased parser. This parser is not part of the

model and cannot be trained. Model details of PRPN as well as the tree extraction procedure will be discussed in Section 4.2.1.

To fix this problem, in this chapter, we propose an imitation learning approach that combines the continuous PRPN model with a discrete parsing model (i.e., Tree-LSTM), both trained without access to gold standard parse trees. We exploit the advantages of the PRPN (it is differentiable and supports backpropagation to syntactic distance) by transferring its knowledge to a discrete parser, which explicitly models tree-building operations. We accomplish the knowledge transfer by training the discrete parser to mimic the behavior of the PRPN. Then, the discrete parser refines its policy by straight-through Gumbel-Softmax (ST-Gumbel, Jang et al. 2017). We evaluate our approach on a Natural Language Inference dataset (Bowman et al., 2015), where the task is to classify the inference relationship between two sentences. Our approach outperforms previous latent tree induction models on this task in terms of parsing F-score, and also improves self-agreement. [1]

4.1 Related Work

4.1.1 Latent Tree Learning Through Downstream Tasks

Tree-structured recursive neural networks (Tree-RvNNs, Socher et al., 2011), especially tree-structured long short-term memory networks (Tree-LSTMs, Tai et al., 2015) have been shown to be effective at sentence understanding tasks such as sentiment analysis (Socher et al., 2011, 2013), textual entailment (Bowman et al., 2016) and machine translation (Eriguchi et al., 2016). Typically, previous work on tree-structured neural models assumes that tree structures are either provided together with the data set or produced by an off-the-shelf syntactic parser (e.g., the Stanford Parser, Klein and Manning 2003). Some variants of these models (Socher et al., 2011; Bowman et al., 2016) can also be trained on the existing annotations to parse unseen sentences that they consume at the inference phase.

As we discussed before, harnessing trees from supervised parsers suffers some limitations. Recent work focuses more on learning tree structures from the data in an unsupervised manner, which is called *latent tree learning*. Yogatama et al. (2017), for the first time, proposed to learn sentence-specific tree-based compositional architectures from a downstream task, rather than using explicit supervision from existing

[1] Our code is available at https://github.com/libowen2121/Imitation-Learning-for-Unsup-Parsing

annotations. The authors used reinforcement learning and took performance on the downstream task that used the computed sentence representation as the reward signal. Maillard et al. (2017) introduced the CYK chart parser (Cocke, 1969; Kasami, 1966; Younger, 1967) to Tree-LSTMs. Since the chart parser is fully differentiable, the proposed model can be trained end-to-end on a downstream task using stochastic gradient descent, although the chart parser also brings additional computational complexity. Choi et al. (2018) proposed Gumbel Tree-LSTMs, where a straight-through Gumbel-Softmax estimator (Jang et al., 2017) is utilized to decide the parent node at each step during the tree building process and calculate gradients of the discrete decision. However, regarding the parsing performance, these models cannot generate plausible constituency tree structures compared to human expert annotated gold parse trees. A systematic study (Williams et al., 2018a) shows that, the learned parsing strategies are not especially consistent across random restarts; the generated constituency trees do not resemble those of an annotated treebank (i.e., Penn Treebank).

Instead of single sentence modeling, Liu et al. (2018) introduced a model to compare two sentences by matching their latent constituents via a CKY chart for sentence matching tasks. Shen et al. (2019a) introduced Ordered Memory (OM), which includes a new memory updating mechanism and a new gated recursive cell, to induce tree structures on synthetic formal language datasets and sentiment analysis.

Besides the constituency trees, dependency trees have also been investigated in the research on latent tree learning. Kim et al. (2017) injected a graph-based CRF dependency parser into recurrent neural networks to bias the self-attention mechanism. Liu and Lapata (2018) proposed to use a variant of Kirchhoff's Matrix-Tree Theorem (Koo et al., 2007) to implicitly consider non-projective dependency tree structures for both sentence and document modeling, where major operations for tree builidng can be parallelized efficiently on GPUs. Corro and Titov (2019b) used differentiable dynamic programming, which allowed for efficient sampling (Corro and Titov, 2019a), to induce projective dependency trees for sentiment analysis and natural language inference tasks.

4.1.2 Latent Tree Learning Through Language Modeling

As a substitute for natural language understanding-based downstream tasks, language modeling has also been studied as an objective to learn latent tree structures. Shen et al. (2018b) proposed the parsing-reading-predict network (PRPN), which can si-

multaneously induce the constituency tree structures from the raw sentences. Then the inferred structures are leveraged to guide the self-attention of a long short-term memory-network (LSTMN, Cheng et al. 2016) to form a better neural language model. The LSTMN is an extension of the standard LSTM architecture, where a memory network takes place of a single memory cell. Htut et al. (2018) confirmed that the PRPN is able to produce meaningful tree structures when compared to the human annotated treebank. Shen et al. (2019b) further proposed a novel inductive bias *Ordered Neurons* and design a new RNN unit ON-LSTM, which enables RNN models to perform tree-like compositions without breaking its sequential form. Similar to the PRPN, syntactic distances are utilized for tree extraction. Such distances are computed from a specifically designed forgetting gate in the ON-LSTM, then a biased top-down parsing procedure (also used in the PRPN) is called to produce the parse tree. In the unsupervised parsing, the ON-LSTM achieves better performance than the PRPN. However, Dyer et al. (2019) pointed out that this biased top-down parser can only recover a fraction of all possible trees in theory. The authors also found that applying this parser, surprisingly, proxies derived from a conventional LSTM language model can produce trees comparably well to the specialized ON-LSTM. Aside from the conventional unidirectional language modeling, Wang et al. (2019) proposed the Tree Transformer, which integrates tree structures into bidirectional Transformer encoder for masked language modeling.

4.1.3 Imitation Learning

Reinforcement learning (RL; Sutton and Barto 2018) is a general-purpose framework that enables agents to reason about sequential decision-making as an optimization process. It usually describes how agents interact with unknown environment, where the goal is to select actions to maximize a future cumulative reward. With the rapid development of deep neural networks, deep reinforcement learning (Mnih et al., 2013, 2015; Silver et al., 2016) has been an active research area that agents learn their own knowledge directly from raw inputs through neural networks.

When reinforcement learning is applied to the latent tree learning problem, each decision step in the parsing process can be considered as an action. Typically, there is no instant reward or supervision for each parsing decision. The ultimate reward or the learning signal can only be obtained from the loss on a downstream task or language modeling, when the entire tree structure is built. It is one of the reasons why the

learning algorithm is fragile and the model is hard to train. Imitation learning (Schaal, 1999; Argall et al., 2009; Billard et al., 2016), as an approach to help alleviate this problem, has been investigated to efficiently and intuitively program autonomous behavior. Typical imitation learning includes two approaches: (1) directly replicating desired behavior from a teacher (a human or a well-trained model) via step-by-step supervised learning, which is called behavioral cloning (Bain and Sammut, 1995); (2) learning the underlying objectives of the desired behavior from demonstrations, which is called inverse optimal control (Kalman, 1964) or inverse reinforcement learning (Russell, 1998). Our approach adopts the first one and follows a two-step strategy: behavioral cloning and policy refinement. Policy refinement is required in our approach because the teacher is imperfect. Our experiments show the benefits of policy refinement in this case.

Our work in this chapter also follows the framework of Mou et al. (2017), who coupled neural and symbolic systems for semantic parsing by pre-training a reinforcement learning executor with neural attention. We extend this idea to syntactic parsing and show the relationship between parsing and downstream tasks.

4.1.4 Knowledge Distillation

Another approach related to our work in this chapter is knowledge distillation. Knowledge distillation (Buciluǎ et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015) has been first proposed to transfer the knowledge from a well-trained large neural network (always called the *teacher*) to a small neural network (always called the *student*). Given a set of training samples, the student network is trained with the true targets and the teacher's predicted probabilities as *soft targets*.

Knowledge distillation has been shown to improve performances for various NLP tasks. Kim and Rush (2016) employed the knowledge distillation for the sequence level knowledge transfer in neural machine translation systems. Hu et al. (2016) proposed a framework to enhance different types of neural networks with declarative first-order logic rules using knowledge distillation, which brings benefits to sentiment analysis and named entity recognition. Kuncoro et al. (2016) distilled an ensemble of greedy transition-based dependency parsers to a first-order graph-based dependency parser. More recently, knowledge distillation has been utilized on large-scale pre-trained transformers. A large volume of research work has explored the application of knowledge distillation to compress pre-trained transformers into smaller transformers or simpler

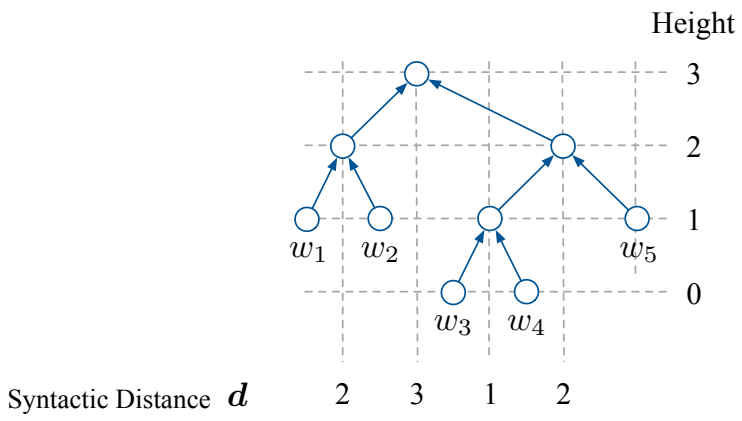


Figure 4.1: An example of a parse tree. The syntactic distance of two adjacent words is defined as the height of the common ancestor of these words in the parse tree.

architectures (e.g., BiLSTMs) in both task-specific (Tang et al., 2019; Turc et al., 2019; Sun et al., 2019a; Aguilar et al., 2020) and task-agnostic (Tsai et al., 2019; Sanh et al., 2019; Jiao et al., 2019; Wang et al., 2020) fashion.

In this chapter, we transfer the knowledge learned from a teacher model to a student model. However, in our case, the teacher and student models are heterogeneous. So we conduct knowledge transfer with predicted output rather than predicted *soft* probabilities. Moreover, our student has a stage of policy refinement, which typically does not exist in traditional distilling approaches.

4.2 Problem Formulation

4.2.1 Parsing-Reading-Predict Network

Models

The first ingredient of our approach is the parsing-reading-predict network (PRPN; Shen et al. 2018b), which is trained using a language modeling objective, i.e., it predicts the next word in the text based on previous words. Given a sentence $[w_1, ..., w_N]$ and corresponding input embeddings $[\mathbf{w}_1, ..., \mathbf{w}_N]$, the PRPN introduces the concept of *syntactic distance* d_t (illustrated in Figure 4.1), defined as the height of the common ancestor of the two consecutive words $(w_{t-1},\ w_t)$ in the parse tree , where t is the position index in a sentence. Since gold standard d_t is not available at training time, the PRPN uses a two-layer convolutional neural network (CNN) to estimate it with $\widehat{d_t}$. Specifically, the convolutional kernel window size for the first layer is L, which determines the

valid history context length. And the window size for the second layer is 1. The input is the embeddings of the current word $\mathbf{w}_t$ and its left context $\mathbf{w}_{t-L}, \mathbf{w}_{t-L+1}, \cdots, \mathbf{w}_{t-1}$. The output is given by

$$\widehat{d_t} = \mathrm{CNN}(\mathbf{w}_{t-L}, \mathbf{w}_{t-L+1}, \cdots, \mathbf{w}_t). \tag{4.1}$$

In fact, absolute distance values are not required, it is sufficient to preserve their order. In other words, if $d_i < d_j$, then it is desired that $\widehat{d_i} < \widehat{d_j}$. However, even the order of d_t is not available at training time, and $\widehat{d_t}$ is learned end-to-end in an unsupervised manner to optimize the language modeling objective.

The PRPN computes the difference between $\widehat{d_t}$ at the current step and all previous steps $\widehat{d_j}$ for $2 \leq j < t$. The differences are normalized to $[0,1]$:

$$\alpha_j^t = \frac{\mathrm{hardtanh}(\tau(\widehat{d_t} - \widehat{d_j})) + 1}{2} \tag{4.2}$$

where $\mathrm{hardtanh}(x) = \max(-1, \min(x, 1))$ and τ is the temperature. Finally, a soft gate is computed right-to-left in a multiplicatively cumulative fashion:

$$g_i^t = \prod_{j=i+1}^{t-1} \alpha_j^t \tag{4.3}$$

for $1 \leq i \leq t-1$. The gate g_i^t is used to reweight another intra-sentence attention $\widetilde{s}_i^t$, which is computed as:

$$\widetilde{s}_i^t = \mathrm{softmax}\left(\frac{\mathbf{h}_i^\top(\mathbf{W}[\mathbf{h}_{t-1}; \mathbf{w}_t])}{\sqrt{\delta_k}}\right), \tag{4.4}$$

where $\mathbf{h}_t$ is the hidden memory tape at time step t, δ_k is the dimension of the hidden state and $\mathbf{W}$ is a weight matrix. The reweighed intra-sentence attention s_i then becomes

$$s_i^t = \frac{g_i^t}{\sum_{i=1}^{t-1} g_i^t} \widetilde{s}_i^t. \tag{4.5}$$

It is then used to compute the convex combination of attention candidate vectors, which are incorporated to recurrently compute an adaptive memory representation to summarize information relevant to the current time step, shown in Figure 4.2. Finally, the next token is predicted based on all memories that are syntactically and directly relevant.

Tree extraction

The syntactic distances in the PRPN are positive real values, from which tree structures are inferred using an external procedure. Given the syntactic distances $\widehat{\mathbf{d}}$ estimated by

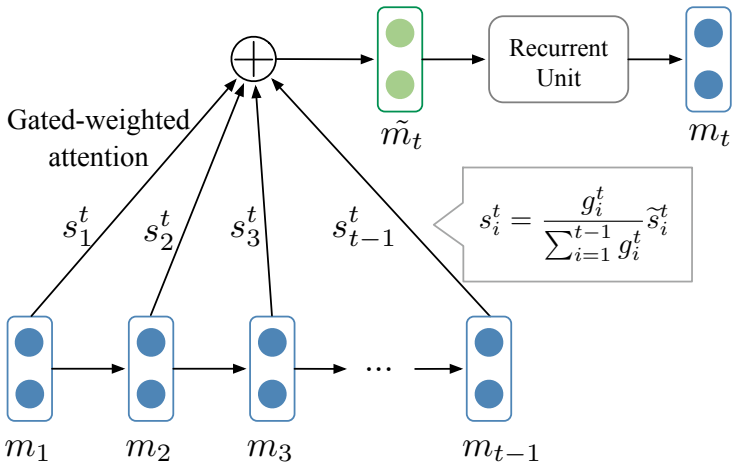


Figure 4.2: The update of the adaptive memory representation in the PRPN language model. m_t is the adaptive memory vector corresponding to the word w_t.

the PRPN, an intuitive tree extraction scheme is that the largest distance $\widehat{d}_i$ is found and the sentence is split into two constituents $(\cdots, w_{i-1})$ and $(w_i, \cdots)$. This process is then repeated recursively on the two new constituents. The trees inferred by this scheme, however, yield poor parsing performance. The results reported by Shen et al. (2018b) are actually obtained using a different scheme: find the largest syntactic distance $\widehat{d}_i$ and obtain two constituents $(\cdots, w_{i-1})$ and $(w_i, \cdots)$. If the latter constituent contains two or more words, then further split it into (w_i) and $(w_{i+1}, \cdots)$, regardless of the syntactic distance $\widehat{d}_{i+1}$. This scheme introduces a bias for right-branching trees, which presumably explains why it yields better parsing performance than the intuitive unbiased parsing scheme for English.

Dyer et al. (2019) called this right-branching biased parser $\overline{\text{COO}}$ parser. A $\overline{\text{COO}}$ parser can generate all binary trees that do not cover the bracketed string "...)((...". The avoidance of **c**lose-**o**pen-**o**pen leads to the name $\overline{\text{COO}}$. This parser only recovers a fraction of possible trees, because the ratio of the extractable parses to all possible binary parsers converges logarithmically to 0 as the sentence length grows. The reliance on this trick illustrates the point we make in this chapter: syntactic distance has the advantage of being a continuous value, which can be computed as an attention score in a differentiable model. However, this comes at a price that the PRPN does not model trees or tree-building operations directly. These operations need to be stipulated externally, in an ad-hoc inference procedure. This procedure is not part of the model and cannot be trained, but yet is crucial for good parsing performance.

4.2.2 Discrete Syntactic Parser

To address this problem, we combine the PRPN with a parser which explicitly models tree-building operations. Specifically, we use the Gumbel Tree-LSTM (Choi et al., 2018) shown in Figure 4.3 (a). Gumbel Tree-LSTM is a pyramid-shaped tree-based LSTM where reinforcement learning in this model is relaxed by Gumbel-Softmax.

Concretely, let $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_N$ be the embeddings of the words in a sentence. The model tries every possible combination of two consecutive words by Tree-LSTM, but then uses softmax (with $N-1$ ways) to predict which composition is appropriate at this step. Let $\mathbf{h}_1^{(1)}, \cdots, \mathbf{h}_{N-1}^{(1)}$ be the candidate Tree-LSTM composition at the bottom layer, where the superscript is the layer index. With $\mathbf{q}$ being a query vector, the model computes a distribution $\mathbf{p}$:

$$p_i^{(1)} = \text{softmax}\{\mathbf{q}^\top \mathbf{h}_i^{(1)}\}. \tag{4.6}$$

Assuming the model selects an appropriate composition at the current step, we copy all other words intactly, shown as orange arrows in Figure 4.3 (a). This process is applied recursively, forming the structure in the figure.

The Tree-LSTM model is learned by straight-through Gumbel-Softmax (ST-Gumbel; Jang et al. 2017). ST-Gumbel resembles reinforcement learning, where it samples actions from its predicted probabilities, exploring different regions of the latent space other than a maximum *posteriori* tree. In the forward propagation of ST-Gumbel training, the model samples an action — in the Tree-LSTM model, the position of composition — from the distribution $\mathbf{p}$ by the Gumbel trick. The sampled action can be represented as a one-hot vector $\mathbf{a}$, whose elements take the form:

$$a_i = \begin{cases} 1, & \text{if } i = \text{argmax}_j\{\log(p_j) + g_j\} \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

where g_i is the *Gumbel noise*, given by:

$$g_i = -\log(-\log(u_i)) \tag{4.8}$$

$$u_i \sim \text{Uniform}(0,1). \tag{4.9}$$

It is shown that $\mathbf{a}$ is an unbiased sample from the original distribution $\mathbf{p}$ (Jang et al., 2017).

During backpropagation, ST-Gumbel substitutes the selected one-hot action $\mathbf{a}$ given by argmax in Equation (4.7) with a softmax operation

$$\widetilde{p}_i = \frac{\exp\{(\log(p_i) + g_i)/\tau\}}{\sum_j \exp\{(\log(p_j) + g_j)/\tau\}}, \tag{4.10}$$

where τ is a temperature parameter that can also be learned by backpropagation. Continuous relaxations of one-hot vectors are suitable for scenarios where we are constrained to sampling discrete variables. But this gradient estimator is clearly biased, as described the biased path derivative estimator in Bengio et al. (2013). More specifically, in the Gumbel Tree-LSTM model, the gradient flow may not satisfy the tree constraint. It is likely to cause issues especially for long sentences. In some extreme case, the true gradient will have a chance to point in the opposite direction of the estimate.

