ENHANCED REPRESENTATIONS AND EFFICIENT ANALYSIS OF SYNTACTIC DEPENDENCIES WITHIN AND BEYOND TREE STRUCTURES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Tianze Shi

August 2021

ENHANCED REPRESENTATIONS AND EFFICIENT ANALYSIS OF SYNTACTIC DEPENDENCIES WITHIN AND BEYOND TREE STRUCTURES

Tianze Shi, Ph.D.

Cornell University 2021

As a fundamental task in natural language processing, dependency-based syntactic analysis provides useful structural representations of textual data. It is supported by an abundance of multilingual annotations and statistical parsers. A common representation format widely adopted by contemporary computational dependency-based syntactic analysis is single-rooted directed trees, where each edge represents a dependency relation. These governor-dependent relations capture bilexical syntactic modifications and facilitate efficient parsing algorithms that break down the analysis of the whole trees into identifications of individual dependency edges. However, it is known that edge-focused dependency-tree representations face practical challenges to properly handle certain linguistic phenomena involving multiple dependency edges, such as valency patterns and certain types of multi-word expressions. Further, dependency tree structures fall short in explicitly representing coordination structures, argument sharing in control and raising constructions, and so on. This thesis aims at addressing the aforementioned issues and improving dependency-based syntactic analysis via augmented and enhanced representations within and beyond tree structures, which involves new challenges in the designs of computational models, learning regimes from empirical data, and inferencing procedures to derive the desired structures.

To guide parsers to consider wider structural contexts and to recognize linguistic constructions as a whole, in addition to predicting individual dependency relations, this thesis introduces two parser designs that combine parsing and tagging modules. In the first parser, taggers are trained to predict valency patterns, which encode the number, types, and linear orderings of each word's dependent syntactic relations (e.g., a transitive verb in English has a subject to its left and a direct object to its right). This method is demonstrated to improve precision on the selected subsets of dependency relations used in the valency patterns. The second effort focuses on headless multi-word expressions (MWEs), which are typically identified with taggers, when full syntactic analysis is not required. By integrating a tagging view of the MWEs into decoding processes, the parsers become more accurate in MWE identification.

Certain syntactic constructions, such as coordination, pose extra representational challenges for dependency trees, and this thesis explores two types of enhanced structures beyond dependency trees and presents methods to analyze natural language texts into those formats. Enhanced Universal Dependencies format removes the tree constraint and the target structures become connected graphs. This thesis details the design of a tree-graph integrated-format parser, which serves as the basis of the winning solution at the IWPT 2021 shared task, in combination with other techniques including a two-stage finetuning strategy and text pre-processing pipelines powered by pre-training. Finally, this thesis revisits Kahane's (1997) idea of bubble trees, which marks span boundaries on top of otherwise dependency-based structures, to provide an explicit mechanism to represent coordination structures. The transition-based system developed to parse into such bubble tree structures shows improvement on the task of coordination structure prediction.

BIOGRAPHICAL SKETCH

Tianze grew up and attended school and college in metropolitan areas in China, but he instantly fell in love with the gorgeous town of Ithaca when he first traveled to the United States to start his Ph.D. journey at Cornell University. His last stop before graduate school was Tsinghua University in Beijing, China, where he first learned how to code and was finally able to apply his computer science skills to help him learn foreign languages, which eventually evolved into an eight-year-long[1] interest in natural language processing. This apparently deviated from Tianze's abandoned childhood dream of becoming an astronaut, but he has come to realize that making machines linguistically capable is by all means an astronomical challenge that is well worth his research endeavors. He plans to continue exploring the "space" of natural language processing even more after his education.

[1] And still counting.

This thesis is dedicated to my parents.

ACKNOWLEDGEMENTS

My deepest gratitude goes to my advisor Lillian Lee. She has been an amazing collaborator, educator, mentor, and friend. I am constantly blown away by how sharp she is in capturing interesting research ideas from all kinds of seemingly unremarkable “little things”, how careful and thorough she is in exhausting all possible ways to investigate and interpret research questions, experiment results, and beyond, and how caring she is for people she knows, including her students, members of Cornell, and the entire NLP community. It is really hard to verbally describe how fortunate I feel about having Lillian as my advisor. She writes on her homepage *“My debt to [my students] is unbounded”*, but in fact *I* am the one owing an unbounded debt to Lillian. Thank you, Lillian! My minor advisor in linguistics, Mats Rooth, is one of my academic role models at Cornell as well. He is incredibly knowledgeable and enthusiastic about linguistic research. I would also like to thank my other former and current committee members, Erik Andersen, and Karthik Sridharan for their feedback.

My Ph.D. experience has benefited greatly from inputs by many people. I thank the entire Cornell NLP group and the NLP seminar attendees, especially Piyush Anand, Maria Antoniak, Yoav Artzi, Ryan Benmalek, Valts Blukis, Rishi Bommasani, Claire Cardie, Jonathan Chang, Xilun Chen, Yao Cheng, Jacob Collard, Cristian Danescu-Niculescu-Mizil, Forrest Davis, Xinya Du, Esin Durmus, Liye Fu, Ge Gao, Max Grusky, John Hale, Yiqing Hua, Jack Hessel, Arzoo Katiyar, Moontae Lee, Jixing Li, David Mimno, Dipendra Misra, Vlad Niculae, Joseph Rhyne, Xanda Schofield, Ana Smith, Alane Suhr, Kai Sun, Marten van Schijndel, Chenhao Tan, Laure Thompson, Felix Wu, Bishan Yang, Justine Zhang, and Karen Zhou, for discussions and feedback. My summer internships at Microsoft DMX Group and Bloomberg AI Group have been eye-opening and rewarding. I especially thank my internship hosts and mentors: Adrian Benton, Ozan İrsoy, Igor Malioutov, and Kaushik Chakrabarti. I would also like to thank my collaborators and coauthors not yet mentioned above: Jordan Boyd-Graber, Luyao Chen, Tao Chen, Weizhu Chen, Hal Daumé III, Shreya Dixit, Heyang Er, Alexander Fabbri, Carlos Gómez-Rodríguez, Liang Huang, Youxuan Jiang, Walter Lasecki, Suyi Li, Zifan Li, Zihan Li, Xi Victoria Lin, Yi Mao, Bo Pang, Oleksandr Polozov, Dragomir Radev, Sungrok Shim, Richard Socher, Yi Chern Tan, Kedar Tatwawadi, Caiming Xiong, Eric Xue, Michihiro Yasunaga, Tao Yu, Rui Zhang, Vincent Zhang, Yuwen Zhang, and Chen Zhao.

I am grateful to my undergrad research mentors, colleagues, coauthors, and collaborators who first introduced me to the world of research: Ao Chen, Xinxiong Chen, Keikichi Hirose, Shun Kasahara, Yosuke Kashiwagi, Yankai Lin, Yang Liu, Zhiyuan Liu, Nobuaki Minematsu, Teeraphon Pongkittiphan, Daisuke Saito, Maosong Sun, Cunchao Tu, Hidetsugu Uchida, Ruobin Xie, Liner Yang, Congying Zhang, and Yu Zhao. During my Ph.D. research, I have gained so much motivation from numerous peer researchers in the field, especially people in the parsing community who have given me detailed, valuable, and constructive feedback and comments on my work anonymously. In particular, I got very inspired to continue doing research on syntactic parsing after getting to know many wonderful people through the CoNLL 2017 shared task: Wanxiang Che, Miryam de Lhoneux, Timothy Dozat, Filip Ginter, Jenna Kanerva, Joakim Nivre, Peng Qi, Djamé Seddah, Xiang Yu, and Daniel Zeman.