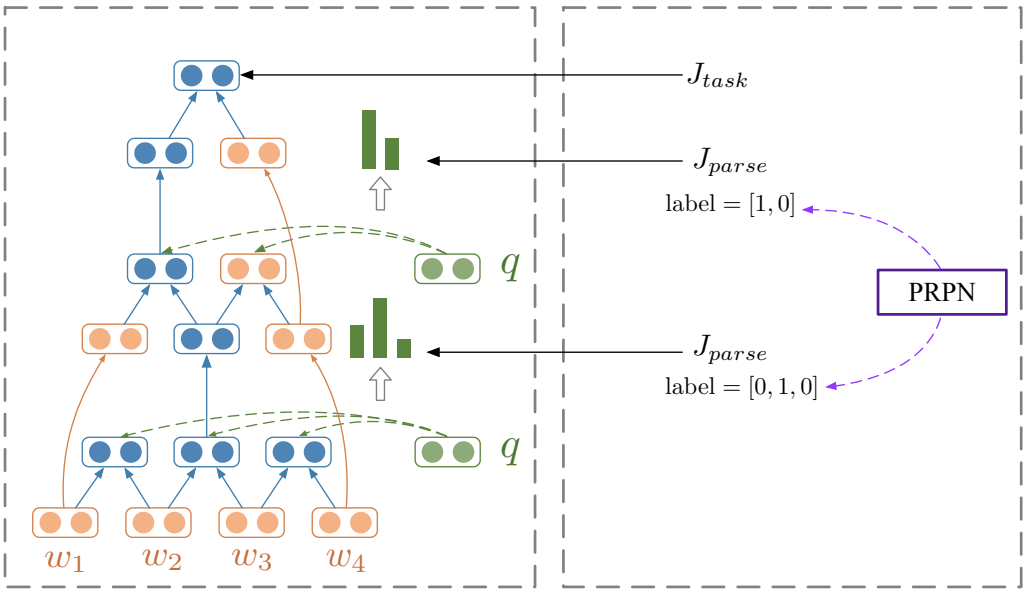
The Tree-LSTM model is trained using the loss in a downstream task (e.g., cross-entropy loss for classification problems). Compared with reinforcement learning, the ST-Gumbel trick allows more information to be propagated back to the bottom of the Tree-LSTM in addition to the selected actions, although it does not follow exact gradient computation. For prediction (testing), the model selects the most probable composition according to its predicted probabilities.

The training involves doubly stochastic gradient descent (Lei et al., 2016): the first stochasticity comes from sampling input from the data distribution, and the second stochasticity comes from sampling actions for each input. Therefore, ST-Gumbel is difficult to train (similar to reinforcement learning), and may be stuck in poor local optima, resulting in low self-agreement for multiple random initializations (Williams et al., 2018a).

4.2.3 Imitation Learning

Our aim is to combine the PRPN's continuous notion of syntactic distance and a parser with discrete tree-building operations. The mapping from the sequence of Tree-LSTM composition operations to a tree structure is not injective. Given a parse tree, we can have multiple different composition sequences. This ambiguity could confuse the Tree-LSTM during training. We solve this problem using the PRPN's notion of syntactic distance.

In a predicted parse tree, if more than one operation is applicable, we first group together the candidates with the lowest syntactic distance. For example, in Figure 4.1, w_3 and w_4 are first merged. For candidate pairs with the same syntactic distance, we set the composition order randomly among them. In this way, we can extract the composition order from the trees infered by a pre-trained PRPN. Then we train the Tree-LSTM model in a *step-by-step* (SbS) supervised fashion. Let $\widehat{\mathbf{t}}^{(j)}$ be a one-hot



(a) Gumbel Tree-LSTM (b) Imitation Learning

Figure 4.3: Overview of our approach. (a): we make use of the Gumbel Tree-LSTM model of Choi et al. (2018). (b): the model is first trained with step-by-step supervision, after which Gumbel-Softmax is applied to refine the policy.

vector for the jth step of Tree-LSTM composition, where the hat denotes imperfect target labels. The parsing loss is defined as:

$$J_{\text{parse}} = -\sum_j \sum_i \hat{t}_i^{(j)} \log p_i^{(j)}, \tag{4.11}$$

where $\mathbf{p}^{(j)}$ is the probability predicted by the Tree-LSTM model. The subscript i indexes the ith position among in $1, \ldots, N_j - 1$, where N_j is the number of nodes at the jth composition step. The overall training objective is a combination of the loss of the downstream task and the parsing loss (weighted by λ):

$$J = J_{\text{task}} + \lambda J_{\text{parse}} \tag{4.12}$$

After step-by-step training, we perform *policy refinement* to let the Tree-LSTM model improve its policy by ST-Gumbel. It should be emphasized that how the Tree-LSTM model builds the tree structure differs between step-by-step training and ST-Gumbel training. For step-by-step training, we assume an imperfect parsing tree is in place; therefore, the Tree-LSTM model exploits existing partial structures and predicts the next composition position. For ST-Gumbel, the tree structure is sampled from its predicted probability, enabling the model to explore the space of trees beyond the given imperfect tree.

4.3 Experiments

4.3.1 Datasets

We conduct experiments on the AllNLI dataset, the concatenation of the Stanford Natural Language Inference Corpus (SNLI; Bowman et al. 2015) and the Multi-Genre NLI Corpus (MultiNLI; Williams et al. 2018b), which is used in other latent tree learning work for its non-syntactic classification labels for the task of textual entailment. As the MultiNLI test set is not publicly available, we follow previous work (Williams et al., 2018a; Htut et al., 2018) and use the development set for testing. For early stopping, we remove 10k random sentence pairs from the AllNLI training set to form a validation set. Thus, our AllNLI dataset contains 931K, 10K, and 10K sample pairs for training, validation, and test, respectively.

4.3.2 Settings

We build the PRPN model and the Tree-LSTM parser following the hyperparameters in previous work (Shen et al., 2018b; Choi et al., 2018) using publicly released code bases [2] [3]. For the SbS training stage, we set λ to be 0.03. For policy refinement stage, the initial temperature is manually set to be 0.5. PRPN is trained using a language modeling loss on the AllNLI training sentences, whereas the Tree-LSTM model is trained using a cross-entropy loss for AllNLI classification. We adopt the standard metric and compute the unlabeled F-score of the constituents predicted by our parsing model against those given by the Stanford PCFG Parser 3.5.2 (Klein and Manning, 2003). Although the Stanford parser itself may make parsing errors, it achieves generally high performance and is a reasonable approximation of correct parse trees.

4.3.3 Experimental Results

Main Parsing Results

Parsing results are given in Table 4.1, where left-/right-branching and balanced trees are included as trivial baselines. The ST-Gumbel Tree-LSTM model and the PRPN were run for five times with different initializations. For imitation learning given a PRPN trajectory, we perform SbS training once, as well as policy refinement for five

[2] https://github.com/yikangshen/PRPN
[3] https://github.com/nyu-mll/spinn/tree/is-it-syntax-release

Model	Mean F	Self-agreement	RB-agreement
w/ Punctuation			
Left-Branching	18.9	-	-
Right-Branching	18.5	-	-
Balanced-Tree	22.0	-	-
ST-Gumbel	21.9	56.8	**38.1**
PRPN	51.6	65.0	27.4
Imitation (SbS only)	52.0	**70.8**	20.6
Imitation (SbS + refine)	**53.7**†	67.4	21.1
w/o Punctuation			
Left-Branching	20.7	-	-
Right-Branching	**58.5**	-	-
Balanced-Tree	39.5	-	-
ST-Gumbel	36.4	57.0	33.8
PRPN	46.0	48.9	51.2
Imitation (SbS only)	45.9	49.5	62.2
Imitation (SbS + refine)	53.3†	**58.2**	**64.9**

Table 4.1: Parsing performance with and without punctuation. Mean F indicates mean parsing F-score against Stanford Parser (early stop by F-score). Self-/RB-agreement indicates self-agreement and agreement with right-branching across multiple runs. † indicates statistically different from corresponding PRPN baseline using a paired one-tailed t-test.

runs. We evaluate two settings in which we keep and remove punctuation and report the average F-score against the Stanford Parser.

In the setting of keeping all punctuation, we see that the Tree-LSTM model, trained by ST-Gumbel from random initialization, does not outperform trivial baselines like balanced trees, whereas the PRPN outperforms them by around 30%. Our PRPN replication results are consistent with Htut et al. (2018).

Our first stage in imitation learning (SbS training) is able to successfully transfer the knowledge induced by the PRPN to the Tree-LSTM model, achieving an F-score of 52.0, which is clearly higher than the 21.9 achieved by Tree-LSTM trained with ST-Gumbel alone, and even slightly higher than the PRPN itself. The second stage, policy refinement, achieves a further improvement in latent tree induction, outperforming the PRPN by 2.1 F-score points. The difference between the PRPN baseline and policy

refinement is as significant as indicated by paired one-tailed t-tests (paired tests are appropriate, as the models in question use the same training trajectories).

We also evaluate the self-agreement by computing the mean F-score across 25 runs for policy refinement and 5 runs for other models. We see that our imitation learning achieves improved self-agreement in addition to parsing performance.

Effect of Punctuation

In the literature of unsupervised parsing, stripping off all punctuation is a normal setting. It is interesting to investigate whether removing punctuation makes the latent tree learning task easier. Table 4.1 also shows the parsing performance without punctuation. In the setting of without punctuation, our imitation learning approach with policy refinement can outperform the PRPN by a larger margin (7.3 F-score points) than the setting of with punctuation. But surprisingly, right-branching sets such a strong baseline that reaches the best parsing performance for this setting. Even using extra right-branching bias in the tree-decoding procedure, the PRPN still cannot outperform the right-branching baseline.

It makes sense that right-branching coincides better with real parsing structures when punctuation are removed. A simple example is that, an ending period is always attached to a high-level constituent while right-branching sets it to the bottom level and consequently causes a lot of errors.

We also compute the agreement with right-branching. In can be found that in the setting of without punctuation, the PRPN sets an initial policy that more or less agrees with right-branching, and the parsing policy coincides better with right-branching after imitation learning. However, in the setting of with punctuation, the right-branching agreement changes in an opposite way. We conjecture that right-branching is a reason why our imitation learning can get larger improvement over the PRPN without punctuation. Right-branching provides a relatively flat local optimum so that imitation learning can do further exploring with a low chance to jump out of it.

In our experiments, we evaluate the constituency trees predicted by our approach against non-gold parse trees, which are produced by the Stanford PCFG Parser 3.5 (Klein and Manning, 2003). This parser is trained on the standard training set as well as on the Brown Corpus (Francis and Kucera, 1979), which has been shown to improve the parse quality of the descriptive sentences and noun phrases (Bowman et al., 2015). In the AllNLI dataset, the sentence length is relatively short (14.9 on average), which makes it easy for a parser. The Stanford parser achieves generally high performance

Type	# Occur	ST-Gumbel	PRPN	Imitation (SbS + refine)
NP	69k	22.6	**53.2**	49.5
VP	58k	4.9	49.4	**57.0**
S	42k	44.3	63.9	**66.0**
PP	29k	13.9	**55.4**	52.4
SBAR	12k	6.9	38.9	**41.4**
ADJP	4k	10.6	44.2	**46.5**

Table 4.2: Parsing accuracy for six phrase types which occur more than 2k times in the MultiNLI development set with keeping punctuation.

and is a reasonable approximation of correct parse trees. It is also the default parser adopted in the AllNLI dataset (Bowman et al., 2015; Williams et al., 2018b). However, the Stanford parser itself may still make parsing errors. It's possible that the Stanford parser produces some bias that our approach could take advantage of (like the right-branching pattern in the experiment without punctuation). If it happens, the reported parsing performance of our approach is thus probably somewhat inflated.

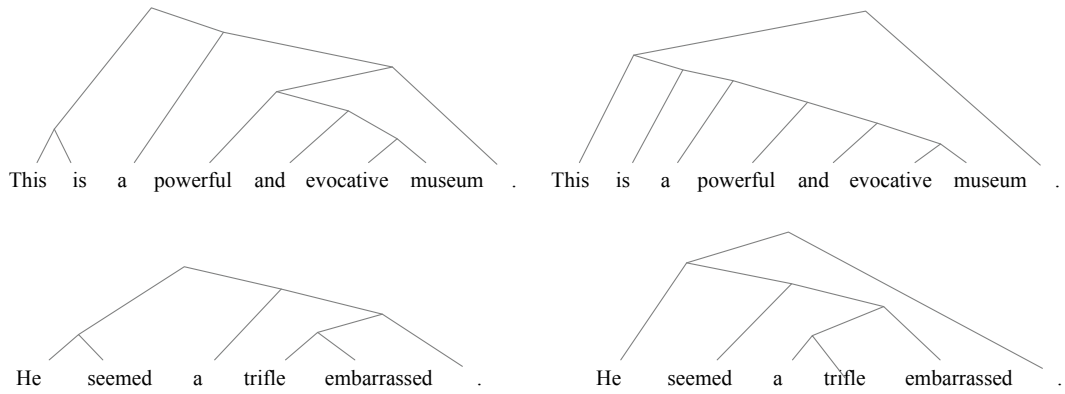
Parsing Performance across Constituent Types

We break down the performance of latent tree induction across constituent types for the setting of keeping punctuation. From Table 4.2, we see that, among the six most common ones, our imitation approach outperforms the PRPN on four types. However, we also notice that for the most frequent type (NP), our approach is 4% worse than the PRPN. This shows that the strengths of the two approaches complement each other to some extent, and in future work ensemble methods may be employed to combine them.

Classification Performance on NLI

Our results show the usefulness of a downstream task for latent tree induction: step-by-step imitation learning with policy refinement improves the parsing performance of a state of the part latent tree model such as the PRPN. This provides evidence against previous studies where researchers claim that an external, non-syntactic task such as NLI does not improve parsing performance (Williams et al., 2018a; Htut et al., 2018).

One natural question is how the learned models perform on the NLI task. Our results are compatible with findings of Shi et al. (2018) that a range of different tree



(a) Tree examples of PRPN (b) Tree examples of our model (SbS + refine)

Figure 4.4: Parse tree examples of PRPN and our model (SbS + refine).

structures yield similar classification accuracy on NLI. The NLI mean accuracy of ST-Gumbel and our imitation learning with policy refinement on AllNLI development set in a setting of keeping punctuation is 69.9% and 69.2% respectively, which means the NLI classification performance of our approach remains the same level as ST-Gumbel.

An interesting follow-up question is why ST-Gumbel improves latent tree induction when trained with an NLI objective. It has been argued that NLI as currently formulated is not a difficult task (Poliak et al., 2018); this is presumably why models can perform well across a range of different tree structures (see above), only some of which are syntactically plausible. However, this does not imply that the Tree-LSTM model will learn nothing when trained with NLI. We can think of its error surface being very rugged with many local optima; the syntactically correct tree corresponds to one of them. If the model is initialized in a meaningful catchment basin, then NLI training is more likely to recover that tree. The intuition also explains why the Tree-LSTM model alone achieves low parsing performance and low self-agreement. On a very rugged high-dimensional error surface, the chance of getting into a particular local optimum (corresponding to a syntactically correct tree) is low, especially in reinforcement learning and ST-Gumbel that are doubly stochastic.

Parse Tree Examples

In Figure 4.4, we present a few parse tree examples generated by the PRPN and our model (SbS + refine). As can be seen, our model is able to handle the period correctly in the examples. Although this could be specified by human-written rules, it is in fact learned by our approach in an unsupervised manner, since a punctuation mark is

treated as a token like other words and our training signal offers no clue regarding how a punctuation mark should be processed.

Moreover, our model is able to parse verb phrases more accurately than the PRPN, such as "is a powerful and evocative museum" and "seemed a trifle embarrassed". This is also evidenced by quantitative measures in Table 4.2.

4.4 Summary

In this chapter, we proposed a novel way of inducing syntactic structure from downstream tasks. We started from a state-of-the-art latent tree induction model learned from a language modeling objective, the PRPN model of Shen et al. (2018b). The structured attention mechanism in the PRPN is continuous and fully differentiable although explicit tree structures are extracted through a non-trainable biased procedure. We pre-trained a PRPN on the raw sentences from a downstream task (i.e., Natural language Inference, NLI) and transfered the knowledge induced by the PRPN to a discrete tree-structured model, the Tree-LSTM, using step-by-step imitation learning. We then used straight-through Gumbel-Softmax gradient estimator trained against the NLI objective to refine the parsing policy of the Tree-LSTM.

Our step-by-step imitation learning as well as the policy refinement resulted in an improvement of around two points in parsing F-score compared with the PRPN model. We also improved the self-agreement at the same time. Our work revealed a new angle towards the latent tree learning problem and provided evidence against previous work that a downstream, non-syntactic task such as NLI does not improve parsing performance (Williams et al., 2018a; Htut et al., 2018). Kim et al. (2019a) also showed that a pre-trained recurrent neural network grammar (RNNG, Dyer et al. 2016) can produce better parsing performance when further fine-tuned on the unsupervised RNNG (URNNG, Kim et al. 2019b) objective, the marginal sentence likelihood.

By observing the generated tree structures, we find that different models are effective at identifying different constituents. In future work, we would like to combine more potential parsers — including chart-style parsing and shift-reduce parsing — and transfer knowledge from one to another in a co-training setting.

Our approach has the potential to be a general framework to couple two heterogeneous neural latent tree learning models, where adaptation should be especially considered. For example, given a pre-trained soft tree model, the syntactic knowledge can be transferred to a student chart parser-based model (e.g., Maillard et al. 2017). In

the knowledge transfer training phase, given the trees produced by the soft tree model, the chart parser can be trained by minimizing the hinge loss Gaddy et al. 2018; Kitaev and Klein 2018. Beyond constituency trees, our approach can also be applied to latent dependency tree learning. For instance, suppose some imperfect dependency tree annotations are available to set up good initialization for a latent dependency tree model (e.g., Corro and Titov 2019b), the parser can be first learned by maximizing the log-likelihood of annotated trees (Lafferty et al., 2001) and then be optimized on downstream tasks.

Chapter 5

Unsupervised Parsing via Pre-trained Language Models

Transformer-based pre-trained language models (PLMs), particularly BERT (Devlin et al., 2019) and others (Yang et al., 2019; Liu et al., 2019c; Radford et al., 2019), have dramatically improved the state of the art in NLP. Such models make it possible to train a large generic language model on vast unannotated datasets, and fine-tune it for a specific task using a small amount of annotated data. The success of PLMs has led to a large literature investigating the linguistic knowledge that PLMs learn implicitly during pre-training (Liu et al., 2019a; Clark et al., 2019; Kovaleva et al., 2019; Pimentel et al., 2020), sometimes referred to as BERTology (Rogers et al., 2020).

BERTology has been particularly concerned with the question whether BERT-type models learn syntactic structure. Typical approaches include test suites of sentences that instantiate specific syntactic structures (Goldberg, 2019; Warstadt et al., 2019; Ettinger, 2020), general probes (also known as diagnostic classifiers, Belinkov and Glass 2019) or specifically designed structural probes (Hewitt and Manning, 2019). All of these approaches are limited: the first one requires the laborious compilation of language- and construction-specific suites of sentences; the second one sometimes fails to adequately reflect differences in representations (Zhang and Bowman, 2018; Hewitt and Liang, 2019); the third one involves designing a novel extraction model that focuses on a specifically designed novel metric, but it lacks justification (Hall Maudslay et al., 2020).

It is therefore natural to use a parsing task to test whether the representations learned by PLMs contain usable syntactic information. This enables us to test syntactic structure in general, rather than specific constructions, and doesn't require a spe-

cialized probe. In this chapter, we propose to construct an *unsupervised constituency parser* by using attention heads in PLMs. Previously, related approaches have been proposed under the heading of *zero-shot* constituency parsing (Kim et al., 2020a,b).[1] However, this prior work crucially relies on an annotated development set for feature selection, identifying transformer heads that are sensitive to syntactic structures. [2] Assuming a development set is not a realistic experimental setup (Kann et al., 2019). For most low resource languages, no such annotated data is available, and often not even an annotation scheme exists. A recent study (Shi et al., 2020) showed that, if a suitable development set is available, an existing supervised parser trained on a few-shot setting can outperform strong unsupervised parsing methods by a significant margin.

In this chapter, we propose a novel approach to build a PLM-based unsupervised parser that requires no annotated development sets: we rank transformer heads based on their inherent properties, such as how likely tokens are to be grouped in a hierarchical structure. We then ensemble the top-K heads to produce constituency trees. We evaluate our approach and previous zero-shot approaches on the English Penn Treebank (PTB) and eight other languages on the SPMRL dataset. If the development set is absent, our approach largely outperforms previous zero-shot approaches on the English PTB. On the other hand, if previous zero-shot approaches are equipped with a development set, our approach can still match the parsing performance of these approaches that use the single best head or layer-wise ensemble. For the multilingual experiment, we take advantage of the top-K heads selected in English and directly parse other languages using our approach. On five out of nine languages, this *crosslingual* unsupervised parser matches previous approaches that rely on a development set in each target language with the single best head or layer-wise ensemble. However, our fully unsupervised method lags behind the previous state-of-the-art zero-shot parser if a top-K ensemble is used.

Furthermore, our approach can be use as a tool to analyze the capability of PLMs in learning syntactic knowledge. As no human annotation is required, our approach has the potential to reveal the grammars PLMs have learned implicitly. Here, we learn neural probabilistic context-free grammars (PCFGs) from the trees induced from PLMs using our approach. We study the learned constituency grammars by comparing them

[1] Like Kim et al. (2020b), we use *zero-shot* to refer to the transfer from language modeling to constituency parsing.

[2] Although it is not rigorously practical to assume the existence of an annotated develop set, a number of previous studies (Shen et al., 2018b, 2019b; Kim et al., 2019a; Drozdov et al., 2019, 2020) have used such a set for hyperparameter tuning or early-stopping.

against the English PTB. Quantitatively, we evaluate the internal tags (both preterminal and nonterminal tags) against the English PTB. Qualitatively, we first visualize the alignment of preterminals and nonterminals of the learned grammars and the gold labels; then we showcase parse trees to illustrate some characteristics of the learned grammars.

5.1 Related Work

In this section, we first review the recent work on unsupervised constituency parsing via neural latent variable models and neural language models. We then discuss emerging studies on the interpretation of pre-trained language models.

5.1.1 Unsupervised Constituency Parsing via Neural Latent Variable Models

Thanks to their superior capability of distributed representation learning and overparameterization (Arora et al., 2018; Du et al., 2019), neural networks have recently renewed interest in unsupervised constituency parsing. Neural models with constituency trees as latent variables are one major approach, where the EM algorithm is used for optimization. Earlier work (Yogatama et al., 2017; Maillard et al., 2017; Choi et al., 2018) attempted to induce grammar by optimizing a sentence classification objective, but this has been proven to be ineffective in parsing (Williams et al., 2018a). Follow-up work (Shen et al., 2018b) showed that a language modeling objective is more suitable for unsupervised parsing and such models were claimed to be able to generate meaningful tree structures (Htut et al., 2018).

Probabilistic context-free grammars (PCFGs) were generally used for grammar induction in earlier days. The standard way to parameterize a PCFG is to simply associate a scalar with each production rule. This *direct parameterization* is algorithmically convenient but struggles to learn meaningful grammars from natural language data (Carroll and Charniak, 1992). Kim et al. (2019a) introduced neural parameterized PCFGs to the latent tree model to overcome this issue and further enhanced the model via latent sentence vectors to reduce the independence assumptions. Furthermore, Zhu et al. (2020) brought lexical dependencies to PCFGs and proposed a unified framework for both constituency and dependency grammar induction. Another model, the unsupervised recurrent neural network grammar (URNNG, Kim et al. 2019b), uses varia-

tional inference over latent trees to perform unsupervised optimization of the RNNG (Dyer et al., 2016).

On the other hand, Drozdov et al. (2019) proposed the deep inside outside recursive autoencoder (DIORA) to compute two representations for each cell in the chart by both bottom-up inside pass and top-down outside pass. DIORA optimizes an autoencoder objective such that the outside representation for each leaf cell in the tree should reconstruct the corresponding leaf input word, analogous to masked language modeling (Devlin et al., 2019). The authors extended DIORA to obtain an improved variant, S-DIORA (Drozdov et al., 2020), to encode a single tree rather than a weighted mixture of trees by using a hard argmax operation and a beam at each cell in the chart.

5.1.2 Extracting Trees from Neural Language Models

Another thread of work is to extract constituency tree structures from neural language models. One way is to manipulate the conventional neural language models to ease the extraction of tree structures. It involves injecting specifically designed tree structure-sensitive components to neural language models or fine-tuning the neural language models to meet linguistic plausibility oriented criteria. Shen et al. (2019b) introduced the Ordered Neuron LSTM (ON-LSTM) model, which is a modified LSTM language model where the forgetting gate typically respects the constituent boundary. Wang et al. (2019) proposed the Tree Transformer, which introduces an extra constraint to attention heads of the Transformer-based language model in order to encourage the attention heads to follow tree structures. Cao et al. (2020) introduced an approach to unsupervised parsing based on the linguistic notion of a constituency test. The authors designed an unsupervised parser by specifying a set of transformations and fine-tuning a transformer-based pre-trained language model as an unsupervised neural acceptability model to make grammaticality decisions. In this approach, given a sentence, tree structures are obtained by aggregating its constituency test judgments and minimum risk decoding. Enhanced by refinement and URNNG fine-tuning, this approach has substantially outperformed the previous best model and reached 71.3 F_1 score [3] on the English Penn Treebank, approaching the performance of the supervised binary RNNG + URNNG with a gap of only 1.5 points.

Instead of intervening in the neural language models for tree extractions, there is also research work focusing on tree extraction in a *parameter-free* manner. With the

[3]This result is obtained by selecting the best model from multiple runs using labeled data, where the mean score is 67.9 F_1. We refer readers to the original paper for more information.

rapid development of BERT-type models, approaches in this line have put special effort into PLMs and also conducted the interpretability study on the syntactic knowledge learned by PLMs. Zero-shot constituency parsing, whose goal is to automatically extract trees from PLMs in a parameter-free fashion, was proposed by Kim et al. (2020a). This parser utilizes the concept of *syntactic distance* (Shen et al., 2018a), where trees are induced by an algorithm that recursively splits a sequence of words in a top-down manner. However, this approach suffers from its greedy search mode, failing to take into account all possible subtrees. The chart-based zero-shot parser (Kim et al., 2020b) applies chart parsing to address this problem. One major drawback for these methods is that they heavily require a separately annotated development set to select the best parsing configuration. Wu et al. (2020) introduced a parameter-free probing technique to analyze PLMs via perturbed masking on constituency, dependency as well as discourse structures. Apart from neural language models, there is also research work on extracting constituency trees from the Transfomer-based neural machine translation system. Mareček and Rosa (2018) proposed heuristic approaches to convert attention weights to trees. They further introduced a chart-based tree extraction method in transformer-based neural machine translation encoders and provided a quantitative study (Mareček and Rosa, 2019).

Our work in this chapter falls in the research line of *parameter-free* tree extraction from the PLMs. Particularly, we rank transformer attention heads based on their inherent properties, and create an ensemble of high-ranking heads to produce the final tree. In this way, our ranking-based parser can work in a fully unsupervised manner where only raw sentences are required.

5.1.3 Interpretation of PLMs

Contextualized embeddings obtained from PLMs such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019c), XLNet (Yang et al., 2019) and ALBERT (Lan et al., 2020) have recently obtained the state of the art on a variety of NLP tasks including both natural language understanding and generation tasks. PLMs are trained on large amounts of unlabeled raw text and subsequently fine-tuned on downstream supervised tasks. Unlike convolutional or recurrent neural networks, the Transformers have little cognitive motivation, and the size of PLMs limits the ability to perform ablation studies on the pre-training phase. Thus, numerous studies recently have attempted to reveal the reasons behind the impressive performance of PLMs. Such studies are often dubbed

BERTology (Rogers et al., 2020).

Attention is widely considered to be essential for understanding Transformer models. Several studies (Raganato and Tiedemann, 2018; Clark et al., 2019; Kovaleva et al., 2019) have investigated the regular patterns of attention in PLMs: attending to the token itself, to previous/next tokens, to the end of the sentence, to special tokens (e.g., `[SEP]`, `[CLS]`, punctuation) and broadly over the entire input sequence.

From a linguistic point of view, researchers are interested in the linguistic knowledge, especially syntactic knowledge, encoded in the PLMs. Artificially crafted test suites or datasets are a popular approach to manipulate features of interests to analyze PLMs. Goldberg (2019) constructed test suites using naturally-occurring and manually crafted stimuli for subject-verb agreement and reflexive anaphora phenomena. Warstadt et al. (2019) used a single linguistic phenomenon, negative polarity item (NPI) licensing in English, to perform a case study for BERT. Ettinger (2020) introduced a suite of diagnostics drawn from psycholinguistic studies to better understand the linguistic competencies acquired by BERT.

Another commonly employed approach in this field is supervised probes, also known as diagnostic classifiers (Belinkov and Glass, 2019). Specifically, auxiliary supervised models, such as linear functions and multi-layer perceptrons (MLPs), are trained from a constrained view of the representation to predict linguistic properties like part-of-speech, morphological information, syntactic and semantic information (Peters et al., 2018b; Tenney et al., 2019b; Liu et al., 2019a). However, as long as a representation is a lossless encoding, a sufficiently expressive probe with enough training data can learn *any* task on top of it. When a probe achieves high accuracy on a linguistic task, it is still hard to conclude that the representation encodes corresponding linguistic knowledge. Zhang and Bowman (2018) first raised this issue and put probing accuracy in context using random representation baselines. Hewitt and Liang (2019) introduced *control tasks*, which associate word types with random outputs, to complement linguistic tasks. In this way, a good probe that reflects the representation, should achieve high probing accuracy and low control task accuracy. Pimentel et al. (2020) argued that one should always select the highest performing probe one can, even if it is more complicated. Because it produces a tighter estimate of the mutual information between a linguistic property and BERT representation. Voita and Titov (2020) proposed the information-theoretic probing with minimum description length to additionally consider the amount of effort that a probe needs to reach the probing accuracy.

Besides supervised probes for general purposes, Hewitt and Manning (2019) proposed a structural probe, which evaluates whether syntax trees are embedded in a linear transformation of a neural model's representation space. The probe identifies a linear transformation under which squared L2 distance encodes the distance between words in the dependency tree, and one in which squared L2 norm encodes depth in the tree. Hall Maudslay et al. (2020) showed that a more traditional parser with an identical lightweight parameterization as this structural probe is able to identify more syntax under a commonly used metric in dependency parsing, undirected unlabeled attachment score (UUAS), while the structural probe outperforms the parser on a novel metric proposed in Hewitt and Manning (2019). The authors (Hall Maudslay et al., 2020) therefore argued that a new metric should be clearly justified when it is applied in probing.

5.2 Zero-shot Constituency Parsing via PLMs

In this section, we briefly review the chart-based zero-shot parser and then introduce our ranking-based zero-shot parser.

5.2.1 Chart-based Zero-shot Parsing

In chart-based zero-shot parsing, a real-valued score $s_{tree}(\mathbf{t})$ is assigned for each tree candidate $\mathbf{t}$, which decomposes as:

$$s_{tree}(\mathbf{t}) = \sum_{(i,j)\in\mathbf{t}} s_{span}(i,j), \tag{5.1}$$

where $s_{span}(i,j)$ is the score (or cost) for a constituent that is located between positions i and j ($1 \leq i \leq j \leq n$, where n is the length of the sentence). Specifically, for a span of length 1, $s_{span}(i,j)$ is defined as 0 when $i = j$. For a span longer than 1, the following recursion applies:

$$s_{span}(i,j) = s_{comp}(i,j) + \min_{i\leq k<j} s_{split}(i,k,j) \tag{5.2}$$

$$s_{split}(i,k,j) = s_{span}(i,k) + s_{span}(k+1,j), \tag{5.3}$$

where $s_{comp}(\cdot,\cdot)$ measures the validity or compositionality of the span (i,j) itself, while $s_{split}(i,k,j)$ indicates how plausible it is to split the span (i,j) at position k. Two alternatives have been developed in Kim et al. (2020b) for $s_{comp}(\cdot,\cdot)$: the pair score function $s_p(\cdot,\cdot)$ and the characteristic score function $s_c(\cdot,\cdot)$.

The pair score function $s_p(\cdot,\cdot)$ computes the average pair-wise distance in a given span:

$$s_p(i,j) = \frac{1}{\binom{j-i+1}{2}} \sum_{(w_x,w_y)\in \texttt{pair}(i,j)} f(g(w_x), g(w_y)), \tag{5.4}$$

where $\texttt{pair}(i,j)$ returns a set consisting of all combinations of two words (e.g., w_x, w_y) inside the span (i,j).

Functions $f(\cdot,\cdot)$ and $g(\cdot)$ are the distance measure function and the representation extractor function, respectively. For g, given l as the number of layers in a PLM, g is actually a set of functions $g = \{g^d_{(u,v)} | u = 1,\ldots,l, v = 1,\ldots,a\}$, each of which outputs the attention distribution of the vth attention head on the uth layer of the PLM. The hidden representations of the given words can also serve as an alternative for g. But Kim et al. (2020a) showed that the attention distributions provide more syntactic clues under the zero-shot setting. In case of the function f, there are also two options, Jensen-Shannon (JSD) and Hellinger (HEL) distance. Thus, $f = \{\text{JSD}, \text{HEL}\}$.

The characteristic score function $s_c(\cdot,\cdot)$ measures the distance between each word in the constituent and a predefined characteristic value $\mathbf{c}$ (e.g., the center of the constituent):

$$\begin{aligned} s_c(i,j) &= \frac{1}{j-i+1} \sum_{i\le x\le j} f(g(w_x), \mathbf{c}), \\ \mathbf{c} &= \frac{1}{j-i+1} \sum_{i\le y\le j} g(w_y). \end{aligned} \tag{5.5}$$

Since $s_{comp}(\cdot,\cdot)$ is well defined, it is straightforward to compute every possible case of $s_{span}(i,j)$ using the CKY algorithm (Cocke, 1969; Kasami, 1966; Younger, 1967). Finally, the parser outputs $\hat{\mathbf{t}}$, the tree that requires the lowest score (cost) to build, as a prediction for the parse tree of the input sentence: $\hat{\mathbf{t}} = \text{argmin}_{\mathbf{t}}\, s_{tree}(\mathbf{t})$.

For attention heads ensemble, both a layer-wise ensemble and a top-K ensemble are considered. The first one averages all attention heads from a specific layer, while the second one averages the top-K heads across different layers. At test time, given an input sentence and selected K heads, K separate chart matrices are first obtained, and then each chart matrix is converted into the corresponding syntactic distance vector (described in Algorithm 2). Finally, the average of the syntactic distance vectors is computed and translated into the final parse tree (described in Algorithm 3). In practice, this ensemble approach, marrying chart-based parser and top-down parser, yields better performance than simply averaging the attention distributions. The chart-based zero-shot parser achieves the state of the art in zero-shot constituency parsing.

Algorithm 2: Chart to Syntactic Distance (from Kim et al. 2020b)

n: length of an input sentence S.
$C \in \mathbb{R}^{n \times n}$: a chart matrix whose elements are $s_{span}(i, j)$.
$P \in \mathbb{R}^{n \times n}$: a matrix, whose $(i, j)^{th}$ element is the split point of the $span(i, j)$ of the sentence S.
s: index of the start position, initialized as 1.
e: index of the end position, initialized as n.

```
Function C2D(C, P, s, e):
    if s = e then
        return [ ]                                   // (empty vector)
    else
        v ← C[s][e]
        p ← P[s][e]
        return [ C2D(C, P, s, p); v; C2D(C, P, p+1, e) ]   // ([· ; ·]:
         vector concatenation)
    end
```

Algorithm 3: Syntactic Distance to Binary Constituency Tree (from Kim et al. 2020a)

$S = [w_1, w_2, ..., w_n]$: words in a sentence of length n.
$d = [d_1, d_2, ..., d_{n-1}]$: a vector whose elements are the syntactic distances between every two adjacent words.
`Leaf(·)`: return a leaf node given a token.
`Node(·, ·)`: return a node given two children.

```
Function D2T(S, d):
    if d = [] then
        node ← Leaf(S[0])
    else
        i ← argmax(d)
        child_l ← D2T(S_≤i, d_<i)
        child_r ← D2T(S_>i, d_>i)
        node ← Node(child_l, child_r)
    end
    return node
```

5.2.2 Ranking-based Zero-shot Parsing

The chart-based zero-shot parser relies on the existing development set of a treebank (e.g., the English PTB) to select the best configuration, i.e., the combination of $\{g \mid g^d_{(u,v)}, u = 1, \ldots, l, v = 1, \ldots, a\}$, $\{f \mid \text{JSD}, \text{HEL}\}$, $\{s_{comp} \mid s_p, s_c\}$, and heads ensemble that achieves the best parsing accuracy. Such a development set always contains hundreds of sentences, hence considerable annotation effort is still required. Another argument against using a development set is that the linguistic assumptions inherent in the expert annotation required to create the development set potentially restrict our exploration of how PLMs model the constituency structures. It could be that the PLM learns valid constituency structures, which however do not match the annotation guidelines that were used to create the development set.

Here, we take a radical departure from the previous work in order to extract constituency trees from PLMs in a fully unsupervised manner. We propose a two-step procedure for unsupervised parsing: (1) identify syntax-related attention heads directly from PLMs without relying on a development set of a treebank; (2) ensemble the selected top-K heads to produce the constituency trees. Figure 5.1 illustrates the pipeline of our ranking-based zero-shot parser.

For identification of the syntax-related attention heads, we rank all heads by scoring them with a chart-based ranker. We borrow the idea of the chart-based zero-shot parser to build our ranker. Given an input sentence and a specific choice of f and s_{comp}, each attention head $g^d_{(u,v)}$ in the PLM yields one unique attention distribution. Using the chart-based zero-shot parser in Section 5.2.1, we can obtain the score of the best constituency tree as:

$$s_{parsing}(u,v) = s_{tree}(\hat{\mathbf{t}}) = \sum_{(i,j)\in\hat{\mathbf{t}}} s_{span}(i,j), \tag{5.6}$$

where $\hat{\mathbf{t}} = \text{argmin}_{\mathbf{t}}\, s_{tree}(\mathbf{t})$.