Cornell is an incredibly supportive community, and I would like to thank Junwen Bai, Eugene Bagdasaryan, Soumya Basu, Ethan Cecchetti, Rahul Chatterjee, Danlu Chen, Xiangyu Chen, Gabriela Calinao Correa, Cong Ding, Jasmim Drigo, Yawen Fang, Molly Feldman, Joshua Gancher, Jing Gao, Jake Gardner, Mia Gong, Chuan Guo, Xun Huang, Nielson Hul, Junteng Jia, George Karagiannis, Kevin Kwong, Eric Lee, Boyi Li, Yixuan Li, Wei-Kai Lin, Fujun Luan, Cheng Perng Phoo, Geoff Pleiss, Dongping Qi, Yexin Qu, Maithra Raghu, Tobias Schnabel, Ashudeep Singh, Zhao Shen, Becky Stewart, Yi Su, Yu Sun, Wil Thomason, Andreas Veit, Lequn Wang, Qianqian Wang, Yan Wang, Ruihan Wu, Mengqi Xia, Himank Yadav, Guandao Yang, Maofan Ted Yin, Yurong You, Fan Zhang, Tianyi Zhang, Wenting Zhao, Xinran Zhu, and Lingzi Zhuang. I would also like to thank my intern mates who added a lot of fun to the summers: Ao, Dong, Hongyuan, Huazheng, Leslie, Lin, Lisa, Pei, Philippe, Sameer, Shi, Shruti, Shuohang, Taining, Xuezhou, Yanqing, Youwei, Yujia, and Zhuoyue.

Special thanks to my friends in the FF group, Youer, Yue, Yun, and Yunhe, who gave me so much warmth during the sometimes-cold Ithaca winters and shared literal tears and laughter with me. Thanks to Zikai and Tian for always being supportive. I also thank Tommy for his encouragement throughout the good and rough times.

Finally, my parents and my family have given me tremendous support throughout my life. I thank them for their love and understanding even when I have been more than 10,000 kilometers away from them for the past few years.

The work in this thesis was supported in part by NSF grant SES-1741441, a Google Focused Research Grant to Cornell University, and a Bloomberg Data Science Ph.D. Fellowship.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Syntactic parsing, or automatic syntactic analysis, is a fundamental task in natural language processing (NLP). It involves identification of grammatical relations within an input sentence, which are useful to a wide range of downstream tasks. For instance, an automatic parser is expected to recognize that the verb *"like"* in the example sentence *"I like syntactic parsers"* has subject *"I"* and object *"syntactic parsers"*, where the object phrase can be further analyzed as a noun *"parsers"* modified by an adjective *"syntactic"*. Downstream applications then have access to a structural "blueprint" of how the interpretation of the full sentence decomposes into meanings of its smaller internal units, in addition to the linear arrangements of the words in the sentence.

Dependency parsing has gained popularity in the NLP community in the past decade, due to wide availability of data annotations, efficient and accurate parser implementations, and the simplicity of the representations. A typical output structure from a dependency-based syntactic analysis is a labeled directed tree spanning over the set of words from an input sentence. The direction of an arc denotes the asymmetric bilexical relation between the governing and the dependent word. Figure 1.1 shows an example dependency parse tree, where the arc labeled nsubj indicates that the word *"I"* is a nominal subject of the verb

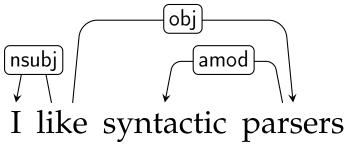


Figure 1.1: An example dependency tree.

"like".

The simplicity of the bilexical relations and the tree representations is a double-edged sword. On the one hand, it facilitates applications of dependency parse trees and developments of automatic parsers by non-experts of the underlying linguistic theories. This point is well articulated by de Marneffe and Nivre (2019, pg. 208):

> *[T]he core structure of dependency trees, that is, binary relations between lexical elements forming a tree, is a conceptually simple representation [...] Anyone can grasp the notion of subject and object and understand that some words can be modified by others. Dependency grammar thus offers a representation that is usable by anyone who wants to build or use systems for text understanding, not only (computational) linguists [...]*

On the other hand, tree structures consisting of only word-word modifications can be indirect, awkward, or even insufficient in handling certain syntactic phenomena. The named entities *"Ezra Cornell"* and *"Mary Ann Wood"* in Figure 1.2 do not have any clear internal modification relations, but flat arcs are introduced to satisfy the representational constraints demanded by a tree structure. Further, there is no direct way of representing modifier sharing across conjuncts in a coordination structure, which leaves the scope of the modifier *"young"* ambiguous as of whether it modifies only the first conjunct or the entire coordinated phrase.

The central aim of this thesis is to improve dependency tree-based syntactic analysis. Specifically, this thesis explores two technical routes, one that preserves the tree-shaped representations but supplements parsers with alternative views of the same underlying structures, and another route that loosens the

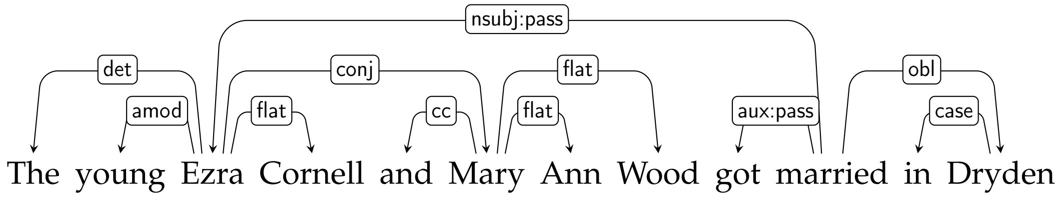

nsubj:pass
det
conj
flat
obl
amod
flat
cc
flat
aux:pass
case
The young Ezra Cornell and Mary Ann Wood got married in Dryden

Figure 1.2: An example dependency tree with coordination and named entities.

tree requirements to allow additional mechanisms for explicit handling of certain syntactic phenomena such as coordination. Both technical routes involve associated new challenges in the designs of computational models, learning regimes from empirical data, and inferencing procedures to derive the desired structures. Each piece of work in this thesis investigates a different method of enhancing the representations, discusses solutions to the aforementioned challenges, and presents empirical improvements on parsing the respective targeted syntactic constructions.

1.1 Motivation: Syntactic Parsing in NLP

In NLP systems, syntactic parsing is typically not a standalone task. Rather, it is commonly used to facilitate downstream text-processing modules. This section first showcases a (small) selection of recent integrations of parsing into a wide range of NLP tasks to give practical motivation for the task of syntactic parsing, and then discusses the desiderata of an "ideal" syntactic representation.

1.1.1 (Selected) Recent Applications of Parsing

Syntactic analysis provides useful information on the compositional structures of the input texts and can be generally useful for many natural language understanding tasks. Among those, predicate-argument structural analysis, or semantic role labeling (SRL; Palmer et al., 2010), bears a close relationship to syntax. Strubell et al. (2018) and Swayamdipta et al. (2018) obtain superior SRL accuracy by supervising the model with dependency parsing signals along with the main task of SRL, and more recently, Shi et al. (2020) reformulate the task of SRL into dependency parsing, based on the empirical evidence that a small set of structural configurations in the dependency trees account for a majority of the SRL relations. The task of coreference resolution, another "core" NLP task, aims at identifying and clustering all the entity mentions in a given document, and a recent work by Jiang and Cohn (2021) presents improvements on this task by incorporating both dependency parse trees and SRL relations. In the context of relation extraction, Zhang et al. (2018) introduce a technique based on graph neural networks to encode pruned dependency trees, which is effective in capturing and leveraging long-range syntactic relations. For the task of question answering, Reddy et al. (2017) generate ungrounded logical forms based on syntactic dependency trees, which are then matched to knowledge graphs to derive answers.