Our ranking method works approximately as a maximum a posteriori probability (MAP) estimate, since we only consider the best tree the attention head generates. In unsupervised parsing, marginalization is a standard method for model development. We have tried to apply marginalization to our ranking algorithm where all possible trees are considered and the sum score is calculated (using the `logsumexp` trick) for ranking. But marginalization does not work well for attention distributions, where an *attending-broadly* head with higher entropy is more favourable under this measurement than a syntax-related head with lower entropy. For this reason, we only consider the

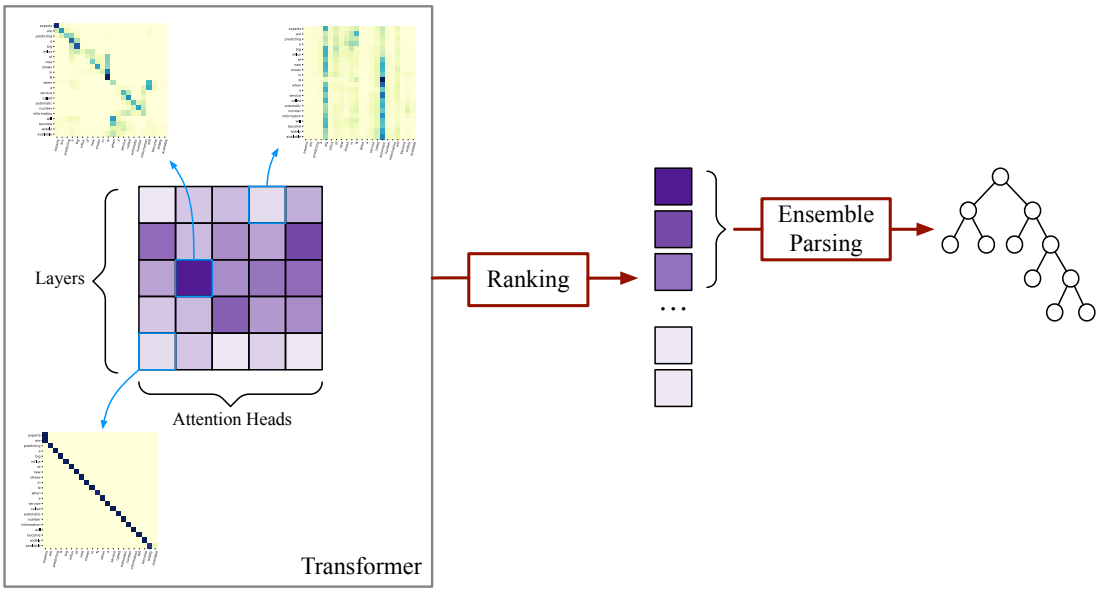


Figure 5.1: Our proposed two-step approach to unsupervised parsing. The purple heat map corresponding to the Transformer illustrates how likely a specific attention head is syntax-related. Each head corresponds to a self-attention distribution (given a sample sentence), showcased by three small heat maps. Intuitively, syntax-related heads show patterns like local chunking, while syntax-unrelated heads show other patterns like diagonals, columns or other noisy patterns.

score of the best parse tree.

It is obvious that all combinations of $\{f \mid \text{JSD}, \text{HEL}\}$ and $\{s_{comp} \mid s_p, s_c\}$ will produce multiple scores for a given head. Here we average the scores of all such combinations to get one single score. Then we rank all attention heads and select the syntax-related heads for parsing. However, directly applying the chart-based zero-shot parser in Section 5.2.1 for ranking delivers a trivial, ill-posed solution. The recursion in Equation (5.3) only encourages the intra-similarity inside the span. Intuitively, one attention head that produces the *same* attention distribution for each token (e.g., a uniform attention distribution or one that forces every token to attend to one specific token) will get the lowest score (cost) and the highest ranking. Such cases do exist in PLMs. Clark et al. (2019) showed that BERT exhibits clear surface-level attention patterns. Some of these patterns will deliver ill-posed solutions in ranking: attending broadly, attending to a special tokens (e.g., `[SEP]`), attending to punctuation (e.g., period). Figure 5.2b showcases some ill-posed patterns.

To address this issue, we first introduce inter-similarity into the recursion in Equa-

tion (5.3) and get the following:

$$s_{split}(i,k,j) = s_{span}(i,k) + s_{span}(k+1,j) - s_{cross}(i,k,j), \tag{5.7}$$

where the cross score $s_{cross}(i,k,j)$ is the similarity between two subspans (i,k) and $(k+1,j)$. However, this formulation forces the algorithm to go to the other extreme: one attention head that produces a *totally different* distribution for each token (e.g., force each token to attend to itself or the previous/next token) will get the highest ranking, which is confirmed in Figure 5.2c. To balance the inter- and intra-similarity and avoid introducing a tunable coefficient, we simply add a length-based weighting term to Equation (5.2) and get:

$$s_{span}(i,j) = \frac{j-i+1}{n}(s_{comp}(i,j) + \min_{i \le k < j} s_{split}(i,k,j)), \tag{5.8}$$

where $j-i+1$ is the length of the span (i,j). The length ratio functions as a regulator to assign larger weights to longer spans. This is motivated by the fact that longer constituents should contribute more to the scoring of the parse tree, since the inter-similarity always has stronger effects on shorter spans. In this way, the inter- and intra-similarity can be balanced. Figure 5.2d shows the top-6 heads selected by our ranking algorithm. Compared to the syntax-related heads decided by the single head parsing performance in Figure 5.2a, although our ranking algorithm is still not perfect, it can successfully identify the syntax-related properties.

With respect to the choice for $s_{cross}(i,k,j)$, we follow the idea of s_p and s_c in Equation (5.4) and (5.5) and propose the pair score function $s_{p\text{x}}$ and the characteristic score function $s_{c\text{x}}$ [4] for cross score computation. $s_{p\text{x}}$ is defined as:

$$s_{p\text{x}}(i,j) = \frac{1}{(k-i+1)(j-k)} \sum_{(w_x,w_y) \in \texttt{prod}(i,k,j)} f(g(w_x), g(w_y)), \tag{5.9}$$

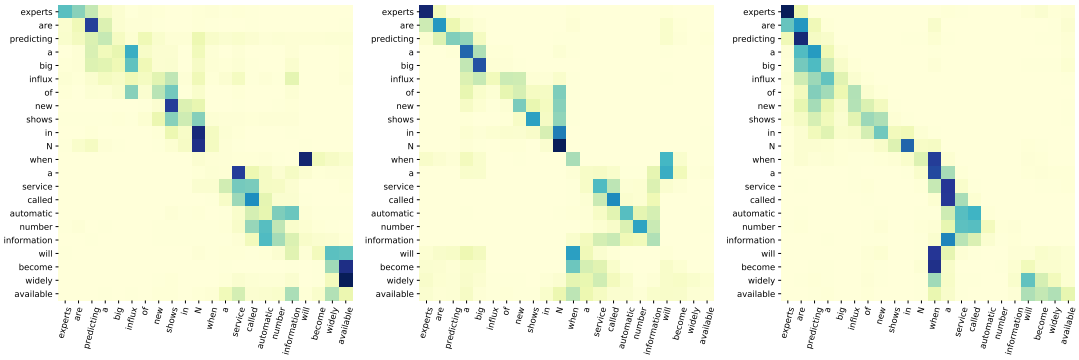
where $\texttt{prod}(i,k,j)$ returns a set of the product of words from the two subspans (i,k) and $(k+1,j)$. And $s_{c\text{x}}$ is defined as:

$$s_{c\text{x}}(i,j) = f(\mathbf{c}_{i,k}, \mathbf{c}_{k+1,j}), \tag{5.10}$$

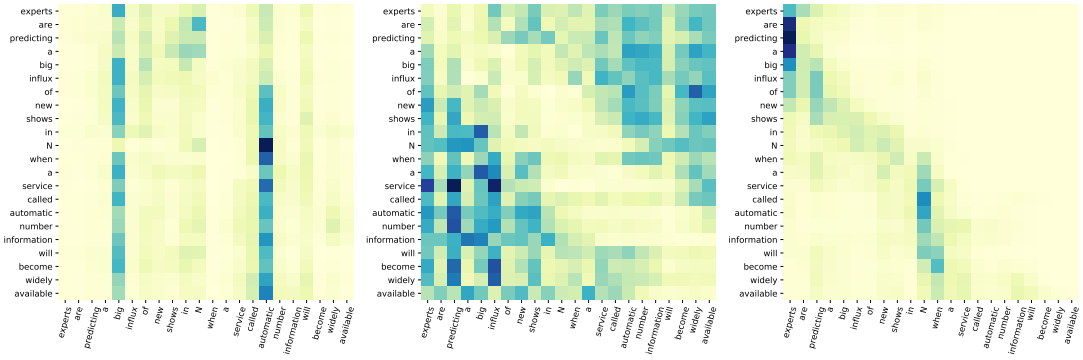
where $\mathbf{c}_{i,k} = \frac{1}{k-i+1}\sum_{i \le x \le k} g(w_x)$, $\mathbf{c}_{k+1,j} = \frac{1}{j-k}\sum_{k+1 \le y \le j} g(w_y)$.

We average all the combinations of $\{f \mid \text{JSD}, \text{HEL}\}$, $\{s_{comp} \mid s_p, s_c\}$ and $\{s_{cross} \mid s_{p\text{x}}, s_{c\text{x}}\}$ to rank all the attention heads and select the top-K heads. After the ranking step, we

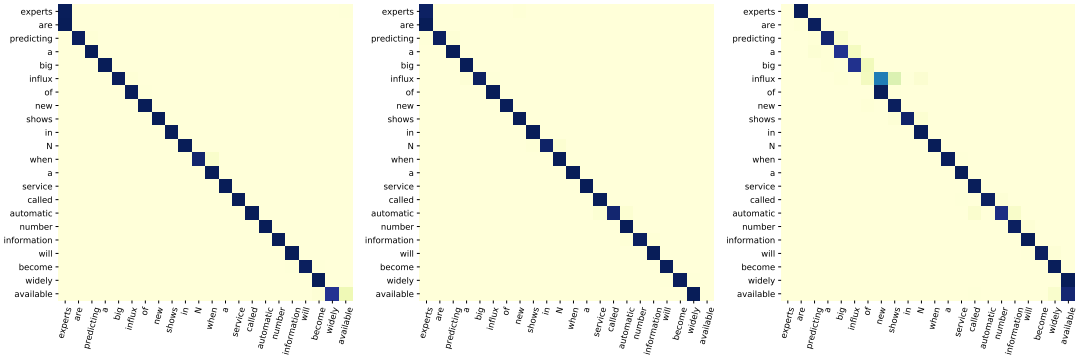
[4] Subscripts in the naming of functions in this paper: p – pair score, c – characteristic score, x – cross score.



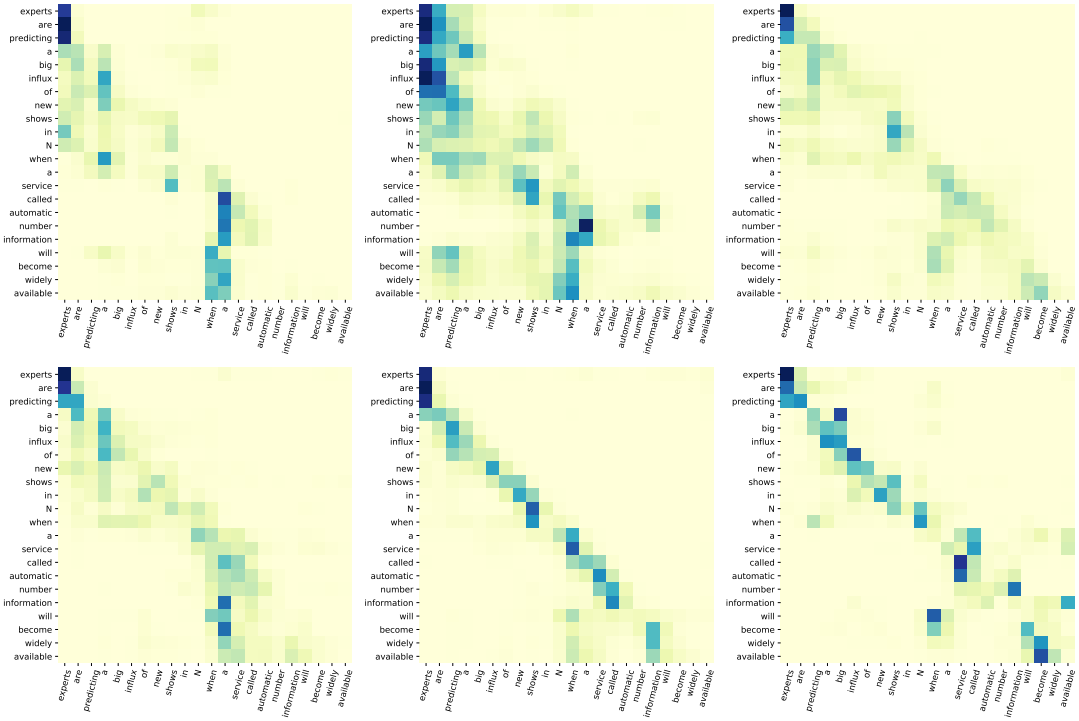
(a) Top-3 attention heads selected by ranking the single head parsing performance.



(b) Top-3 attention heads selected by ranking algorithm: the zero-shot chart-based parser.



(c) Top-3 attention heads selected by ranking algorithm: the zero-shot chart-based parser + cross score.



(d) Top-6 attention heads selected by ranking algorithm: the zero-shot chart-based parser + cross score + length weighting (ours).

Figure 5.2: Top-K attention heads selected by different ranking algorithms.

perform constituency parsing by ensembling the selected heads. We employ the ensemble method in Section 5.2.1 and average all the combinations of $\{f \mid \text{JSD}, \text{HEL}\}$ and $\{s_{comp} \mid s_p, s_c\}$ to get a single predicted parse tree for a given sentence.

5.2.3 How to select K

For ensemble parsing, Kim et al. (2020b) proposed three settings: the best head, layer-wise ensemble, and top-K ensemble. To prevent introducing a tunable hyperparameter, we propose to select a value for K dynamically based on the property of the ranking score in Equation (5.6).

Since we use a similarity-based distance, the lower the ranking score, the higher the ranking. Assuming that scores are computed for all attention heads, we can sort the scores in ascending order. Intuitively, given the order, we would like to choose the k for which ranking score increases the most, which means syntactic relatedness drops the most. Suppose $s_{parsing}(k)$ is the ranking score where k is the head index in the ascending order, then this is equivalent to finding the k with the greatest gradient on the curve of the score. We first estimate the gradient of $s_{parsing}(k)$ and then find the

k with the greatest gradient. Finally, K is computed as:

$$K = \underset{k}{\operatorname{argmax}} \sum_{\substack{k-\delta \leq j \leq k+\delta \\ j \neq k}} \frac{s_{parsing}(k+j) - s_{parsing}(k)}{j}, \tag{5.11}$$

where we smooth the gradient by considering δ steps. Here, we set $\delta = 3$.

In practice, we find that the greatest gradient always happens in the head or the tail of the curve. For the robustness, we select the K from the middle range of the score function curve, i.e., starting from 30 and ending with 75% of all heads. Although our ranking algorithm can filter out *noisy* heads, by observing the attention heatmaps, we find that noisy heads sometimes still rank high. We do not do any post-processing to further filter out the noisy heads, so we empirically search k starting at 30. We also provide a *lazy* option for K selection, which simply assume a fixed value of 30 for the top-K ensemble.

5.3 Grammar Induction

We are also interested in exploring to what extent the syntactic knowledge acquired by PLMs resembles human-annotated constituency grammars. For this exploration, we infer a constituency grammar, in the form of probabilistic production rules, from the trees induced from PLMs. This grammar can then be analyzed further, and compared to human-derived grammars. Thanks to the recent progress in neural parameterization, neural PCFGs have been successfully applied to unsupervised constituency parsing (Kim et al., 2019a). We harness this model [5] to learn probabilistic constituency grammars from PLMs by maximizing the joint likelihood of sentences and parse trees induced from PLMs. In the following, we first briefly review the neural PCFG and then introduce our training algorithm.

5.3.1 Neural PCFGs

A probabilistic context-free grammar (PCFG) consists of a 5-tuple grammar $\mathcal{G} = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R})$ and rule probabilities $\pi = \{\pi_r\}_{r \in \mathcal{R}}$, where S is the start symbol, $\mathcal{N}$ is a finite set of nonterminals, $\mathcal{P}$ is a finite set of preterminals, Σ is a finite set of terminal symbols, and $\mathcal{R}$ is a finite set of rules associated with probabilities π. The rules

[5] A more advanced version of the neural PCFG, the compound PCFG, has also been developed in Kim et al. (2019a). In this model variant, a compound probability distribution is built upon the parameters of a neural PCFG. In preliminary experiments, we found the compound PCFG learns similar grammars as the neural PCFG. So we only use the more light-weight neural PCFG in this chapter.

are of the form:

$$\begin{aligned} S &\rightarrow A, && A \in \mathcal{N} \\ A &\rightarrow BC, && A \in \mathcal{N}, \quad B,C \in \mathcal{N} \cup \mathcal{P} \\ T &\rightarrow w, && T \in \mathcal{P}, w \in \Sigma. \end{aligned}$$

Assuming $\mathcal{T}_{\mathcal{G}}$ is the set of all possible parse trees of $\mathcal{G}$, the probability of a parse tree $\mathbf{t} \in \mathcal{T}_{\mathcal{G}}$ is defined as $p(\mathbf{t}) = \prod_{r \in t_{\mathcal{R}}} \pi_r$, where $t_{\mathcal{R}}$ is the set of rules used in the derivation of $\mathbf{t}$. A PCFG also defines the probability of a given sentence $\mathbf{x}$ (string of terminals $\mathbf{x} \in \Sigma^*$) via $p(\mathbf{x}) = \sum_{\mathbf{t} \in \mathcal{T}_{\mathcal{G}}(\mathbf{x})} p(\mathbf{t})$, where $\mathcal{T}_{\mathcal{G}}(\mathbf{x}) = \{\mathbf{t} | \texttt{yield}(\mathbf{t}) = \mathbf{x}\}$, i.e., the set of trees $\mathbf{t}$ such that $\mathbf{t}$'s leaves are $\mathbf{x}$.

The traditional way to parameterize a PCFG is to assign a scalar to each rule π_r under the constraint that valid probability distributions must be formed. For unsupervised parsing, however, this parameterization has been shown to be unable to learn meaningful grammars from natural language data (Carroll and Charniak, 1992). Distributed representations, the core concept of the modern deep learning, have been introduced to address this issue (Kim et al., 2019a). Specifically, embeddings are associated with symbols and rules are modeled based on such distributed and shared representations.

In the neural PCFG, the log marginal likelihood:

$$\log p_\theta(\mathbf{x}) = \log \sum_{\mathbf{t} \in \mathcal{T}_{\mathcal{G}}(\mathbf{x})} p_\theta(\mathbf{t}) \tag{5.12}$$

can be computed by summing out the latent parse trees using the inside algorithm (Baker, 1979), which is differentiable and amenable to gradient based optimization. We refer readers to the original paper of Kim et al. (2019a) for details on the model architecture and training scheme.

5.3.2 Learning Grammars from Induced Trees

Given the trees induced from PLMs (described in Section 5.2.2), we use neural PCFGs to learn constituency grammars. In contrast to unsupervised parsing, where neural PCFGs are trained solely on raw natural language data, we train them on the sentences and the corresponding tree structures induced from PLMs. Note that this differs from a fully supervised parsing setting, where both tree structures and internal constituency tags (nonterminals and preterminals) are provided in the treebank. In our case, the trees induced from PLMs have no internal annotations.

For the neural PCFG training, the joint likelihood is given by:

$$\log p(\mathbf{x}, \hat{\mathbf{t}}) = \sum_{r \in \hat{t}_{\mathcal{R}}} \log \pi_r, \tag{5.13}$$

where $\hat{\mathbf{t}}$ is the induced tree and $\hat{t}_{\mathcal{R}}$ is the set of rules applied in the derivation of $\hat{\mathbf{t}}$. Although tree structures are given during training, marginalization is still involved: all internal tags will be marginalized to compute the joint likelihood. Therefore, the grammars learned by our method are anonymized: nonterminals and preterminals will be annotated as NT-*id* and T-*id*, respectively, where *id* is an arbitrary ID number.

5.4 Experiments

We conduct experiments to evaluate the unsupervised parsing performance of our ranking-based zero-shot parser on English and eight other languages (Basque, French, German, Hebrew, Hungarian, Korean, Polish, Swedish). For the grammars learned from the induced parse trees, we perform qualitative and quantitative analysis on how the learned grammars resemble the human-crafted grammar of the English PTB.