Syntactic structures are also beneficial to natural language generation tasks. In abstractive summarization, Song et al. (2018) propose a copy mechanism that guides their model to copy words from dependency-parsed source documents into summary sentences. Zhang et al. (2019b) design a variational auto-encoder based on syntactic trees for improving the grammaticality of the generated sentences, and show improvements on both language modeling and unsupervised paraphrase generation. Machine translation is another popular area where syntactic parsing is proven to be helpful. Bastings et al. (2017) and Zhang et al. (2019a) enhance the encoders in their machine translation system with source-side syntactic parsing, and Gū et al. (2018) and Akoury et al. (2019) exploit syntactic structures during decoding by first generating the syntactic constituent labels before producing the lexicalized target sentences.

Last but not least, syntactic parses can help researchers gain insights into language variation and change. For example, Johannsen et al. (2015) relate demographic factors including age and gender to syntactic variations observed on automatically parsed texts obtained from an online review website. Newberry et al. (2017) study the grammatical changes in English based on a large-scale diachronic corpus automatically annotated with syntactic dependency trees (Lin et al., 2012).

1.1.2 Is Parsing (Still) Useful?

With the recent successful applications of large-scale language models trained on massive corpora (Peters et al., 2018; Devlin et al., 2019; Brown et al., 2020), the state-of-the-art paradigm in NLP research appears to be shifting towards end-to-end modeling through finetuning language models directly on end tasks, without the need of constructing any intermediate representations including syntactic parse trees. It becomes an increasingly relevant question whether parsing is still useful in NLP. By using structural probes, Hewitt and Manning (2019) and Chi et al. (2020) observe the presence of syntactic information in large-scale language models that are trained without explicit syntactic supervision, which invites further discussion on whether any explicit syntactic supervision is necessary. Glavaš and Vulić (2021) examine the effect of intermediate parsing training during language model finetuning for downstream language understanding tasks and they report *“very limited and inconsistent effect”*, but they conclude their paper with a footnote (pg. 3098) stating that *“formalized syntactic structures will still be an important source of inductive bias, especially in setups without sufficient text data for large-scale pretraining”*.[1]

Indeed, the power of syntactic analysis comes from structural abstraction and generalization, which are especially important in low-resource scenarios. In the setting of cross-lingual event detection, Liu et al. (2019) use a syntax-based encoder that abstracts out surface order differences across different languages to facilitate cross-lingual transfer. Similarly, Ahmad et al. (2021) show that dependency trees are helpful in a graph convolutional network-based models for cross-lingual relation and event extraction.

Along with the reduction of the amount of supervision data are interpretability and controllability of NLP systems. While large-scale learning-based neural models achieve good accuracies on benchmarks, they are not the only requirements for successfully deploying an NLP system. Chiticariu et al. (2013) observe a disproportionately large number of rule-based information extraction systems in industrial and commercial settings, compared with the dominance of learning-based systems in academic publications, and they explain the divide (in part) by the emphasis of interpretability and controllability in real-world de-

[1] Similar investigation on the utility of supervised syntactic parsing has been performed more than a decade ago by Bod (2007), who concluded at the time that *“in the field of syntax-based language models the end of supervised parsing has come in sight”*. But this conclusion has not stopped NLP researchers to find other useful applications of parsing (see, for example, §1.1.1).

ployment. Syntactic structures are among the top candidates to address these practical concerns, and thus they are widely adopted in rule-based systems. Valenzuela-Escárcega et al. (2015, 2020) present rule-based information extraction systems based on syntactic dependency trees. The rule-based open information extraction system developed by Zhang et al. (2017a) operates completely on dependency parse trees, and the rule sets are sharable across languages due to the use of language-universal syntactic relations. Dhole and Manning (2020) use dependency trees and shallow semantic analysis to write rules for question generation.

Additionally, as an integral part of human languages, syntactic performance is also desirable for artificial intelligence systems to generate human-like utterances. While current large-scale language models show significant improvements on standard evaluation metrics such as perplexity as bigger models are trained on larger amount of data, these performance gains do not always translate into better syntactic generalization (Hu et al., 2020). Shen et al. (2021) show that models receiving explicit syntactic supervision achieve better perplexity and syntactic generation with a smaller number of parameters. Kuncoro et al. (2020) also provide further evidence that language models become more effective at downstream structured prediction tasks when they are equipped with more explicit syntactic biases.

The question proposed in this subsection is about the practical value of syntactic analysis in NLP, so correspondingly, the answer has to be an empirical one, and there may be no universal conclusions across all task settings, evaluation targets, data domains, and languages. Further, variations in syntactic representations result in different coverage of syntactic content, difficulties in parsing, and usefulness to NLP systems. The following subsection discusses the (sometimes conflicting) design criteria for an end-task-friendly syntactic representation; these motivate this thesis's focus on specific syntactic constructions such as coordination.

1.1.3 Desiderata of an "Ideal" Syntactic Representation (in NLP)

From an end-task point of view, a good syntactic representation needs to satisfy many factors. The following are some (but not the only) practical considerations:

- **Coverage**: For an NLP system to benefit from having an intermediate syntactic representation, it needs to have access to the syntactic structures that are likely to be informative to solving the target task. For example, an information extraction module to be eventually used for a question answering system may rely on extracted factual statements from texts, and it is important to have a direct representation of predicate-argument and coordination structures.
- **Accuracy**: The inclusion of intermediate structures opens up risks of error propagation. Thus, it is crucial to have highly accurate automatic syntactic parsers. Despite targeting the same syntactic phenomena, different representations may lead to different difficulties in model learning and parsing (Schwartz et al., 2012).
- **Availability**: The success of modern machine-learning-based NLP systems relies heavily on the availability of high-quality and large-quantity

data annotations. It is especially challenging to support accurate text analysis for low-resource languages where data annotations are scarce. Cross-lingual knowledge transfer is a promising technique to alleviate this issue, but it requires a common representation framework with consistent annotations across multiple languages.

- **Simplicity**: As argued in the beginning of this chapter, conceptually simple syntactic structures are more accessible to NLP practitioners and researchers. Further, representations describable in simpler formal/theoretical terms are likely to support algorithmic designs better if the same structures have been applied to other research domains and the associated techniques can be adapted to parsing. For example, spanning trees, the currently dominant representation format in dependency-based syntactic analysis, are well-studied in graph theory, and it is advantageous to have access to the existing literature on efficient algorithms for finding the maximum spanning tree for a given graph.

The (incomplete list of) factors above all contribute to the overall usefulness of the intermediate syntactic structures. As discussed in the beginning of this chapter, simplicity is an advantage for dependency-based syntactic analysis. Additionally, recent releases of large collections of multilingual treebanks (Nivre et al., 2016, 2020) support applications of dependency parsing in lower-resource languages. This thesis aims to improve dependency-based syntactic analysis mostly on the *"coverage"* and *"accuracy"* aspects, while preserving its *"simplicity"* by either maintaining the canonical dependency tree representation or modifying the tree constraint to handle certain constructions that cannot be easily modeled by trees.

1.2 Road Map

The rest of this thesis is organized into 5 chapters.

Chapter 2 through Chapter 5 present four published projects focusing on different issues in edge-factored dependency tree structures regarding specific syntactic constructions and then proposing solutions to enhance the representations, as well as computational models to parse the new structures. These four chapters are divided into two parts: Chapter 2 and Chapter 3 preserve the tree structures, but enforce prediction consistency across a tagging view and a parsing view of the same data; Chapter 4 and Chapter 5 deviate from dependency trees and adopt more expressive structures to better support syntactic analysis of certain syntactic constructions including coordination.

Chapter 2 revisits the idea of valency from dependency grammar, where core arguments of a predicate are not only analyzed as independent dependency edges, but also part of the predicate's valency pattern. Valency patterns can be used to represent, for example, the different numbers and types of arguments of transitive and intransitive verbs. Biasing parsers towards seen valency patterns during training can also reduce the parsers' "excessive creativity" in terms of supposedly-stable core argument structures during inference. This chapter presents a valency-augmented decoder that factors in the number and types of each token's modifiers. Empirical multilingual evaluation shows that this method leads to improved precision on selected subsets of relations, including and beyond core arguments.

Chapter 3 highlights the issue of recognizing headless multi-word expressions in dependency trees. Despite that a flat named entity has no clear internal structures, a tree format forces the selection of a representational head. The example sentence in Figure 1.2 contains two such instances (*"Ezra Cornell"* and *"Mary Ann Wood"*). Outside of parsing, named entities are typically extracted by a tagging solution. This chapter empirically compares parsing and tagging approaches to extracting headless multi-word expressions and presents a joint decoder that combines model scores from these two views of the same underlying structures.

Chapter 4 presents a parser for enhanced Universal Dependencies, where the output syntactic structures are not trees, but connected graphs. A larger number of candidate dependency relations in a graph compared with a tree allows better representations of argument sharing in raising, control, relative clauses, and coordination constructions. This chapter also explores multilingual training and language-specific finetuning to improve model accuracy on low-resource languages. Models developed in this chapter obtained the best performance among all system submissions to the IWPT 2021 shared task of multilingual parsing from raw texts to enhanced Universal Dependencies.

Chapter 5 focuses on coordination structures. As illustrated in Figure 1.2, commonly adopted dependency tree representations are insufficient to disambiguate the scope of certain modifiers within coordinated phrases. This chapter proposes using Kahane's (1997) bubble trees to allow explicit markings of coordinated phrase boundaries, symmetric relations among conjuncts within the same coordinated phrase, and unambiguous private/shared modifier attachments. This chapter proposes a transition system to parse into such enriched bubble tree representations and presents empirical improvements on the task of coordination structure prediction.

Chapter 6 concludes this thesis with suggestions for future work.

1.3 Bibliographic Notes

The main contents of this thesis are based on published conference papers co-authored with Lillian Lee.

- Chapter 2 appeared as "Tianze Shi and Lillian Lee. 2018. Valency-augmented dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1291, Brussels, Belgium. Association for Computational Linguistics". We thank the three anonymous reviewers for their insightful comments, Jungo Kasai for assistance in setting up the TAG parsing experiments, and Xilun Chen, Jason Eisner, Jungo Kasai and Ana Smith for discussion and comments. We also thank CoNLL'17 shared task organizers and participants for publicizing system outputs.
- Chapter 3 appeared as "Tianze Shi and Lillian Lee. 2020. Extracting headless MWEs from dependency parse trees: Parsing, tagging, and joint modeling approaches. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8780–8794, Online. Association for Computational Linguistics". We thank the three anonymous reviewers for their comments, and Igor Malioutov, Ana Smith and the Cornell NLP group for discussion and comments.
- Chapter 4 appeared as "Tianze Shi and Lillian Lee. 2021a. TGIF: Tree-graph integrated-format parser for enhanced UD with two-stage generic-to individual-language finetuning. In *Proceedings of the 17th International*

Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies, Online. Association for Computational Linguistics". We thank the anonymous reviewers for their detailed and constructive feedback.

- Chapter 5 appeared as "Tianze Shi and Lillian Lee. 2021b. Transition-based bubble parsing: Improvements on coordination structure prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics". We thank the anonymous reviewers for their constructive comments, Yue Guo for discussion, and Hiroki Teranishi for help with experiment setup.

The Use of Pronouns in This Thesis Chapter 2 through Chapter 5 use the pronoun "we" as they are based on collaborative work with Lillian Lee.

CHAPTER 2

VALENCY

In this chapter, we present a complete, automated, and efficient approach for utilizing valency analysis in making dependency parsing decisions. It includes extraction of valency patterns, a probabilistic model for tagging these patterns, and a joint decoding process that explicitly considers the number and types of each token's syntactic dependents. On 53 treebanks representing 41 languages in the Universal Dependencies data, we find that incorporating valency information yields higher precision and F1 scores on the core arguments (subjects and complements) and functional relations (e.g., auxiliaries) that we employ for valency analysis. Precision on core arguments improves from 80.87 to 85.43. We further show that our approach can be applied to an ostensibly different formalism and dataset, Tree Adjoining Grammar as extracted from the Penn Treebank; there, we outperform the previous state-of-the-art labeled attachment score by 0.7. Finally, we explore the potential of extending valency patterns beyond their traditional domain by confirming their helpfulness in improving PP attachment decisions. Our implementation is available at `https://github.com/tzshi/valency-parser-emnlp18`.

2.1 Introduction

Many dependency parsers treat attachment decisions and syntactic relation labeling as two independent tasks, despite the fact that relation labels carry important subcategorization information. For example, the number and types of the syntactic arguments that a predicate may take is rather restricted for natural languages — it is not common for an English verb to have more than one

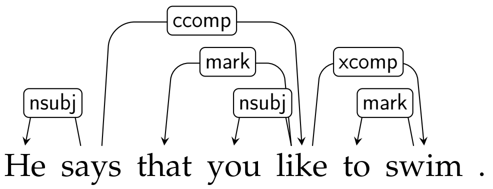


Figure 2.1: Sample annotation in UD, encoding the core valency pattern nsubj ⋄ ccomp for *"says"*, nsubj ⋄ xcomp for *"like"*, and so on (see §2.2-§2.4)

syntactic subject or more than two objects.

In this work, we present a parsing approach that explicitly models subcategorization of (some) syntactic dependents as *valency patterns* (see Figure 2.1 for examples), and operationalize this notion as extracted supertags.[1] An important distinction from prior work is that our definition of valency-pattern supertags is relativized to a user-specified subset of all possible syntactic relations (see §2.3). We train supertaggers that assign probabilities of potential valency patterns to each token, and leverage these probabilities during decoding to guide our parsers so that they favor more linguistically plausible output structures.

We mainly focus on two subsets of relations in our analysis, those involving core arguments and those that represent functional relations, and perform experiments over a collection of 53 treebanks in 41 languages from the Universal Dependencies dataset (UD; Nivre et al., 2017). Our valency-aware parsers improve upon strong baseline systems in terms of output linguistic validity, measured as the accuracy of the assigned valency patterns. They also have higher precision and F1 scores on the subsets of relations under analysis, suggesting a potentially controlled way to balance precision-recall trade-offs.

We further show that our approach is not limited to a particular treebank annotation style. We apply our method to parsing another grammar formal-

[1] The term *"supertags"* is borrowed from Joshi and Srinivas (1994) to refer to elementary structures (valency patterns in our case) that are associated with lexical items.

ism, Tree Adjoining Grammar, where dependency and valency also play an important role in both theory and parser evaluation. Our parser reaches a new state-of-the-art LAS score of 92.59, with more than 0.6 core-argument F1-score improvement over our strong baseline parser.