5.4.1 General Setup

We prepare the PTB (Marcus et al., 1993) for English and the SPMRL dataset (Seddah et al., 2013) for eight other languages. We adopt the standard split of each dataset to divide it into development and test sets. For preprocessing, we follow the setting in Kim et al. (2019a,b). Regarding PLMs, we follow the treatment in Kim et al. (2020b) for special functional tokens (e.g., [`CLS`], [`SEP`]). Specifically, special tokens are kept untouched at the forward pass of PLMs, then the corresponding dimensions in the attention matrix are trimmed.

We run our ranking algorithm on the development set to select the syntax-related heads and the ensemble parsing algorithm on the test set. We only use the raw sentences in the development set, without any syntactic annotations. We average all configurations both for ranking (f, s_{comp} and s_{cross}) and parsing (f and s_{comp}); hence we do not tune any hyperparameters for our algorithm. For K selection, we experiment with fixed top-K (i.e., top-30) and dynamically searching the best K described in Section 5.2.3, dubbed dynamic K. We report the unlabeled sentence-level F_1 score to evaluate the extent to which the induced trees resemble the corresponding gold standard trees.

For neural PCFG training, we modify some details but keep most of the model configurations of Kim et al. (2019a); we refer readers to the original paper for more information. We train the models on longer sentences for more epochs. Specifically,

Model	Top-down	Chart-based			Our ranking-based		
Configuration	Single /Layer†	Single /Layer†	Top -K	Top -K‡	Top -K	Dynamic K	Full heads
	w/ dev trees			w/o dev trees			
BERT-base-cased	32.6	37.5	**42.7**	29.3	34.8	**37.1**	35.8
BERT-large-cased	36.7	41.5	**44.6**	21.5	36.1	**38.7**	33.2
XLNet-base-cased	39.0	40.5	**46.4**	38.4	41.2	**42.7**	42.4
XLNet-large-cased	37.3	39.7	**46.4**	34.1	40.6	41.1	**41.2**
RoBERTa-base	38.0	41.0	**45.0**	35.9	41.7	**42.1**	39.6
RoBERTa-large	33.8	38.6	**42.8**	30.2	33.1	**37.5**	35.7
GPT2	35.4	34.5	**38.5**	21.9	26.1	**27.2**	26.1
GPT2-medium	37.8	38.5	**39.8**	19.4	**29.1**	**29.1**	27.2
AVG	36.3	39.0	**43.3**	28.8	35.3	**36.9**	35.1
AVG w/o GPT2 *	36.2	39.8	**44.7**	31.6	37.9	**39.8**	38.0

Table 5.1: Unlabeled sentence-level parsing F_1 scores on the English PTB test set. †: the best results of the top single head and layer-wise ensemble. ‡: directly applying the chart-based parser for ranking (without development set trees) and ensembling the top-K heads for parsing. *: average F_1 scores without GPT2 and GPT2-medium. Bold figures highlight the best scores for the two different groups: with and without development trees.

we train on sentences of length up to 30 in the first epoch, and increase this length limit by five until the length reaches 80. We train for 30 epochs and use a learning rate scheduler.

5.4.2 Results on the English PTB

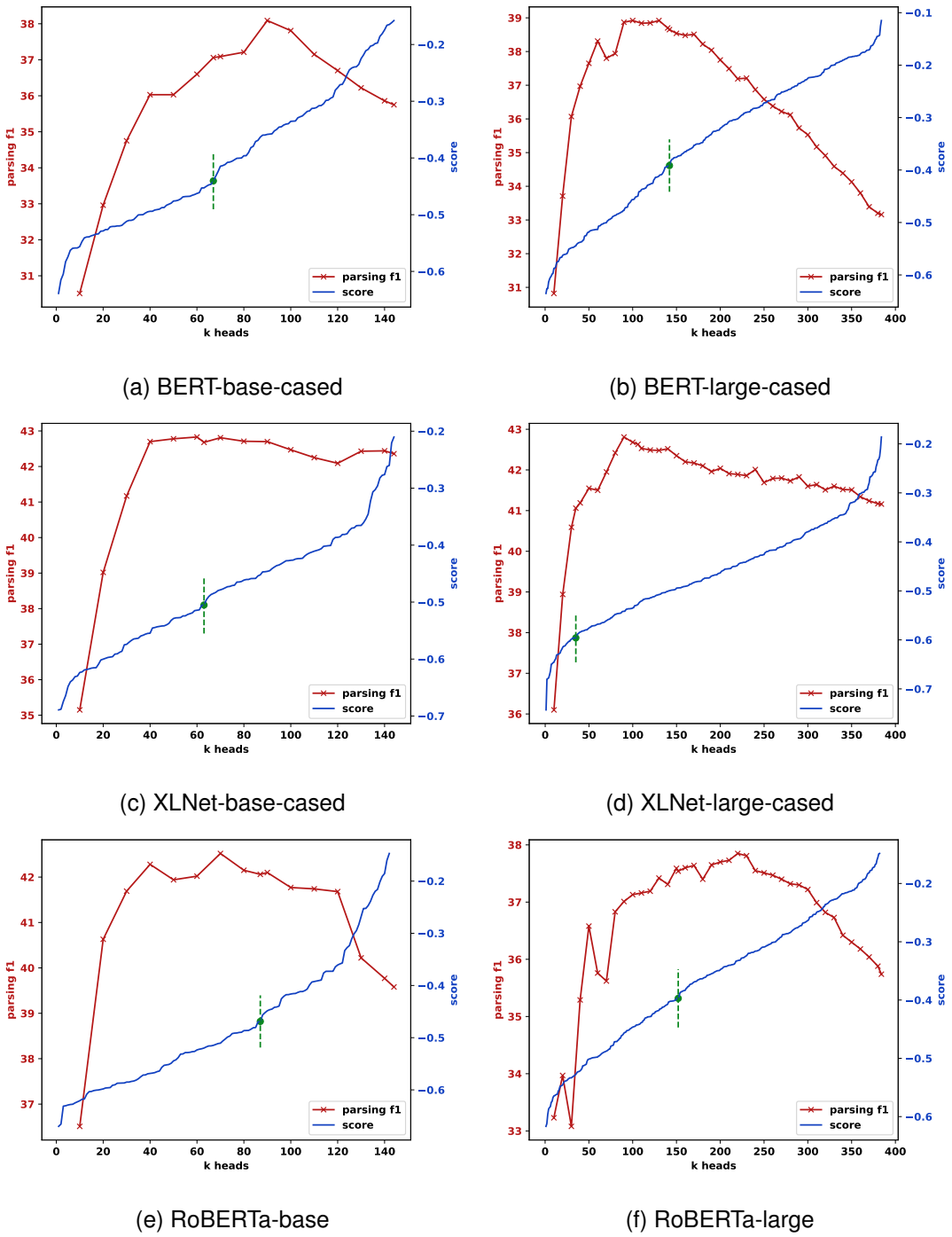
We first evaluate our ranking-based zero-shot parser on the English PTB dataset. We apply our methods to four different PLMs for English: BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019c), and GPT2 (Radford et al., 2019). We follow previous work (Kim et al., 2020a,b) in using two variants for each PLM, where the X-base variants consist of 12 layers, 12 attention heads, and 768 hidden dimensions, while the X-large ones have 24 layers, 16 heads, and 1024 dimensions.

With regard to GPT2, the GPT2 model corresponds to X-base while GPT2-medium to X-large.

Table 5.1 shows the unlabeled F_1 scores for our ranking-based zero-shot parser as well as for previous zero-shot parsers in two settings, with and without an annotated development set. We employ the chart-based parser in a setting without development trees, where Equations (5.2) and (5.3) are used for ranking and ensembling the top-K (i.e., top-30) heads. Compared to our method under the same configuration, its poor performance confirms the effectiveness of our ranking algorithm.

With respect to the K selection, our dynamic K method beats both fixed top-30 and full heads. Surprisingly, using all attention heads for ensemble parsing yields nearly the same performance as using top-30 heads. This suggests that although our ranking algorithm filters out some noisy heads, it is still not perfect. On the other hand, the ensemble parsing method is robust to noisy heads when full attention heads are used. Figure 5.3 shows how the ensemble parsing performance changes given different K selection. We can identify a roughly concave shape of the parsing performance curve, which indicates why our ranking algorithm works. Interestingly, the parsing performance does not drop too much when K reaches the maximum for XLNet. We conjecture that syntactic knowledge is more broadly distributed across heads in XLNet.

Our ranking-based parser performs badly on GPT2 and GPT2-medium, which is not unexpected. Unlike other PLMs, models in the GPT2 category are auto-regressive language models, whose attention matrix is strictly lower triangular. It makes it hard for our ranking algorithm to work properly. But for top-down and chart-based zero-shot parsers, tuning against an annotated development set can alleviate this problem. We focus on BERT, XLNet and RoBERTa and only evaluate these three models in the rest of our experiments. Except for GPT2 variants, our parser with dynamic K outperforms the top-down parser in all cases. On average (without GPT2 variants), even though our parser only requires raw sentence data, it still matches the chart-based parser with the top single head or layer-wise ensemble. To explore the limit of the chart-based parser, we also present the results by selecting the top-K (i.e., top-20) heads using the annotated development set (Kim et al., 2020b). Selecting heads against a development set ensures the quality of high ranking heads; top-20 heads are optimal in this setting (Kim et al., 2020b), unlike top-30 in our setting. Note that in this setting, the best configuration, i.e., the combination of g, f and s_{comp} as well as K are selected against the development set. This setting serves as an upper bound of the chart-based zero-shot parsing and largely outperforms our ranking-based method.



(a) BERT-base-cased

(b) BERT-large-cased

(c) XLNet-base-cased

(d) XLNet-large-cased

(e) RoBERTa-base

(f) RoBERTa-large

Figure 5.3: Relations between K for top-K and parsing performance on different PLMs. The blue curve shows the ranking score of heads where heads are sorted in an ascending order. The red curve shows the parsing performance that is evaluated on the PTB test set given every 10 heads. The green dashed line indicates the dynamic K.

Table 5.2 presents the parsing scores as well as recall scores on different constituents of trivial baselines and our parser. It indicates that trees induced from XLNet-

Model	F_1	SBAR	NP	VP	PP	ADJP	ADVP
Balanced	18.5	7	27	8	18	27	25
Left branching	8.7	5	11	0	5	2	8
Right branching	39.4	**68**	24	**71**	42	27	38
BERT-base-cased	37.1	36	49	30	42	40	69
BERT-large-cased	38.7	38	50	30	46	42	**72**
XLNet-base-cased	**42.7**	45	**58**	31	46	46	**72**
XLNet-large-cased	41.1	44	54	30	42	**48**	64
RoBERTa-base	42.1	38	**58**	31	**47**	42	71
RoBERTa-large	37.5	35	53	29	33	36	54

Table 5.2: Unlabeled parsing scores and recall scores on six constituency tags of trivial baseline parse trees as well as ones achieved by our parser using dynamic K on different PLMs.

base-cased, XLNet-large-cased and RoBERTa-base can outperform the right-branching baseline without resembling it. This confirms that PLMs can produce non-trivial parse trees. Large gains on NP, ADJP and ADVP compared to the right branching baseline show that PLMs can better identify such constituents.

5.4.3 Results for Languages other than English

Low-resource language parsing is one of the main motivations for the development of unsupervised parsing algorithms, which makes a multilingual setting ideal for evaluation. Multilingual PLMs are attractive in this setting because they are trained to process over one hundred languages in a language-agnostic manner. Kim et al. (2020b) investigated the zero-shot parsing capability of multilingual PLMs assuming that a small annotated development set is available. Here, by taking advantage of our ranking-based parsing algorithm, we only require raw sentences in the target language. Furthermore, we also evaluate a more radical crosslingual setting, where we rank attention heads only on sentences in English and directly apply the parser to eight other languages. We follow Kim et al. (2020b) and use four multilingual PLMs: a multilingual version of the BERT-base model (M-BERT, Devlin et al. 2019), the XLM model (Conneau and Lample, 2019), the XLM-R and XLM-R-large models (Conneau et al., 2020). Each multilingual PLM differs in architecture and pre-training data, and we refer readers to

	Language	English	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	AVG
	Trivial baselines										
	Balanced	18.5	24.4	12.9	15.2	18.1	14.0	20.4	26.1	13.3	18.1
	Left branching	8.7	14.8	5.4	14.1	7.7	10.6	16.5	28.7	7.6	12.7
	Right branching	39.4	22.4	1.3	3.0	0.0	0.0	21.1	0.7	1.7	10.0
Target language for head selection	Chart-based (Single/Layer) †										
	M-BERT	41.2	38.1	30.6	32.1	31.9	30.4	46.4	43.5	27.5	35.7
	XLM	43.0	35.3	35.6	41.6	39.9	34.5	35.7	51.7	33.7	39.0
	XLM-R	44.4	40.4	31.0	32.8	34.1	32.4	47.5	44.7	29.2	37.4
	XLM-R-large	40.8	36.5	26.4	30.2	32.1	26.8	45.6	47.9	25.8	34.7
	AVG	42.4	37.6	30.9	34.2	34.5	31.0	43.8	46.9	29.1	36.7
	Chart-based (Top-K) †										
	M-BERT	45.0	41.2	35.9	35.9	37.8	33.2	47.6	51.1	32.6	40.0
	XLM	**47.7**	41.3	**36.7**	**43.8**	**41.0**	36.3	35.7	**58.5**	**36.5**	**41.9**
	XLM-R	47.0	**42.2**	35.8	37.7	40.1	**36.6**	**51.0**	52.7	32.9	41.8
	XLM-R-large	45.1	40.2	29.7	37.1	36.2	31.0	46.9	47.9	27.8	38.0
	AVG	46.2	41.2	34.5	38.6	38.8	34.3	45.3	52.6	32.5	40.4
	Ranking-based (Top-K) ‡										
	M-BERT	41.5	38.9	33.9	30.2	36.3	30.9	39.0	18.4	26.3	31.7
	XLM	44.6	21.0	29.8	39.2	30.5	25.2	23.8	55.2	30.3	31.9
	XLM-R	44.8	36.0	34.1	31.8	36.4	32.5	40.3	29.6	26.7	33.4
	XLM-R-large	41.1	36.8	30.3	26.8	33.4	24.9	37.4	17.5	26.3	29.2
	AVG	43.0	33.2	32.0	32.0	34.2	28.4	35.1	30.2	27.4	31.6
	Ranking-based (Dynamic K) ‡										
	M-BERT	40.7	39.1	28.4	25.5	26.9	31.2	41.3	22.2	21.3	29.5
	XLM	44.9	20.8	29.9	40.3	34.4	27.7	23.6	55.1	31.2	32.9
	XLM-R	45.5	37.3	30.7	31.5	31.8	34.1	40.8	36.0	27.4	33.7
	XLM-R-large	41.0	36.5	29.0	30.1	32.6	25.3	43.9	30.0	25.5	31.6
	AVG	43.0	33.4	29.5	31.9	31.4	29.6	37.4	35.8	26.4	31.9
English for head selection	Crosslingual ranking-based (Top-K) ‡										
	M-BERT	-	37.9	33.4	31.2	31.5	29.4	45.3	33.4	27.2	34.5
	XLM	-	25.9	34.4	39.2	39.5	31.9	27.5	**50.4**	**34.2**	36.4
	XLM-R	-	37.9	33.9	35.1	36.8	33.3	44.7	39.7	30.3	37.4
	XLM-R-large	-	35.7	28.5	28.5	34.7	25.5	44.5	36.9	27.1	33.6
	AVG	-	34.3	32.6	33.5	35.6	30.0	40.5	40.1	29.7	35.5
	Crosslingual ranking-based (Dynamic K) ‡										
	M-BERT	-	**38.2**	31.0	31.0	29.0	27.1	43.3	30.7	25.8	33.0
	XLM	-	26.6	**35.8**	**39.7**	**39.6**	32.9	28.0	50.1	34.1	36.9
	XLM-R	-	**38.2**	34.0	35.5	36.7	**33.5**	**45.2**	39.4	29.9	**37.6**
	XLM-R-large	-	37.9	28.0	28.0	31.3	24.6	44.4	32.2	24.9	32.5
	AVG	-	34.7	32.4	33.5	35.0	29.8	40.4	39.2	29.2	35.3

Table 5.3: Parsing results on nine languages with multilingual PLMs. Except for the trivial baselines, all experimental results are divided into two groups: using target language for head selection and using English for head selection (crosslingual). †: results of the best configurations of f, g, s_{comp} and K are decided on an annotated development set. ‡: results where only raw sentences are required. For top-K, 20 is used for chart-based and 30 is used for our ranking-based. Bold figures highlight the best scores for the two different groups: using target language and English for head selection.

the original papers for more details.

We present a comprehensive analysis of the chart-based parser and our ranking-based parser on the multilingual setting in Table 5.3. For our ranking-based method, we conduct experiments using target language or English for head selection with both Top-K (i.e., top-30) ensemble and dynamic K ensemble. Our method outperforms the trivial baselines in all cases by a large margin. Through the comparison between different K selection strategies, we find that our ranking-based parser with Top-K ensemble matches that using dynamic K ensemble. In contrast to the superiority of dynamic K on English PLMs in Table 5.1, multilingual PLMs produce similar parsing performance with a *lazy* top-30 ensemble. We conjecture that there could be no clear concave pattern (unlike in Figure 5.3) in the relation of K and parsing performance in this crosslingual setting.

In terms of head selection via different languages, interestingly, we observe a considerable parsing performance drop on both top-K and dynamic K ensemble. We suspect that our chart-based ranking algorithm (e.g., the inherent context free grammar assumption) does not work equally well on all languages, at least for the annotation scheme provided by the SPMRL dataset. In this scenario, using English for head selection has a better chance to capture syntax-related attention heads. Again, as we discussed before, using annotated trees in the target language can always ensure the quality of the selected top-K heads.

Compared with the chart-based parser with the top head or layer-wise ensemble, our crosslingual parser can match the performance on five out of nine languages. Our method still lags behind the chart-base zero-shot parser with a top-K ensemble. Among four model variants, XLM-R and XLM-R-large have identical training settings and pre-training data, and so form a controlled experiment. By directly comparing XLM-R and XLM-R-large, we conjecture that, as the capacity of the PLM scales, the model has more of a chance to learn separate hidden spaces for different languages. This is consistent with a recent study on multilingual BERT (Dufter and Schütze, 2020) showing that underparameterization is one of the main factors that contribute to multilinguality. In their study, the authors demonstrate that if the Transformer is severely overparameterized, the model has enough capacity to model each language separately without creating a multilingual space. However, if the number of parameters is small, the model is likely to identify common structures among languages and model them together.

5.4.4 Grammar Analysis

By not relying on an annotated development set, we have an unbiased way of investigating the tree structures as well as the grammars that are inherent in PLMs. Specifically, we first parse the raw sentences using our ranking-based parser described in Section 5.2.2 and then train a neural PCFG given the induced trees using the method in Section 5.3.2. We conduct our experiments on the English PTB and evaluate how the learned grammar resembles PTB syntax in a quantitative way on preterminals (POS tags) and production rules. We visualize the alignment of preterminals and nonterminals of the learned grammar and the gold labels as a qualitative study. We also showcase parse trees of the learned grammar to get a glimpse of some distinctive characteristics of the learned grammar. For brevity, we refer to a neural PCFG learned from trees induced of a PLM as PCFG_{PLM} and to a neural PCFG learned from the gold parse trees as $\text{PCFG}_{\text{Gold}}$.