Finally, we demonstrate the applicability of our valency analysis approach to other syntactic phenomena less associated with valency in its traditional linguistic sense. In a case study of PP attachment, we analyze the patterns of two syntactic relations commonly used in PP attachment, and include them in the joint decoding process. Precision of the parsers improves by an absolute 3.30% on these two relation types.

2.2 Syntactic Dependencies and Valencies

According to Nivre (2005), the modern dependency grammar can be traced back to Tesnière (1959), with its roots reaching back several centuries before the Common Era. The theory is centered on the notion of *dependency*, an asymmetrical relation between words of a sentence. Tesnière distinguishes three node types when analyzing simple predicates: verb equivalents that describe actions and events, noun equivalents as the arguments of the events, and adverb equivalents for detailing the (temporal, spatial, etc.) circumstances. There are two types of relations: (1) verbs dominate nouns and adverbs through a dependency relation; (2) verbs and nouns are linked through a *valency* relation. Tesnière compares a verb to an atom: a verb can attract a certain number of arguments, just as the valency of an atom determines the number of bonds it can engage in (Ágel and Fischer, 2015). In many descriptive lexicographic works (Helbig and

Dataset	Subset	Syntactic Relations
UD	Core	nsubj, obj, iobj, csubj, ccomp, xcomp
	Func.	aux, cop, mark, det, clf, case
	PP (§2.8)	nmod, obl
TAG	Core	0 (subject), 1 (object), 2 (indirect object)
	Co-head	CO

Table 2.1: Sets of syntactic relations we used for valency analysis. UD subsets come from the official categorization in the annotation guidelines.

Schenkel, 1959; Herbst et al., 2004), valency is not limited to verbs, but also includes nouns and adjectives. For more on the linguistic theory, see Ágel et al. (2003, 2006).

Strictly following the original notion of valency requires distinguishing between arguments and adjuncts, as well as obligatory and optional dependents. However, there is a lack of consensus as to how these categorizations may be distinguished (Tutunjian and Boland, 2008), and thus we adopt a more practical definition in this work.

2.3 Computational Representation

Formally, we fix a set of syntactic relations $\mathcal{R}$, and define the *valency pattern* of a token w_i with respect to $\mathcal{R}$ as the linearly-ordered[2] sequence $a_{-j} \cdots a_{-1} \diamond a_1 \cdots a_k$: the $\diamond$ symbol denotes the center word w_i, and each a_l asserts the ex-

[2] Our approach, whose full description is in §2.5, can be adapted to cases where linear ordering is de-emphasized. The algorithm merely requires a distinction between left and right dependents. We choose to encode linearity since it appears that most languages empirically exhibit word order *preferences* even if they allow for relatively free word order.

istence of a word w dominated by w_i via relation $a_l \in \mathcal{R}$, $w_i \xrightarrow{a_l} w$. For a_l and a_m, when $l < m$, the syntactic dependent for a_l linearly precedes the syntactic dependent for a_m. As an example, consider the UD-annotated sentence in Figure 2.1. The token *"says"* has a core-relation[3] valency pattern nsubj $\diamond$ ccomp, and *"like"* has the pattern nsubj $\diamond$ xcomp. If we consider only functional relations, both *"like"* and *"swim"* have the pattern mark $\diamond$.[4] We sometimes employ the abbreviated notation $\alpha^L \diamond \alpha^R$, where α indicates a sequence and the letters L and R distinguish left dependencies from right dependencies.

We make our definition of valency patterns dependent on choice of $\mathcal{R}$ not only because some dependency relations are more often obligatory and closer to the original theoretical definition of valency, but also because the utility of different types of syntactic relations can depend on the downstream task. For example, purely functional dependency labels are semantically vacuous, so they are often omitted in the semantic representations extracted from dependency trees for question answering (Reddy et al., 2016, 2017). There are also recent proposals for parser evaluation that downplay the importance of functional syntactic relations (Nivre and Fang, 2017).

2.4 (Retrospective) Pilot Study: Sanity Checks

We consider two questions that need to be addressed at the outset:[5]

[3] UD core and functional relations are listed in Table 2.1.

[4] The (possibly counterintuitive) direction for *"that"* and *"to"* is a result of UD's choice of a content-word-oriented design.

[5] We actually performed these sanity checks after implementation of and experiments with our approach, because we missed this idea originally, perhaps in part because it requires access to test sets that we abstained from looking at during model development.

1. How well do the extracted patterns generalize to unseen data?
2. Do state-of-the-art parsers already capture the notion of valency implicitly, though they are not explicitly optimized for it?

The first question checks the feasibility of learning valency patterns from a limited amount of data; the second probes the potential for any valency-informed parsing approach to improve over current state-of-the-art systems.

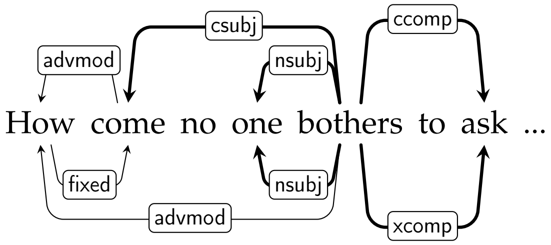
To answer these questions, we use the UD 2.0 dataset for the CoNLL 2017 shared task (Zeman et al., 2017) and the system outputs[6] of the top five performing submissions (Dozat et al., 2017; Shi et al., 2017b; Björkelund et al., 2017; Che et al., 2017; Lim and Poibeau, 2017). Selection of treebanks is the same as in §2.6. We extract valency patterns relative to the set of 6 UD core arguments given in Table 2.1 because they are close to the original notion of valency and we hypothesize that these patterns should exhibit few variations. This is indeed the case: the average number of valency patterns we extract is 110.4 per training treebank, with Turkish (`tr`) having the fewest at 34, and Galician (`gl`) having the most at 298 patterns. We observe that in general, languages with higher degree of flexibility in word order tend to generate more patterns in the data, as our patterns encode linear word order information.

Next, we extract valency patterns from the test set and compare them against those from the training set. On average, out of the 55.4 patterns observed in the gold-standard test sets, only 5.5, or 9.98%, are new and unseen with respect to training. In comparison, 36.2% of the word types appearing in the test sets are not seen during training. This suggests that the valency pattern space is

[6] Retrieved from https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2424.

relatively restricted, and the patterns extracted from training sets do generalize well to test sets.

Finally, we consider the average number of valency patterns extracted from the top-performing system outputs and the number of those not observed in training.[7] All 5 systems are remarkably "hallucinatory" in inventing valency relations, introducing 16.8 to 35.5 new valency patterns, significantly larger than the actual number of unseen patterns. Below we show an error committed by the state-of-the-art Dozat et al. (2017) parser (upper half) as compared to the gold-standard annotation (lower half), and we highlight the core argument valency relations of the verb *"bothers"* in bold. The system incorrectly predicts *"how come"* to be a clausal subject.



Each such non-existent new pattern implies at least some (potentially small) parsing error that can contribute to the degradation of downstream task performance.

[7] The CoNLL 2017 shared task is an end-to-end parsing task, so the participating systems do not have access to gold-standard tokenization, which is a potential explanation for the presented analysis. On the other hand, the conclusion still holds even if we restrict to system outputs with perfect or nearly perfect segmentations.