In Table 5.4, we report preterminal (unsupervised POS tagging) accuracies and production rule accuracies of PCFG_{PLM} and $\text{PCFG}_{\text{Gold}}$ on the corpus level. For preterminal evaluation, we map the anonymized preterminals to gold POS tags using many-to-one (M-1) mapping (Johnson, 2007), where each anonymized preterminal is matched onto the gold POS tag with which it shares the most tokens. For production rule evaluation, we map both nonterminals and preterminals to gold tags using M-1 mapping to get the binary production rules. Regarding the gold annotations, we drop all unary rules. For n-ary rules ($n > 2$), we convert them to binary rules by right branching and propagating the parent tag. For example, a n-ary rule $A \rightarrow B\ C\ D$ yields $A \rightarrow B\ A$ and $A \rightarrow C\ D$.

Regarding unsupervised POS tagging, we find that all PCFG_{PLM} grammars except for $\text{PCFG}_{\text{RoBERTa-large}}$ outperform a discrete HMM baseline (62.7, He et al. 2018) but are far from the state of the art for neural grammar induction (80.8, He et al. 2018). All PCFG_{PLM} produce similar accuracies on preterminals as $\text{PCFG}_{\text{Gold}}$. However, for the production rules, PCFG_{PLM} lags behind $\text{PCFG}_{\text{Gold}}$ by a large margin. This makes sense as presumably the tree structures heavily affect nonterminal learning. We also present the parsing F_1 scores of corresponding trees against the gold trees in Table 5.4 for comparison. We observe that for all PCFG_{PLM}, both preterminal accuracies and production rule accuracies correlate with the parsing F_1 scores of the corresponding trees.

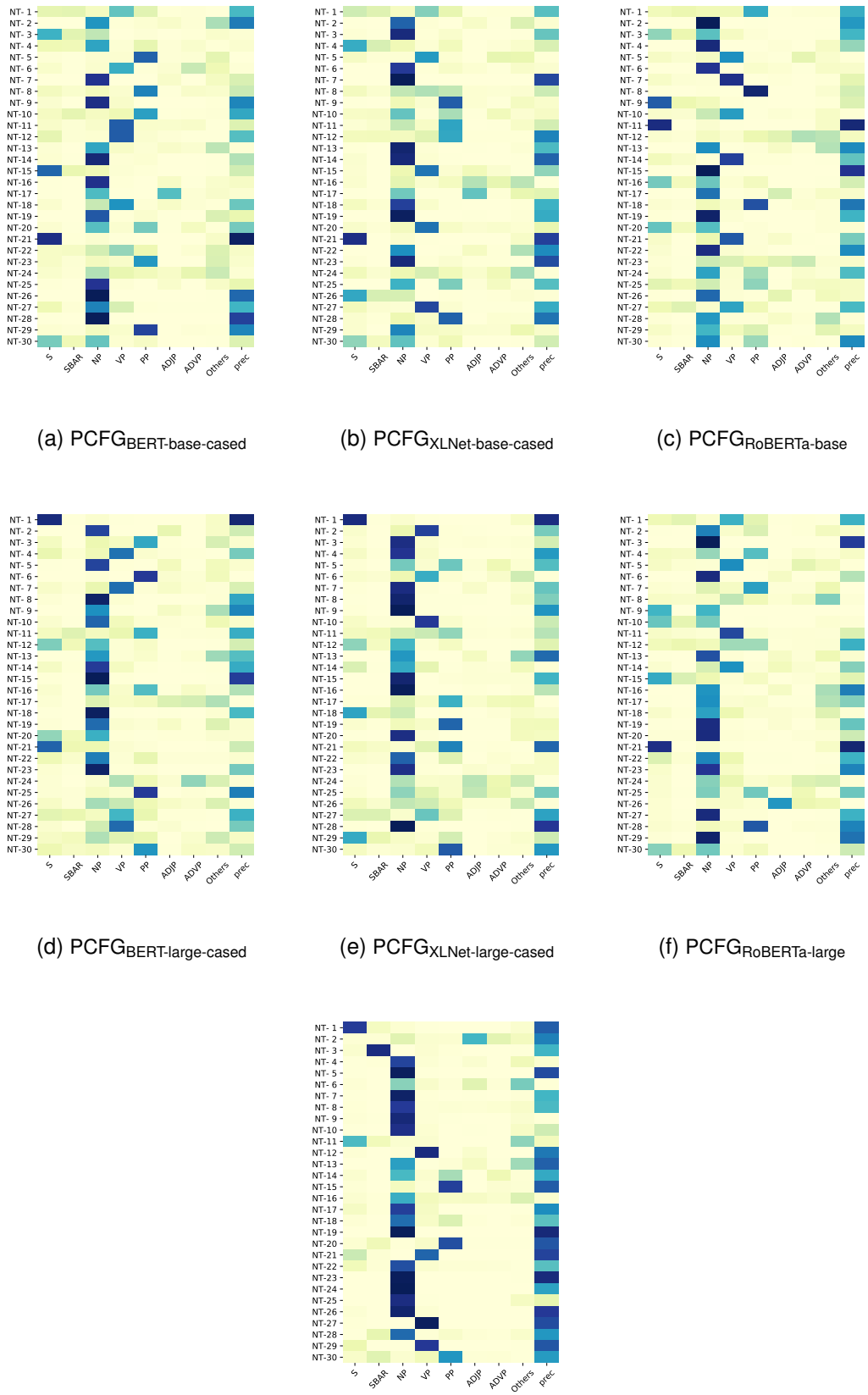
Since the recall scores in Table 5.2 have shown ability of PLMs to identify dif-

Trees	Preterminal Acc†	Rule Acc‡	Parsing F_1
Gold*	66.1	**46.2**	-
BERT-base-cased	64.4	24.8	37.1
BERT-large-cased	64.0	22.3	38.7
XLNet-base-cased	**67.7**	26.1	42.7
XLNet-large-cased	65.8	27.3	41.1
RoBERTa-base	65.7	27.2	42.1
RoBERTa-large	62.4	25.1	37.5

Table 5.4: Preterminal (POS tag) and production rule accuracies of PCFG_{PLM} and $\text{PCFG}_{\text{Gold}}$ on the entire PTB. †: POS tagging accuracy using the many-to-one mapping (Johnson, 2007). ‡: production rule accuracy where anonymized nonterminals and preterminals are mapped to the gold tags using the many-to-one mapping. *: $\text{PCFG}_{\text{Gold}}$.

ferent nonterminals, here we visualize the alignment between PCFG internal tags and corresponding gold labels in Figures 5.4 and 5.5. For the nonterminal alignment, some of the learned nonterminals clearly align to gold standard labels, in particular for frequent ones like NP and VP. Compared to $\text{PCFG}_{\text{Gold}}$, PCFG_{PLM} learns a more uncertain grammar, resulting in overall lower precision. But for the preterminal (POS tag) alignment, no clear difference can be identified between $\text{PCFG}_{\text{Gold}}$ and PCFG_{PLM}. This is consistent with the finding in Table 5.4 that all PCFG_{PLM} produce similar accuracies on preterminals as $\text{PCFG}_{\text{Gold}}$.

In Figure 5.6, we show parse trees obtained by $\text{PCFG}_{\text{Gold}}$, PCFG_{PLM} and the gold standard reference on a sample sentence. In this sample, $\text{PCFG}_{\text{Gold}}$ predicts the constituency tree structure accurately. On the development set, $\text{PCFG}_{\text{Gold}}$ reaches around 72 unlabeled F_1 score, as it is supervised by the PTB trees. Although this is a low F_1-score, it is not untypical for PCFG-based models, which are limited by their insufficiently flexible rules and their lack of lexicalization. Also note that the oracle binary trees yield 84.3 F_1 (Cao et al., 2020), which are produced by by taking the gold trees and binarizing them arbitrarily. PCFG_{PLM} perform worse than $\text{PCFG}_{\text{Gold}}$ when compared against the gold tree. They are able to identify short NPs, but don't work well for larger constituents. We also observe some frequent incorrect patterns which are also present in this example, e.g., grouping VBD with the preceding NP, or IN with the preceding VBD.

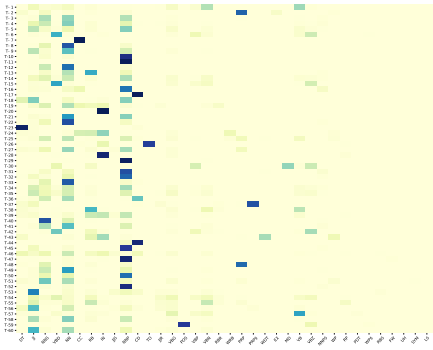


(a) $\text{PCFG}_{\text{BERT-base-cased}}$ (b) $\text{PCFG}_{\text{XLNet-base-cased}}$ (c) $\text{PCFG}_{\text{RoBERTa-base}}$

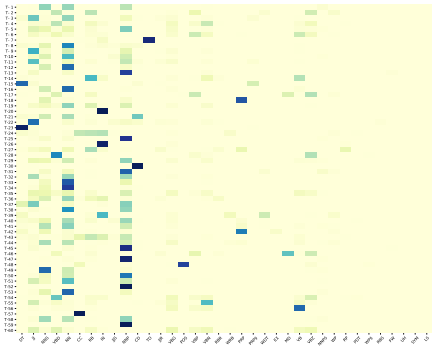
(d) $\text{PCFG}_{\text{BERT-large-cased}}$ (e) $\text{PCFG}_{\text{XLNet-large-cased}}$ (f) $\text{PCFG}_{\text{RoBERTa-large}}$

(g) $\text{PCFG}_{\text{Gold}}$

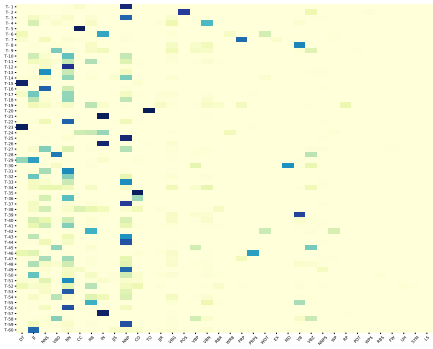
Figure 5.4: Alignment of induced nonterminals of PCFG_{PLM} and $\text{PCFG}_{\text{Gold}}$ on the entire PTB. The last column `prec` shows the precision with which a nonterminal predicts a particular gold constituent.



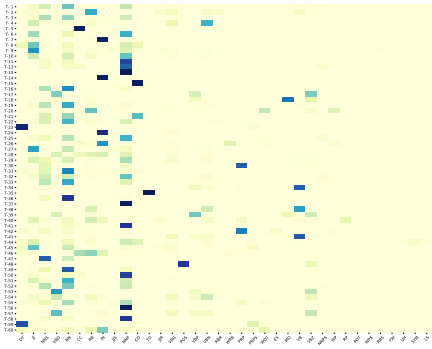
(a) $\text{PCFG}_{\text{BERT-base-cased}}$



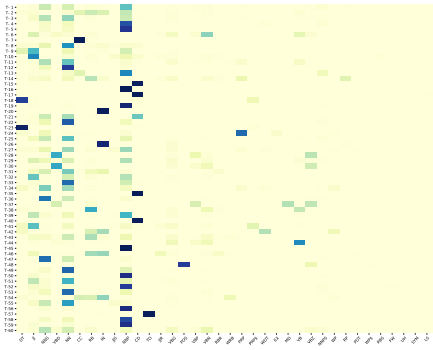
(b) $\text{PCFG}_{\text{BERT-large-cased}}$



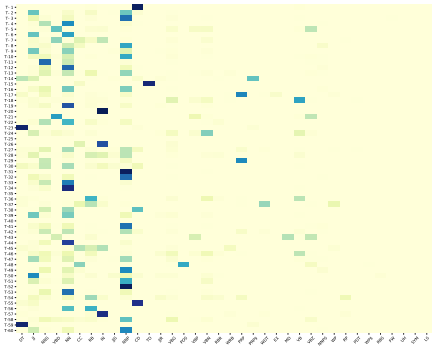
(c) $\text{PCFG}_{\text{XLNet-base-cased}}$



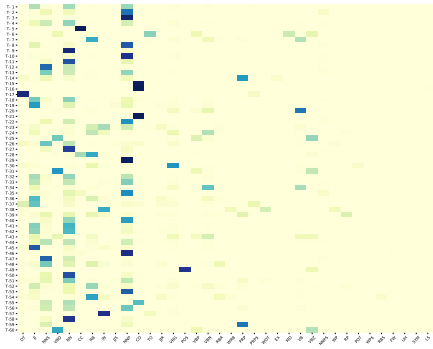
(d) $\text{PCFG}_{\text{XLNet-large-cased}}$



(e) $\text{PCFG}_{\text{RoBERTa-base}}$

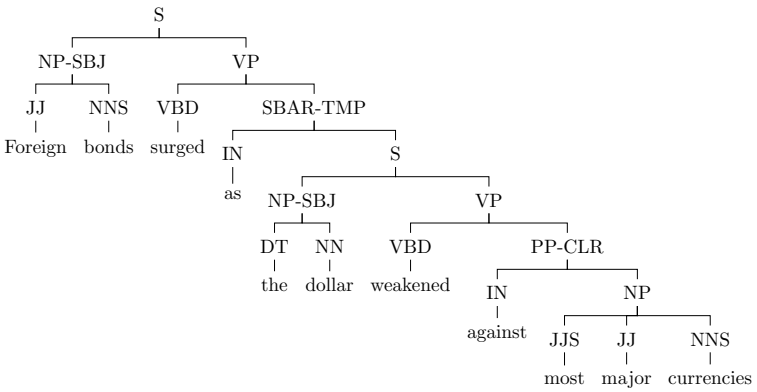


(f) $\text{PCFG}_{\text{RoBERTa-large}}$

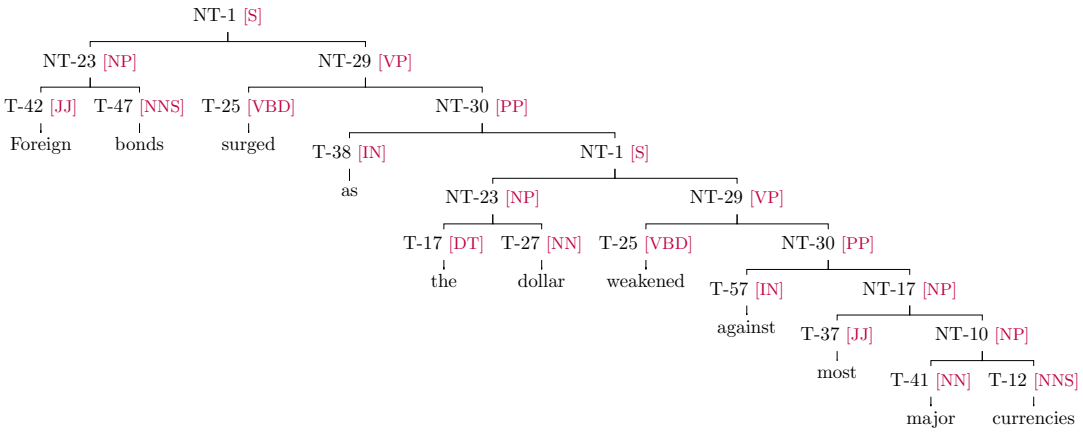


(g) $\text{PCFG}_{\text{Gold}}$

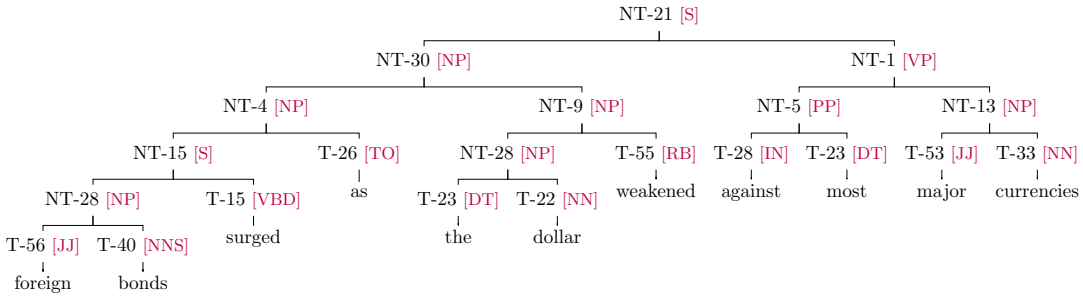
Figure 5.5: Alignment of induced preterminals (POS tags) of PCFG_{PLM} and $\text{PCFG}_{\text{Gold}}$ on the entire PTB.



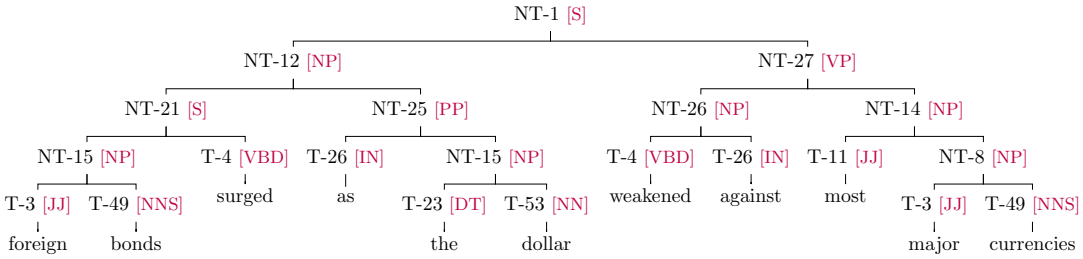
(a) Gold



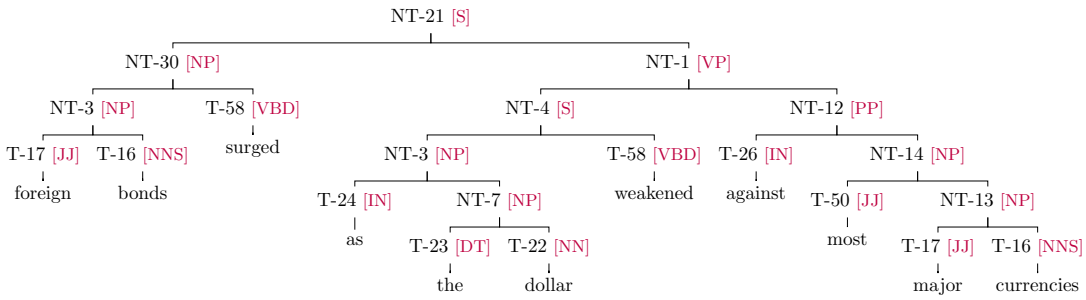
(b) $\text{PCFG}_{\text{Gold}}$



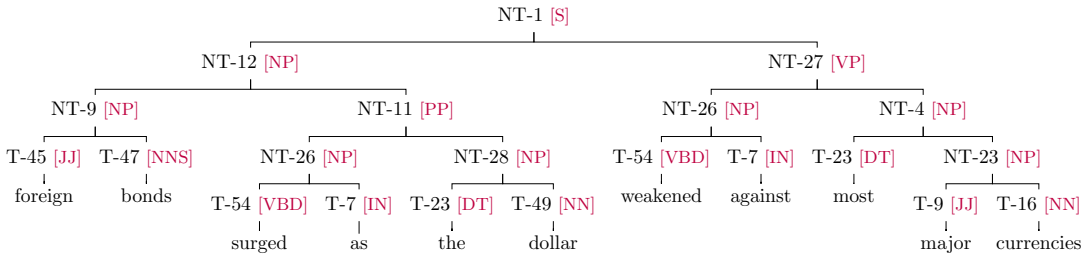
(c) $\text{PCFG}_{\text{BERT-base-cased}}$



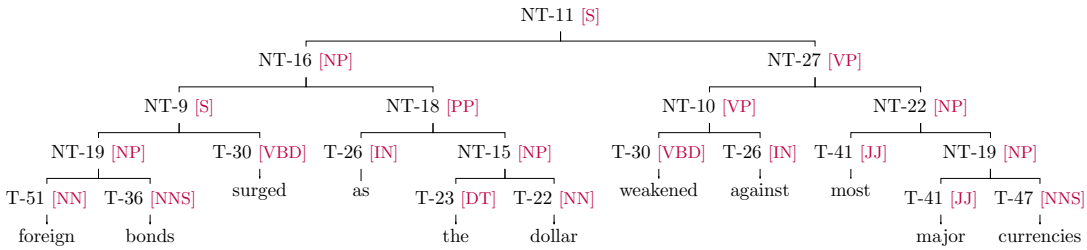
(d) $\text{PCFG}_{\text{BERT-large-cased}}$



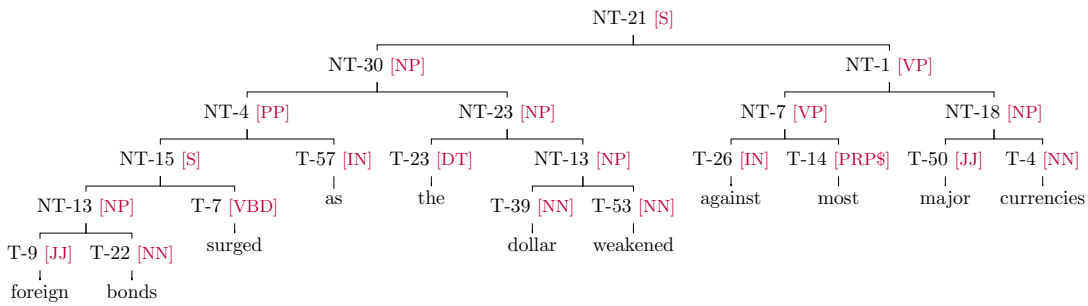
(e) $\text{PCFG}_{\text{XLNet-base-cased}}$



(f) $\text{PCFG}_{\text{XLNet-large-cased}}$



(g) $\text{PCFG}_{\text{RoBERTa-base}}$



(h) $\text{PCFG}_{\text{RoBERTa-large}}$

Figure 5.6: Parse tree samples of gold standard, $\text{PCFG}_{\text{Gold}}$, and PCFG_{PLM}. The mapped tag (marked in red) for each anonymized nonterminal and preterminal is obtained via many-to-one mapping.