2.5 Valency-Aware Dependency Parsing

2.5.1 Overview

Our model is based on the following probability factorization for a given sentence $x = w_1, \ldots, w_n$ and parse tree y for x:

$$P(y|x) = \frac{1}{Z_x} \prod_{i=1}^{n} P(v_i|w_i) P(h_i|w_i) P(r_i|w_i, h_i),$$

where Z_x is the normalization factor, v_i is the valency pattern extracted for w_i from y, h_i is the index of the syntactic governor of w_i, and r_i is the syntactic relation label of the dependency relation between w_{h_i} and w_i. We first assume that we have a feature extractor that associates each token in the sentence w_i with a contextualized feature vector $\mathbf{w}_i$, and explain how to calculate the factored probabilities (§2.5.2). Then we discuss decoding (§2.5.3) and training (§2.5.4). Our decoder can be viewed as a special-case implementation of head-automaton grammars (Alshawi, 1996; Eisner and Satta, 1999). Finally, we return to the issue of feature extraction (§2.5.5).

2.5.2 Parameterization

We parameterize $P(v_i|w_i)$ as a softmax distribution over all candidate valency patterns:

$$P(v_i|w_i) \;\propto\; \exp(\text{score}^{\text{VAL}}_{v_i}(\mathbf{w}_i)),$$

where $\text{score}^{\text{VAL}}$ is a multi-layer perceptron (MLP).

For each word w_i, we generate a probability distribution over all potential syntactic heads in the sentence (Zhang et al., 2017b). After we have selected the head of w_i to be w_{h_i}, we decide on the syntactic relation label based on another probability distribution. We use two softmax functions:

$$\begin{aligned} P(h_i|w_i) &\propto \exp(\text{score}^{\text{HEAD}}(\mathbf{w}_{h_i}, \mathbf{w}_i)), \\ P(r_i|w_i, h_i) &\propto \exp(\text{score}_{r_i}^{\text{LABEL}}(\mathbf{w}_{h_i}, \mathbf{w}_i)), \end{aligned}$$

where both $\text{score}^{\text{HEAD}}$ and $\text{score}^{\text{LABEL}}$ are parameterized by deep biaffine scoring functions (Dozat and Manning, 2017).

2.5.3 Decoding

For joint decoding, we adopt Eisner's (1996) algorithm to handle valency patterns as the state information in Eisner and Satta (1999). The algorithm is depicted in Figure 2.2. For each complete and incomplete span, visualized as triangles and trapezoids respectively, we annotate the head with its valency pattern. We adopt Earley's (1970) notation of $\bullet$ to outward-delimit the portion of a valency pattern, starting from the center word $\diamond$, that has already been collected within the span. INIT generates a minimal complete span with hypothesized valency pattern; the $\bullet$ is put adjacent to $\diamond$. COMB matches an incomplete span to a complete span with compatible valency pattern, yielding a complete analysis on the relevant side of $\diamond$. LINK either advances the $\bullet$ by attaching a syntactic dependent with the corresponding relation label, or attaches a dependent with a relation label irrelevant to the current valency analysis. This algorithm can be easily extended to cases where we analyze multiple subsets of valency relations simultaneously: we just need to annotate each head with multiple layers

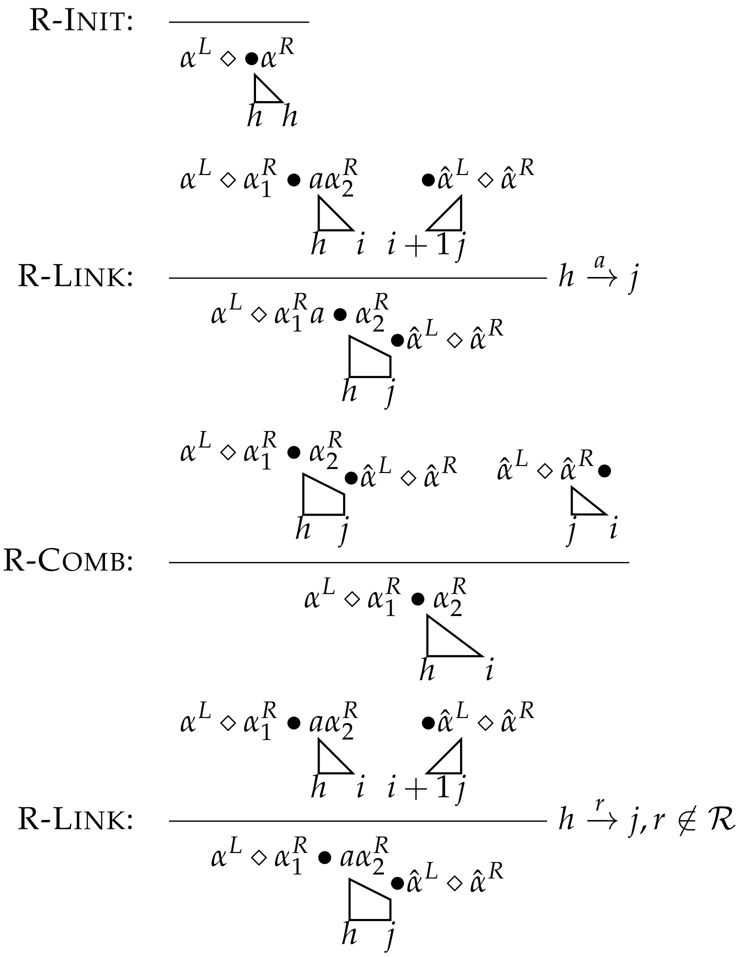


Figure 2.2: Eisner's (1996)/Eisner and Satta's (1999) algorithm, with valency-pattern annotations, incorporated as state information, shown explicitly. We show only the R-rules; the L-rules are symmetric.

of valency patterns, one for each subset.[8]

The time complexity of a naïve dynamic programming implementation is $O(|V|^2|\alpha|n^3)$, where $|V|$ is the number of valency patterns and $|\alpha|$ is the maximum length of a valency pattern. In practice, $|V|$ is usually larger than n, making the algorithm prohibitively slow. We thus turn to A* parsing for a more

[8] To allow our model to account for unseen patterns in new data, we create a special wildcard valency pattern that allows dependents with arbitrary relations in the decoding process, and during training, treat valency patterns occurring fewer than 5 times as examples of the wildcard pattern.

Algorithm 1 Agenda-based best-first parsing algorithm, adapted from Lewis et al. (2016), Alg. 1.

Helper Functions: Init(s) returns the set of spans generated by Init. C.Rules(p) returns the set of spans that can be derived by combining p with existing entries in C through Comb or Link.

```
procedure Parse($s$)
    // Empty priority queue $A$
    $A \leftarrow \emptyset$
    // Initialize $A$ with minimal complete spans
    for $p \in$ Init($s$) do
        $A$.Insert($p$);
    // Empty chart $C$
    $C \leftarrow \emptyset$
    while $A \neq \emptyset$ do
        $p \leftarrow A$.PopMax()
        // Found the global optimal solution
        if $p$ is a full parse then return $p$
        else if $p \notin C$ then
            $C$.Add($p$)
            // Extend the chart
            for $p' \in C$.Rules($p$) do
                $A$.Insert($p'$)
```

practical solution.

A* parsing We take inspiration from A* CCG parsing (Lewis and Steedman, 2014; Lewis et al., 2016; Yoshikawa et al., 2017). The idea (see Algorithm 1) is to estimate the best compatible full parse for every chart item (in our case, complete and incomplete spans), and expand the chart based on the estimated priority scores. Our factorization of probability scores allows the following admissible heuristic: for each span, we can optimistically estimate its best full parse score by assigning to every token outside the span the best possible valency pattern, best possible attachment and best relation label.

2.5.4 Training

We train all components jointly and optimize for the cross entropy between our model prediction and the gold standard, or, equivalently, the sum of the log-probabilities for the three distributions comprising our factorization from §2.5.1. This can be thought of as an instance of multi-task learning (MTL; Caruana, 1997), which has been shown to be useful in parsing (Kasai et al., 2018). To further reduce error propagation, instead of using part-of-speech tags as features, we train a tagger jointly with our main parser components (Zhang and Weiss, 2016).

2.5.5 Feature Extraction

We adopt bi-directional long short-term memory networks (bi-LSTMs; Hochreiter and Schmidhuber, 1997) as our feature extractors, since they have proven successful in a variety of syntactic parsing tasks (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016; Stern et al., 2017; Shi et al., 2017a). As inputs to the bi-LSTMs, we concatenate one pre-trained word embedding, one randomly-initialized word embedding, and the output of character-level LSTMs for capturing sub-token level information (Ballesteros et al., 2015). The bi-LSTM output vectors at each timestep are then assigned to each token as its contextualized representation $\mathbf{w}_i$.

2.6 Experiments

Data and Evaluation Our main experiments are based on UD version 2.0, which was prepared for the CoNLL 2017 shared task (Zeman et al., 2017). We used 53 of the treebanks[9] across 41 languages that have train and development splits given for the shared task. In contrast to the shared-task setting, where word and sentence segmentation are to be performed by the system, we directly use the test-set gold segmentations in order to focus directly on parsing; but this does mean that the performance of our models cannot be directly compared to the officially-reported shared-task results. For evaluation, we report unlabeled and labeled attachment scores (UAS and LAS respectively). Further, we explicitly evaluate precision, recall and F1 scores (P/R/F) for the syntactic relations from Table 2.1, as well as valency pattern accuracies (VPA) involving those relations.

Implementation Details We use three-layer bi-LSTMs with 500 hidden units (250 in each direction) for feature extraction. The valency analyzer uses a one-hidden-layer MLP with ReLU activation function (Nair and Hinton, 2010), while the head selector and labeler use 512- and 128-dimensional biaffine scoring functions respectively. Our models are randomly initialized (Glorot and Bengio, 2010) and optimized with AMSgrad (Reddi et al., 2018) with initial learning rate 0.002. We apply dropout (Srivastava et al., 2014) to our MLPs and variational dropout (Gal and Ghahramani, 2016) to our LSTMs with a keep rate of 0.67 during training.

[9] We exclude the two large treebanks `cs` and `ru_syntagrus` due to experiment resource constraints. There are other Czech and Russian treebanks in our selected collection.

			Core			Func.		
Subsets	UAS	LAS	#	VPA	P / R / F	#	VPA	P / R / F
Baseline	87.59	83.64	2.75	95.83	80.87 / 81.31 / 81.08	4.85	97.51	91.99 / 92.43 / 92.20
Core MTL	87.71	83.80	2.73	96.02	81.96 / **81.98** / 81.96	4.85	97.51	91.96 / 92.50 / 92.23
+ Joint Decoding	87.80	83.93	2.60	**96.68**	**85.43** / 81.75 / **83.53**	4.86	97.50	91.81 / 92.65 / 92.22
Func. MTL	87.67	83.71	2.75	95.80	80.69 / 81.21 / 80.94	4.84	97.58	92.30 / 92.57 / 92.44
+ Joint Decoding	87.72	83.75	2.75	95.80	80.64 / 81.32 / 80.96	4.80	**97.74**	**93.16** / 92.42 / 92.79
Core + Func. MTL	87.67	83.79	2.73	95.99	81.72 / 81.81 / 81.75	4.84	97.59	92.27 / 92.62 / 92.44
+ Joint Decoding	**87.81**	**83.99**	2.63	96.60	84.70 / 81.90 / 83.26	4.82	**97.74**	92.98 / **92.69** / **92.83**

Table 2.2: Macro-averaged results on UD 2.0 across 53 treebanks. VPA=valency pattern accuracy; MTL=multi-task learning; #=average number of predicted attachments per sentence. Best results for each metric are highlighted in bold.

Treebank	Baseline	Joint	ER	Treebank	Baseline	Joint	ER	Treebank	Baseline	Joint	ER
Dutch$_{\text{MAX}}$	84.91	89.83	32.63	Portuguese$_{\text{MAX}}$	92.24	93.46	15.66	Norwegian$_{\text{MIN}}$	90.38	91.41	10.76
Greek	85.82	89.50	25.95	Portuguese$_{\text{MIN}}$	86.90	88.88	15.10	Indonesian	81.53	83.51	10.75
Swedish$_{\text{MAX}}$	86.97	90.21	24.92	Spanish$_{\text{MAX}}$	85.04	87.29	15.06	Latvian	71.33	74.41	10.74
Finnish$_{\text{MAX}}$	88.14	90.87	22.96	Hungarian	79.11	82.13	14.45	French$_{\text{MIN}}$	90.56	91.51	10.01
Italian	87.04	90.00	22.85	Arabic	74.33	77.97	14.16	Basque	76.79	78.97	9.39
Latin$_{\text{MAX}}$	82.52	86.37	22.03	Urdu	70.47	74.63	14.07	Hindi	80.38	82.16	9.04
Danish	85.85	88.93	21.75	Dutch$_{\text{MIN}}$	76.03	79.18	13.13	German	81.05	82.73	8.85
Finnish$_{\text{MIN}}$	87.95	90.44	20.68	Swedish$_{\text{MIN}}$	85.51	87.36	12.76	Czech$_{\text{MIN}}$	76.68	78.64	8.42
Slovenian	86.03	88.59	18.31	Croatian	84.14	86.12	12.46	Polish	86.67	87.72	7.84
Old Slavonic	76.79	81.02	18.25	Gothic	72.15	75.60	12.38	A. Greek$_{\text{MIN}}$	59.00	62.17	7.74
French$_{\text{MAX}}$	89.86	91.71	18.21	A. Greek$_{\text{MAX}}$	74.31	77.48	12.34	Turkish	58.43	61.59	7.61
Estonian	72.02	77.09	18.09	Hebrew	80.27	82.60	11.80	Korean	83.33	84.55	7.31
Slovak	80.39	83.78	17.32	Persian	80.83	83.08	11.72	Chinese	73.81	75.66	7.07
Czech$_{\text{MAX}}$	85.58	88.02	16.90	Norwegian$_{\text{MAX}}$	91.20	92.21	11.50	English$_{\text{MIN}}$	84.69	85.60	5.96
Latin$_{\text{MIN}}$	76.60	80.55	16.89	Catalan	88.19	89.54	11.49	Vietnamese	48.45	51.49	5.91
Romanian	82.60	85.51	16.74	English$_{\text{MID}}$	84.26	86.05	11.37	Galician	72.17	73.70	5.49
English$_{\text{MAX}}$	90.96	92.43	16.20	Spanish$_{\text{MIN}}$	88.72	89.98	11.17	Japanese	91.87	91.88	0.14
Russian	82.17	85.05	16.13	Bulgarian	83.98	85.75	11.02	Average	81.08	83.53	13.80

Table 2.3: Treebank-specific F1 scores on core argument relations, comparing the baseline models to our Core MTL + joint decoding models, sorted by the error reduction (ER, %) rate. When comparing a model with performance s_2 against baseline score s_1, ER is defined as $(s_2 - s_1)/(1 - s_1)$. For languages with two or three treebanks, we include multiple entries differentiated by the subscripts MAX/MID/MIN, corresponding to the treebanks with the highest/median/lowest ER, respectively. A. Greek = Ancient Greek.