5.5 Summary

In this chapter, we set out to analyze the syntactic knowledge learned by transformer-based pre-trained language models. In contrast to previous work relying on test suites and probes, we proposed to use a zero-shot unsupervised parsing approach. This approach parses sentences by ranking and ensembling the attention heads of the PLM. Our approach is able to completely do away with a development set annotated with syntactic structures, which makes it ideal in a strictly unsupervised setting, e.g., for low resource languages. We evaluated our method against previous methods on nine languages. When only raw sentences are available, our method can outperform previous methods by a large margin. When annotated development sets are available for previous methods, our method can match them or produce competitive results if they use the top single head or layer-wise ensemble of attention heads, but lags behind them if they ensemble the top-K heads. Furthermore, we presented an analysis of the grammars learned by our approach: we used the induced trees to train a neural PCFG and evaluated the pre-terminal and non-terminal symbols of that grammar. In future work, we will develop further methods for analyzing the resulting grammar rules. Another avenue for follow-up research is to use our method to determine how the syntactic structures inherent in PLMs change when these models are fine-tuned on a specific task.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

In this thesis, we investigated the problem of unsupervised parsing for two formalisms, dependency and constituency. It has been considered a difficult problem due to ambiguity and huge search space. With the rapid development of deep neural networks, neural models have been brought to bear on unsupervised parsing to ease optimization and smooth rule probabilities. We explored how to incorporate external knowledge into neural unsupervised parsing models and demonstrated that neural models can be improved by leveraging three knowledge sources: (1) symbolic linguistic rules; (2) alternative learning objectives; (3) large-scale pre-trained language models. We also induced probabilistic context-free grammars for constituency structures.

Firstly, we studied the problem of unsupervised dependency parsing. Previous state-of-the-art models all rely on global inference with $O(n^3)$ run time. For transition-based models, although they enable faster inference with $O(n)$ run time and perform well in supervised parsing, their performance on unsupervised parsing still lags behind. In Chapter 3, we used an autoencoder to integrate discriminative and generative transition-based parsers, dependency variants of recurrent neural network grammars (RNNGs; Dyer et al. 2016), yielding a reconstruction process with parse trees as latent variables. For regularization, we augmented the model with posterior regularization (Ganchev et al., 2010), which allowed us to seamlessly integrate linguistic knowledge in the shape of symbolic linguistic rules and still maintained the efficiency of transition-based systems. Furthermore, we proposed a novel variance reduction method to stabilize neural variational inference with discrete latent variables. Experimental results on English and eight other languages showed that our model outperforms previous

unsupervised transition-based dependency parsers and substantially increases parsing speed over global inference-based models.

Apart from symbolic linguistic rules, knowledge from alternative learning objectives can also be beneficial for grammar induction. In Chapter 4, we focused on latent tree learning for unsupervised constituency parsing. We proposed an imitation learning approach to combine two typical latent tree models, a continuous soft model (i.e., PRPN; Shen et al. 2018b) and a discrete hard model (i.e., Tree-LSTM). We exploited the advantages of the PRPN (being differentiable) by transferring its knowledge learned from a language modeling objective to a discrete parser, which explicitly models tree-building operations. Then the discrete parser refined its policy by solely trained on a classification task. Experiments were performed on the Natural Language Inference dataset (Bowman et al., 2015). Empirical results showed that our approach outperforms previous models on this task in terms of parsing performance and self-agreement, confirming the effectiveness of the proposed approach. Our results also indicated that a downstream, non-syntactic task can be useful for latent tree induction.

Pre-trained language models (PLMs) have achieved remarkable success in many NLP tasks. Recent studies (Goldberg, 2019; Liu et al., 2019a) showed that PLMs can learn considerable syntactical knowledge. This suggests that PLMs can be employed as a resource of knowledge for grammar induction. In Chapter 5, we proposed a novel approach to build a PLM-based unsupervised constituency parser without requiring an annotated development set. More specifically, we ranked Transformer heads based on their inherent properties and then ensembled the top-K heads to produce constituency trees. On English and eight other languages, our approach yields competitive parsing performance. For grammar induction, we learned neural probabilistic context-free grammars (PCFGs) from the trees induced from PLMs using our approach. We confirmed that PLMs have captured considerable syntactic knowledge and provided a novel approach to extract full grammars (not only parse trees) from PLMs.

6.2 Future Work

We would like to once again discuss *why* grammar induction, or general unsupervised structure learning, is worth investigation in the future research. Concerning downstream NLP tasks like language modeling or text classification, from an absolute performance standpoint, neural models that contain no structure-aware components or employ no structure knowledge are incredibly effective. Substantial studies, including

our work in Chapter 5, have shown that linguistic structures are captured *implicitly* within hidden layers of such neural models (e.g., PLMs) through end-to-end training. Nonetheless, unsupervised structure learning can be useful in modern NLP. *Explicitly* modeling structures as latent variables provides a principled probabilistic way to inject effective inductive bias and linguistically plausible constraints into neural NLP models. It also delivers interpretability, transparency and controllability to end-to-end neural models that typically lack these properties. On the other hand, from a utilitarian point of view, high-quality syntactical analysis, *can* still bring performance gains to large-scale PLMs on downstream tasks such as structured prediction (Kuncoro et al., 2020) and information extraction (Sachan et al., 2020). [1] Recent work (Cao et al., 2020) has obtained promising results on unsupervised parsing by utilizing an English PLM (i.e., RoBERTa; Liu et al. 2019c), approaching the performance of a supervised parser. It sets up a strong starting point and leaves open questions for follow-up research. An unsupervised parsing system can be applied to real-world applications if it could finally reach the performance of a supervised one.

With respect to model development, there are still remaining challenges. Firstly, regarding deep latent variable models, backpropagating through discrete structures is a challenging problem. It is also a key factor whether meaningful latent structures can be learned in end-to-end training. To this end, more advanced gradient estimators could be studied and adapted to unsupervised structure learning. Apart from two popular approximations used in this thesis, REINFORCE and Gumbel-Softmax, alternative approaches have been actively studied such as differentiable sparse mappings (Niculae et al., 2018; Correia et al., 2020) and reducing sampling noise (Grathwohl et al., 2018; Liu et al., 2019b; Kool et al., 2020). Secondly, combining heterogeneous parsers could be beneficial. Different parsers may have different assumptions and characteristics, a combination can achieve ensemble effects. Similar to our work in Chapter 4, fine-tuning on URNNG (Kim et al., 2019b) has been shown to be effective in unsupervised constituency parsing (Kim et al., 2019a; Cao et al., 2020). Thirdly, external knowledge from other modalities is worth investigating for grammar induction. In this thesis, we have shown that harnessing external knowledge can effectively improve grammar induction systems. Visually grounded grammar induction work (Shi et al., 2019; Zhao and Titov, 2020) suggests that visual groundings can produce more accurate and stable parsing models than text-only approaches. Another direction is language games, such

[1] In Sachan et al. (2020), the authors stress that the performance gains are highly contingent on the availability of human-annotated dependency parses.

as language based multi-agent simulations. While current research work (Ren et al., 2020; Chaabouni et al., 2020) focuses on identifying *compositionality* in the context of concepts transmission, grammar learning can be performed in more sophisticated simulations. Last but not the least, multilingual unsupervised parsing is a fascinating avenue for future research. Low resource languages are one of the main motivations to develop unsupervised parsing algorithms. In Chapter 5, we have shown that multilingual PLMs, which are jointly trained on over 100 languages, have the potential for cross-lingual parsing. Multilingual PLMs offer a new angle and could inspire follow-up work on this problem.

Appendix A

Appendix

A.1 Appendix 1

In Chapter 3, we modify the score function in Equation (3.19) to be a simpler version as Equation (3.9). We show how this modification affects the loss function.

$$\begin{aligned}
&\mathbb{E}_{q(\mathbf{a})}[\log \gamma(\mathbf{x},\mathbf{a})] \\
&\quad = \sum_{\mathbf{a}} q(\mathbf{a}) \log \gamma(\mathbf{x},\mathbf{a}) \\
&\quad = \sum_{\mathbf{a}} q_\omega(\mathbf{a}|\mathbf{x}) \gamma(\mathbf{x},\mathbf{a}) \log \gamma(\mathbf{x},\mathbf{a}) \\
&\quad \leqslant \sum_{\mathbf{a}} q_\omega(\mathbf{a}|\mathbf{x}) \gamma(\mathbf{x},\mathbf{a}) \gamma(\mathbf{x},\mathbf{a}) \\
&\quad = \mathbb{E}_{q_\omega(\mathbf{a}|\mathbf{x})}[\gamma^2(\mathbf{x},\mathbf{a})] \\
&\quad = \mathrm{Var}_{q_\omega(\mathbf{a}|\mathbf{x})}[\gamma(\mathbf{x},\mathbf{a})] + \mathbb{E}_{q_\omega(\mathbf{a}|\mathbf{x})}[\gamma(\mathbf{x},\mathbf{a})]^2
\end{aligned} \tag{A.1}$$

Since

$$\mathbb{E}_{q_\omega(\mathbf{a}|\mathbf{x})}[\gamma(\mathbf{x},\mathbf{a})] = \sum_{\mathbf{a}} q_\omega(\mathbf{a}|\mathbf{x}) \gamma(\mathbf{x},\mathbf{a}) = 1 \tag{A.2}$$

we have

$$\begin{aligned}
&\mathbb{E}_{q(\mathbf{a})}[\log \frac{p_\theta(\mathbf{x},\mathbf{a})}{q_\omega(\mathbf{a}|\mathbf{x})}] \\
&\quad = \mathbb{E}_{q(\mathbf{a})}[\log \frac{p_\theta(\mathbf{x},\mathbf{a})}{q_\omega(\mathbf{a}|\mathbf{x})\gamma(\mathbf{a},\mathbf{x})} + \log \gamma(\mathbf{a},\mathbf{x})] \\
&\quad = \mathbb{E}_{q(\mathbf{a})}[\log \frac{p_\theta(\mathbf{x},\mathbf{a})}{q(\mathbf{a})}] + \mathbb{E}_{q(\mathbf{a})}[\log \gamma(\mathbf{a},\mathbf{x})] \\
&\quad \leqslant \mathcal{L}_x + \mathrm{Var}_{q_\omega(\mathbf{a}|\mathbf{x})}[\gamma(\mathbf{x},\mathbf{a})] + 1.
\end{aligned} \tag{A.3}$$

Thus, in theory, $\mathrm{Var}_{q_\omega(\mathbf{a}|\mathbf{x})}[\gamma(\mathbf{x},\mathbf{a})]$ can be viewed as a regularization for posterior regularization. And practically, this help the learning process to be more steady.

Bibliography

Aguilar, G., Ling, Y., Zhang, Y., Yao, B., Fan, X., and Guo, C. (2020). Knowledge distillation from internal representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7350–7357.

Aharoni, R. and Goldberg, Y. (2017). Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver, Canada.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

Arora, S., Cohen, N., and Hazan, E. (2018). On the optimization of deep networks: Implicit acceleration by overparameterization. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 244–253.

Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bain, M. and Sammut, C. (1995). A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129.

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

Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Gao, J., Piao, S., Zhou, M., and Hon, H.-W. (2020). UniLMv2: Pseudo-masked language models

for unified language model pre-training. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 642–652, Virtual.

Belinkov, Y. and Glass, J. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena scientific Belmont.

Billard, A. G., Calinon, S., and Dillmann, R. (2016). Learning from humans. In *Springer handbook of robotics*, pages 1995–2014. Springer.

Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*.

Blunsom, P. and Cohn, T. (2010). Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213, Cambridge, MA.

Bod, R. (2006). An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872, Sydney, Australia.

Bohnet, B., McDonald, R., Simões, G., Andor, D., Pitler, E., and Maynez, J. (2018). Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652, Melbourne, Australia. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal.

Bowman, S. R., Gauthier, J., Rastogi, A., Gupta, R., Manning, C. D., and Potts, C. (2016). A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477, Berlin, Germany.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

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

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Buciluă, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

Buys, J. and Blunsom, P. (2015). Generative incremental dependency parsing with neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 863–869, Beijing, China.

Cai, J., Jiang, Y., and Tu, K. (2017). CRF autoencoder for unsupervised dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1638–1643, Copenhagen, Denmark.

Cao, S., Kitaev, N., and Klein, D. (2020). Unsupervised parsing via constituency tests. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 4798–4808, Online.

Carroll, G. and Charniak, E. (1992). Two experiments on learning probabilistic dependency grammars from corpora. In *Workshop Notes, Statistically Based NLP Techniqies*, pages 1–13. Proceedings of the AAAI Conference on Artificial Intelligence.

Chaabouni, R., Kharitonov, E., Bouchacourt, D., Dupoux, E., and Baroni, M. (2020). Compositionality and generalization in emergent languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4427–4442, Online.

Charniak, E. (1996). *Statistical language learning*. MIT press.

Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750, Doha, Qatar.

Chen, S. F. (1995). Bayesian grammar induction for language modeling. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 228–235, Cambridge, Massachusetts, USA.

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas.

Cheng, J., Lopez, A., and Lapata, M. (2017). A generative parser with a discriminative recognition algorithm. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 118–124, Vancouver, Canada.

Chi, Z., Dong, L., Wei, F., Wang, W., Mao, X.-L., and Huang, H. (2020). Cross-lingual natural language generation via pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7570–7577.

Choi, J., Yoo, K. M., and Lee, S.-g. (2018). Learning to compose task-specific tree structures. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5094–5101.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and control*, 2(2):137–167.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT press.

Chomsky, N. (2018). Tool module: Chomsky's universal grammar.

Chomsky, N. et al. (2006). On cognitive structures and their development: A reply to piaget. *Philosophy of mind: Classical problems/contemporary issues*, 751.

Chomsky, N. and Lightfoot, D. W. (2002). *Syntactic structures*. Walter de Gruyter.

Chung, J., Ahn, S., and Bengio, Y. (2016). Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*.

Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. (2019). What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy.

Cocke, J. (1969). *Programming languages and their compilers: Preliminary notes*.

Cohen, S., Gimpel, K., and Smith, N. A. (2009). Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems*, volume 21, pages 321–328.

Cohen, S. and Smith, N. A. (2009). Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82, Boulder, Colorado.

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online.

Conneau, A. and Lample, G. (2019). Cross-lingual language model pretraining. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett,

R., editors, *Advances in Neural Information Processing Systems 32*, pages 7059–7069.

Correia, G. M., Niculae, V., Aziz, W., and Martins, A. F. (2020). Efficient marginalization of discrete and structured latent variables via sparsity. In *Advances in Neural Information Processing Systems*.

Corro, C. and Titov, I. (2019a). Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. In *Proceedings of the International Conference on Learning Representations*.

Corro, C. and Titov, I. (2019b). Learning latent trees with stochastic perturbations and differentiable dynamic programming. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5508–5521, Florence, Italy.

Cowie, F. et al. (1999). *What's within?: nativism reconsidered*. Oxford University Press on Demand.

Cross, J. and Huang, L. (2016). Incremental parsing with minimal features using bi-directional lstm. *arXiv preprint arXiv:1606.06406*.

Cui, Y., Che, W., Liu, T., Qin, B., Yang, Z., Wang, S., and Hu, G. (2019). Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Daumé III, H. (2009). Unsupervised search-based structured prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 209–216. ACM.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. (2019). Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Drozdov, A., Rongali, S., Chen, Y.-P., O'Gorman, T., Iyyer, M., and McCallum, A. (2020). Unsupervised parsing with S-DIORA: Single tree encoding for deep inside-outside recursive autoencoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 4832–4845, Online.

Drozdov, A., Verga, P., Yadav, M., Iyyer, M., and McCallum, A. (2019). Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1129–1141, Minneapolis, Minnesota.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2019). Gradient descent provably optimizes over-parameterized neural networks. In *Proceedings of the International Conference on Learning Representations*.

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

Dufter, P. and Schütze, H. (2020). Identifying necessary elements for bert's multilinguality. *arXiv preprint arXiv:2005.00396*.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China.

Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016). Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California.

Dyer, C., Melis, G., and Blunsom, P. (2019). A critical analysis of biased parsers in unsupervised parsing. *arXiv preprint arXiv:1909.09428*.

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

Eriguchi, A., Hashimoto, K., and Tsuruoka, Y. (2016). Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the As-*

sociation for Computational Linguistics (Volume 1: Long Papers), pages 823–833, Berlin, Germany.

Ettinger, A. (2020). What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Evans, N. and Levinson, S. C. (2009). The myth of language universals: Language diversity and its importance for cognitive science. *Behavioral and brain sciences*, 32(5):429–448.

Feldman, J. (1967). *First thoughts on grammatical inference*. Stanford University.

Feldman, J. A., Gips, J., Horning, J. J., and Reder, S. (1969). Grammatical complexity and inference. Technical report, Stanford University.

Finkel, J. R. and Manning, C. D. (2009). Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334, Boulder, Colorado.

Finkel, J. R., Manning, C. D., and Ng, A. Y. (2006). Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626, Sydney, Australia.

Francis, W. N. and Kucera, H. (1979). Brown corpus manual.

Gaddy, D., Stern, M., and Klein, D. (2018). What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 999–1010, New Orleans, Louisiana. Association for Computational Linguistics.

Gaifman, H. (1965). Dependency systems and phrase-structure systems. *Information and control*, 8(3):304–337.

Ganchev, K., Gillenwater, J., Taskar, B., et al. (2010). Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049.

Gelling, D., Cohn, T., Blunsom, P., and Graça, J. (2012). The PASCAL challenge on grammar induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 64–80, Montréal, Canada.

Glynn, P. W. (1987). Likelilood ratio gradient estimation: an overview. In *Proceedings of the 19th conference on Winter simulation*, pages 366–375.

Gold, E. M. (1967). Language identification in the limit. *Information and control*, 10(5):447–474.

Goldberg, Y. (2019). Assessing BERT's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Golland, D., DeNero, J., and Uszkoreit, J. (2012). A feature-rich constituent context model for grammar induction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 17–22, Jeju Island, Korea.

Gómez-Rodríguez, C., Shi, T., and Lee, L. (2018). Global transition-based non-projective dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2664–2675, Melbourne, Australia.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. (2018). Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *Proceedings of the International Conference on Learning Representations*.

Grave, É. and Elhadad, N. (2015). A convex and feature-rich discriminative approach to dependency grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1375–1384, Beijing, China.

Hall Maudslay, R., Valvoda, J., Pimentel, T., Williams, A., and Cotterell, R. (2020). A tale of a probe and a parser. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7389–7395, Online.

Han, W., Jiang, Y., and Tu, K. (2017). Dependency grammar induction with neural lexicalization and big training data. In *Proceedings of the 2017 Conference on Em-*

pirical Methods in Natural Language Processing, pages 1683–1688, Copenhagen, Denmark.

Han, W., Jiang, Y., and Tu, K. (2019). Enhancing unsupervised generative dependency parser with contextual information. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5315–5325, Florence, Italy.

Harper, M. P. and Helzerman, R. A. (1995). Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, 9(3):187–234.

Harris, Z. S. (1951). Methods in structural linguistics.

Hays, D. G. (1964). Dependency theory: A formalism and some observations. *Language*, 40(4):511–525.

He, J., Neubig, G., and Berg-Kirkpatrick, T. (2018). Unsupervised learning of syntactic structure with invertible neural projections. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302, Brussels, Belgium.

Headden III, W. P., Johnson, M., and McClosky, D. (2009). Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado.

Hewitt, J. and Liang, P. (2019). Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China.

Hewitt, J. and Manning, C. D. (2019). A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hinzen, W. (2012). The philosophical significance of universal grammar. *Language Sciences*, 34(5):635–649.

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

Hockett, C. F. (1958). A course in modern linguistics.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65.

Horning, J. J. (1969). A study of grammatical inference. Technical report, Stanford Computer Science Tech Report.

Hovy, E., Gerber, L., Hermjakob, U., Junk, M., and Lin, C.-Y. (2000). Question answering in webclopedia. In *TREC*, volume 52, pages 53–56.

Htut, P. M., Cho, K., and Bowman, S. (2018). Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 371–373, Brussels, Belgium.

Hu, Z., Ma, X., Liu, Z., Hovy, E., and Xing, E. (2016). Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2410–2420, Berlin, Germany.

Huang, Y., Zhang, M., and Tan, C. L. (2012). Improved constituent context model with features. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 564–573.

Hudson, R. (2007). *Language networks: The new word grammar*. Oxford University Press.

Hudson, R. (2010). *An introduction to word grammar*. Cambridge University Press.

Hudson, R. A. (1984). *Word grammar*. Blackwell Oxford.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with Gumbel-softmax. In *Proceedings of the International Conference on Learning Representations*.

Jarvinen, T. and Tapanainen, P. (1998). Towards an implementable dependency grammar. In *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pages 1–10.

Jiang, Y., Han, W., and Tu, K. (2016). Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas.

Jiang, Y., Han, W., and Tu, K. (2017). Combining generative and discriminative approaches to unsupervised dependency parsing via dual decomposition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1689–1694, Copenhagen, Denmark.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. (2019). Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Johansson, R. and Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, pages 105–112, Tartu, Estonia.

Johansson, R. and Nugues, P. (2008). Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187, Manchester, England.

Johnson, M. (2007). Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic.

Johnson, M. and Charniak, E. (2004). A TAG-based noisy-channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 33–39, Barcelona, Spain.

Johnson, M., Griffiths, T., and Goldwater, S. (2007). Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York.

Kalman, R. E. (1964). When is a linear control system optimal?

Kann, K., Cho, K., and Bowman, S. R. (2019). Towards realistic practices in low-resource natural language processing: The development set. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3342–3349, Hong Kong, China.

Karlsson, F. (1990). Constraint grammar as a framework for parsing running text. In *COLNG 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics*.

Karlsson, F., Voutilainen, A., Heikkilä, J., and Anttila, A. (1995). *Constraint Grammar: a language-independent system for parsing unrestricted text*. De Gruyter Mouton.

Kasami, T. (1966). An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Kim, T., Choi, J., Edmiston, D., and goo Lee, S. (2020a). Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *Proceedings of the International Conference on Learning Representations*.

Kim, T., Li, B., and Lee, S.-g. (2020b). Multilingual zero-shot constituency parsing. *arXiv preprint arXiv:2004.13805v2*.

Kim, Y., Denton, C., Hoang, L., and Rush, A. M. (2017). Structured attention networks. In *Proceedings of the International Conference on Learning Representations*.

Kim, Y., Dyer, C., and Rush, A. (2019a). Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy.

Kim, Y., Rush, A., Yu, L., Kuncoro, A., Dyer, C., and Melis, G. (2019b). Unsupervised recurrent neural network grammars. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1105–1117, Minneapolis, Minnesota.

Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas.

Kiperwasser, E. and Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Kitaev, N. and Klein, D. (2018). Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Klein, D. and Manning, C. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485, Barcelona, Spain.

Klein, D. and Manning, C. D. (2002). A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Koo, T., Globerson, A., Carreras, X., and Collins, M. (2007). Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic.

Kool, W., van Hoof, H., and Welling, M. (2020). Estimating gradients for discrete random variables by sampling without replacement. In *Proceedings of the International Conference on Learning Representations*.

Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. (2019). Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China.

Kübler, S., McDonald, R., and Nivre, J. (2009). Dependency parsing. *Synthesis lectures on human language technologies*, 1(1):1–127.

Kuncoro, A., Ballesteros, M., Kong, L., Dyer, C., and Smith, N. A. (2016). Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1744–1753, Austin, Texas.

Kuncoro, A., Kong, L., Fried, D., Yogatama, D., Rimell, L., Dyer, C., and Blunsom, P. (2020). Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 8:776–794.

Kurihara, K. and Sato, T. (2006). Variational bayesian grammar induction for natural language. In *International Colloquium on Grammatical Inference*, pages 84–96. Springer.

Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Lamb, S. M. (1961). On the mechanization of syntactic analysis. *Readings in Machine Translation*, pages 109–114.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. In *Proceedings of the International Conference on Learning Representations*.

Lari, K. and Young, S. J. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Le, H., Vial, L., Frej, J., Segonne, V., Coavoux, M., Lecouteux, B., Allauzen, A., Crabbé, B., Besacier, L., and Schwab, D. (2019). Flaubert: Unsupervised language model pre-training for french. *arXiv preprint arXiv:1912.05372*.

Le, P. and Zuidema, W. (2015). Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 651–661, Denver, Colorado.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lei, T., Barzilay, R., and Jaakkola, T. (2016). Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online.

Li, J., Luong, M.-T., Jurafsky, D., and Hovy, E. (2015). When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Li, J., Zhou, G., and Ng, H. T. (2010). Joint syntactic and semantic parsing of Chinese. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1108–1117, Uppsala, Sweden.

Liang, P., Petrov, S., Jordan, M., and Klein, D. (2007). The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697, Prague, Czech Republic.

Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., and Smith, N. A. (2019a). Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota.

Liu, R., Regier, J., Tripuraneni, N., Jordan, M., and Mcauliffe, J. (2019b). Rao-blackwellized stochastic gradients for discrete distributions. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4023–4031, Long Beach, California, USA.

Liu, Y., Gardner, M., and Lapata, M. (2018). Structured alignment networks for matching sentences. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1554–1564, Brussels, Belgium.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Liu, Y. and Lapata, M. (2018). Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019c). Roberta: A robustly optimized bert pretraining approach. In *arXiv preprint arXiv:1907.11692*.

Maillard, J., Clark, S., and Yogatama, D. (2017). Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *arXiv preprint arXiv:1705.09189*.

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

Mareček, D. and Rosa, R. (2018). Extracting syntactic trees from transformer encoder self-attentions. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 347–349, Brussels, Belgium.

Mareček, D. and Rosa, R. (2019). From balustrades to pierre vinken: Looking for syntax in transformer self-attentions. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 263–275, Florence, Italy.

Mareček, D. and Straka, M. (2013). Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290, Sofia, Bulgaria.

Mareček, D. and Žabokrtský, Z. (2012). Exploiting reducibility in unsupervised dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 297–307, Jeju Island, Korea.

Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de la Clergerie, É. V., Seddah, D., and Sagot, B. (2019). Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*.

Maruyama, H. (1990). Structural disambiguation with constraint propagation. In *28th Annual Meeting of the Association for Computational Linguistics*, pages 31–38.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

McGee, R. J. and Warms, R. L. (2013). *Theory in social and cultural anthropology: An encyclopedia*. Sage Publications.

Mel'cuk, I. A. et al. (1988). *Dependency syntax: theory and practice*. SUNY press.

Menzel, W. and Schroder, I. (1998). Decision procedures for dependency parsing using graded constraints. In *Processing of Dependency-Based Grammars*.

Miao, Y. and Blunsom, P. (2016). Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328, Austin, Texas.

Miao, Y., Yu, L., and Blunsom, P. (2016). Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1727–1736, New York, New York, USA.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1791–1799, Bejing, China.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *Proceedings of the 2013 NIPS Deep Learning Workshop*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020). Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62.

Mou, L., Lu, Z., Li, H., and Jin, Z. (2017). Coupling distributed and symbolic execution for natural language queries. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2518–2526.

Naseem, T., Chen, H., Barzilay, R., and Johnson, M. (2010). Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, MA.

Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Ballesteros, M., Chiang, D., Clothiaux, D., Cohn, T., Duh, K., Faruqui, M., Gan, C., Garrette, D., Ji, Y., Kong, L., Kuncoro, A., Kumar, G., Malaviya, C., Michel, P., Oda, Y., Richardson, M., Saphra, N., Swayamdipta, S., and Yin, P. (2017). Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Niculae, V., Martins, A., Blondel, M., and Cardie, C. (2018). SparseMAP: Differentiable sparse structured inference. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3799–3808, Stockholmsmässan, Stockholm Sweden.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160, Nancy, France.

Nivre, J. (2005). Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D.

(2016). Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia.

Nivre, J., de Marneffe, M.-C., Ginter, F., Hajič, J., Manning, C. D., Pyysalo, S., Schuster, S., Tyers, F., and Zeman, D. (2020). Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France.

Noji, H., Miyao, Y., and Johnson, M. (2016). Using left-corner parsing to encode universal structural constraints in grammar induction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 33–43, Austin, Texas.

Paskin, M. A. (2002). Grammatical bigrams. In *Advances in Neural Information Processing Systems*, pages 91–97.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pate, J. K. and Johnson, M. (2016). Grammar induction from (lots of) words alone. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 23–32, Osaka, Japan.

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

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Peters, M., Neumann, M., Zettlemoyer, L., and Yih, W.-t. (2018b). Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium.

Pimentel, T., Valvoda, J., Hall Maudslay, R., Zmigrod, R., Williams, A., and Cotterell, R. (2020). Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online.

Poliak, A., Naradowsky, J., Haldar, A., Rudinger, R., and Van Durme, B. (2018). Hypothesis only baselines in natural language inference. In *Proc. 7th Joint Conf. Lexical and Computational Semantics*, pages 180–191.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Raganato, A. and Tiedemann, J. (2018). An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium.

Rasooli, M. S. and Faili, H. (2012). Fast unsupervised dependency parsing with arc-standard transitions. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 1–9, Avignon, France.

Ren, Y., Guo, S., Labeau, M., Cohen, S. B., and Kirby, S. (2020). Compositional languages emerge in a neural iterated learning model. In *Proceedings of the International Conference on Learning Representations*.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in bertology: What we know about how bert works. *arXiv preprint arXiv:2002.12327*.

Russell, S. (1998). Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103.

Sachan, D. S., Zhang, Y., Qi, P., and Hamilton, W. (2020). Do syntax trees help pre-trained transformers extract information? *arXiv preprint arXiv:2008.09084*.

Sag, I. A., Wasow, T., and Bender, E. M. (2003). *Syntactic theory: A formal introduction*. Center for the Study of Language and Information, Stanford, CA.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Proceedings of the 2019 NeurIPS Workshop on Energy Efficient Machine Learning and Cognitive Computing*.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Schwartz, R., Abend, O., Reichart, R., and Rappoport, A. (2011). Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 663–672, Portland, Oregon, USA.

Seddah, D., Kübler, S., and Tsarfaty, R. (2014). Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland.

Seddah, D., Tsarfaty, R., Kübler, S., Candito, M., Choi, J. D., Farkas, R., Foster, J., Goenaga, I., Gojenola Galletebeitia, K., Goldberg, Y., Green, S., Habash, N., Kuhlmann, M., Maier, W., Nivre, J., Przepiórkowski, A., Roth, R., Seeker, W., Versley, Y., Vincze, V., Woliński, M., Wróblewska, A., and Villemonte de la Clergerie, E. (2013). Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA.

Seginer, Y. (2007). Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, Prague, Czech Republic.

Sgall, P., Hajicová, E., Hajicová, E., Panevová, J., and Panevova, J. (1986). *The meaning of the sentence in its semantic and pragmatic aspects*. Springer Science & Business Media.

Shen, Y., Lin, Z., Jacob, A. P., Sordoni, A., Courville, A., and Bengio, Y. (2018a). Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia.

Shen, Y., Lin, Z., wei Huang, C., and Courville, A. (2018b). Neural language modeling by jointly learning syntax and lexicon. In *Proceedings of the International Conference on Learning Representations*.

Shen, Y., Tan, S., Hosseini, A., Lin, Z., Sordoni, A., and Courville, A. C. (2019a). Ordered memory. In *Advances in Neural Information Processing Systems*, volume 32, pages 5037–5048.

Shen, Y., Tan, S., Sordoni, A., and Courville, A. (2019b). Ordered neurons: Integrating tree structures into recurrent neural networks. In *Proceedings of the International Conference on Learning Representations*.

Shi, H., Livescu, K., and Gimpel, K. (2020). On the role of supervision in unsupervised constituency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7611–7621, Online.

Shi, H., Mao, J., Gimpel, K., and Livescu, K. (2019). Visually grounded neural syntax acquisition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1842–1861, Florence, Italy.

Shi, H., Zhou, H., Chen, J., and Li, L. (2018). On tree-based neural sentence modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4631–4641, Brussels, Belgium.

Shi, T., Huang, L., and Lee, L. (2017). Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of*

the 2017 Conference on Empirical Methods in Natural Language Processing, pages 12–23, Copenhagen, Denmark.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Sipser, M. (1996). Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29.

Sleator, D. D. and Temperley, D. (1993). Parsing english with a link grammar. In *Third International Workshop on Parsing Technologies (IWPT)*, pages 277–292.

Smith, N. A. and Eisner, J. (2005). Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*, pages 73–82.

Smith, N. A. and Eisner, J. (2006). Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 569–576, Sydney, Australia.

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

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA.

Solomonoff, R. J. (1959). A new method for discovering the grammars of phrase structure languages. In *Communications of the ACM*, volume 2, pages 20–20. Association For Computing Machinery.

Solomonoff, R. J. (1964). A formal theory of inductive inference. *Information and control*, 7(2):224–254.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. (2019). MASS: Masked sequence to sequence pre-training for language generation. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5926–5936, Long Beach, California, USA.

Spitkovsky, V. I., Alshawi, H., Chang, A. X., and Jurafsky, D. (2011). Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1281–1290, Edinburgh, Scotland, UK.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2009). Baby steps: How "less is more" in unsupervised dependency parsing. *NIPS: Grammar Induction, Representation of Language and Language Learning*, pages 1–10.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2010). From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2012). Three dependency-and-boundary models for grammar induction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 688–698, Jeju Island, Korea.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2013). Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1983–1995, Seattle, Washington, USA.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. (2019a). Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical*

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4323–4332, Hong Kong, China.

Sun, Y., Wang, S., Li, Y., Feng, S., Chen, X., Zhang, H., Tian, X., Zhu, D., Tian, H., and Wu, H. (2019b). Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Sun, Y., Wang, S., Li, Y.-K., Feng, S., Tian, H., Wu, H., and Wang, H. (2020). Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8968–8975.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China.

Tang, R., Lu, Y., Liu, L., Mou, L., Vechtomova, O., and Lin, J. (2019). Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Taylor, W. L. (1953). "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

Tenney, I., Das, D., and Pavlick, E. (2019a). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S., Das, D., and Pavlick, E. (2019b). What do you learn from

context? probing for sentence structure in contextualized word representations. In *Proceedings of the International Conference on Learning Representations*.

Tesnière, L. (1959). Éléments de syntaxe structurale.

Tsai, H., Riesa, J., Johnson, M., Arivazhagan, N., Li, X., and Archer, A. (2019). Small and practical BERT models for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3632–3636, Hong Kong, China.

Tsarfaty, R., Nivre, J., and Andersson, E. (2011). Evaluating dependency parsing: Robust and heuristics-free cross-annotation evaluation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 385–396, Edinburgh, Scotland, UK.

Tu, K. and Honavar, V. (2012). Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1324–1334.

Turc, I., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Well-read students learn better: The impact of student initialization on knowledge distillation. *arXiv preprint arXiv:1908.08962*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Virtanen, A., Kanerva, J., Ilo, R., Luoma, J., Luotolahti, J., Salakoski, T., Ginter, F., and Pyysalo, S. (2019). Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*.

Voita, E. and Titov, I. (2020). Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 183–196, Online.

Wang, P. and Blunsom, P. (2013). Collapsed variational bayesian inference for pcfgs. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 173–182.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. (2020). Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*.

Wang, Y., Lee, H.-Y., and Chen, Y.-N. (2019). Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1061–1070, Hong Kong, China.

Warstadt, A., Cao, Y., Grosu, I., Peng, W., Blix, H., Nie, Y., Alsop, A., Bordia, S., Liu, H., Parrish, A., Wang, S.-F., Phang, J., Mohananey, A., Htut, P. M., Jeretic, P., and Bowman, S. R. (2019). Investigating BERT's knowledge of language: Five analysis methods with NPIs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2877–2887, Hong Kong, China.

White, L. and White, L. (2003). *Second language acquisition and universal grammar*. Cambridge University Press.

Williams, A., Drozdov, A., and Bowman, S. R. (2018a). Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267.

Williams, A., Nangia, N., and Bowman, S. R. (2018b). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wu, Z., Chen, Y., Kao, B., and Liu, Q. (2020). Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual*

Meeting of the Association for Computational Linguistics, pages 4166–4176, Online.

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 195–206.

Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32*, pages 5753–5763.

Yogatama, D., Blunsom, P., Dyer, C., Grefenstette, E., and Ling, W. (2017). Learning to compose words into sentences with reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, Toulon, France.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n3. *Information and control*.

Yu, K. and Arkhipov, M. (2019). Adaptation of deep bidirectional multilingual transformers for russian language. *Computational Linguistics and Intellectual Technologies*, (18):333–339.

Zhang, K. and Bowman, S. (2018). Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361, Brussels, Belgium.

Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., and Liu, Q. (2019). ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy.

Zhao, Y. and Titov, I. (2020). Visually grounded compound PCFGs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 4369–4379, Online.

Zhu, H., Bisk, Y., and Neubig, G. (2020). The return of lexical dependencies: Neural lexicalized PCFGs. *Transactions of the Association for Computational Linguistics*.