Efficiency Our A* parsers are generally reasonably efficient; for the rare ($<$ 1%) cases where the A* search does not finish within 500,000 chart expansion steps, we back off to a model without valency analysis. When analyzing three or more relation subsets, the initialization steps become prohibitively slow due to the large number of valency pattern combinations. Thus, we limit the number of combinations for each token to the highest-scoring 500.

Results on UD We present our main experimental results on UD in Table 2.2. The baseline system does not leverage any valency information (we only train the head selectors and labelers, and use the original Eisner decoder). We compare the baseline to settings where we train the parsers jointly with our proposed valency analyzers, distinguishing the effect of using this information only at training (multi-task learning; MTL) vs. both at training and decoding.

Including valency analysis into the training objective already provides a slight improvement in parsing performance, in line with the findings of Kasai et al. (2018). With our proposed joint decoding, there is a mild improvement to the overall UAS and LAS, and a higher boost to VPA. The output parse trees are now more precise in the analyzed valency relations: on core arguments, precision increases by as much as 4.56. As shown by Table 2.3, the performance gain of joint decoding varies across treebanks, ranging from an error reduction rate of over 30% (Dutch Lassy Small Treebank) on core argument relations to nearly 0% (Japanese). Overall, our approach exhibits a clearly positive impact on most of the treebanks in UD. We do not see performance correlating to language typology, although we do observe smaller error-reduction rates on treebanks with lower baseline performances, that is, on “harder” languages.

			Core		CO	
	UAS	LAS	VPA	P / R / F	VPA	P / R / F
Friedman et al. (2017)	90.31	88.96	–	–	–	–
Kasai et al. (2017)	90.97	89.68	–	–	–	–
Kasai et al. (2018)	93.26	91.89	–	–	–	–
Baseline	93.66	92.44	97.06	92.45 / 92.76 / 92.60	99.22	73.11 / **87.20** / 79.54
Core + CO MTL	93.71	92.53	97.19	92.74 / 93.20 / 92.97	**99.24**	75.43 / 84.44 / 79.68
+ Joint Decoding	**93.75**	**92.59**	**97.47**	**93.27** / **93.22** / **93.24**	**99.24**	**76.06** / 83.70 / **79.70**

Table 2.4: Experimental results for parsing TAGs.

2.7 Parsing Tree Adjoining Grammar

Dependency and valency relations also play an important role in formalisms other than dependency grammar. In this section, we apply our proposed valency analysis to Tree Adjoining Grammar (TAG; Joshi and Schabes, 1997), because TAG derivation trees, representing the process of inserting obligatory arguments and adjoining modifiers, can be treated as a dependency representation (Rambow and Joshi, 1997). We follow prior art and use Chen's (2001) automatic conversion of the Penn Treebank (Marcus et al., 1993) into TAG derivation trees. The dataset annotation has labels 0, 1 and 2, corresponding to subject, direct object, and indirect object; we treat these as our core argument subset in valency analysis.[10] Additionally, we also analyze CO (co-head for phrasal verbs) as a separate singleton subset.[11] We leave out adj (adjuncts) in defining our valency patterns. We strictly follow the experiment protocol of previous work (Bangalore et al., 2009; Chung et al., 2016; Friedman et al., 2017; Kasai et al., 2017, 2018), and report the results in Table 2.4. The findings are consistent with our main experiments: MTL helps parsing performance, and joint decoding further improves on core argument F1 scores, reaching a new state-of-the-art result of 92.59 LAS. The precision/recall trade-off is pronounced for the CO relation subset (up to a precision improvement of 2.95 and a recall loss of 3.50).

[10] We choose not to use the sparse labels 3 and 4, which encode additional complements.

[11] CO relations represent *headed* multi-word expressions (MWEs), while Chapter 3 focuses on *headless* MWEs.

	UAS	LAS	Core P / R / F	Func. P / R / F	PP P / R / F
Baseline	87.59	83.64	80.87 / 81.31 / 81.08	91.99 / 92.43 / 92.20	77.29 / 77.99 / 77.62
PP MTL	87.67	83.70	80.61 / 81.23 / 80.91	92.03 / 92.50 / 92.26	78.30 / **78.38** / 78.32
+ Joint Decoding	87.68	83.69	79.93 / 81.50 / 80.69	91.92 / 92.51 / 92.21	**80.59** / 77.68 / **79.04**
Core + PP MTL	87.70	83.77	81.62 / 81.81 / 81.71	91.93 / 92.52 / 92.22	77.93 / 78.25 / 78.08
+ Joint Decoding	87.80	83.91	**84.18** / **81.97** / **83.05**	91.68 / **92.65** / 92.16	79.71 / 78.03 / 78.83
Core + Func. + PP MTL	87.67	83.75	81.35 / 81.68 / 81.50	92.18 / 92.61 / 92.39	77.99 / 78.22 / 78.08
+ Joint Decoding	**87.81**	**83.94**	83.88 / **81.97** / 82.90	**92.78** / 92.63 / **92.70**	79.54 / 78.11 / 78.78

Table 2.5: Experimental results involving analyzing PPs as valency patterns.

2.8 Case Study on PP Attachment

Although valency information has traditionally been used to analyze complements or core arguments,[12] in this section, we show the utility of our approach in analyzing other types of syntactic relations. We choose the long-standing problem of prepositional phrase (PP) attachment (Hindle and Rooth, 1993; Brill and Resnik, 1994; Collins and Brooks, 1995; de Kok et al., 2017), which is known to be a major source of parsing mistakes (Kummerfeld et al., 2012; Ng and Curran, 2015). In UD analysis, PPs usually have the labels obl or nmod with respect to their syntactic parents, whereas adpositions are attached via a case relation, which is included in the functional relation subset. Thus, we add another relation subset, obl and nmod, to our valency analysis.

Table 2.5 presents the results for different combinations of valency relation subsets. We find that PP-attachment decisions are generally harder to make, compared with core and functional relations. Including them during training distracts other parsing objectives (compare Core + PP with only analyzing Core in §2.6). However, they do permit improvements on precision for PP attachment by 3.30, especially with our proposed joint decoding. This demonstrates the usage of our algorithm outside traditional notions of valency — it can be a general method for training parsers to focus on specific subsets of syntactic relations.

[12] There are also proposals to analyze valency without distinguishing complements and adjuncts (Čech et al., 2010).

2.9 Further Related Work

Supertagging Supertagging (Bangalore and Joshi, 2010) has been proposed for and used in parsing TAG (Bangalore and Joshi, 1999; Nasr and Rambow, 2004), CCG (Curran and Clark, 2003; Curran et al., 2006), and HPSG (Ninomiya et al., 2006; Blunsom and Baldwin, 2006). Within dependency parsing, supertags have also been explored in the literature, but prior work mostly treats them as additional features. Ambati et al. (2013, 2014) use CCG supertags to improve dependency parsing results, while Ouchi et al. (2014, 2016) leverage dependency-based supertags as features. Faleńska et al. (2015) compare supertagging to parser stacking, where they extract supertags from base parsers to provide additional features for stacked parsers, instead of having a supertagger as a separate component.

Constrained Dependency Grammar Another line of research (Wang and Harper, 2004; Foth et al., 2006; Foth and Menzel, 2006; Bharati et al., 2002, 2009; Husain et al., 2011) utilizes supertags in dependency parsing within the framework of constraint dependency grammar (CDG; Maruyama, 1990; Heinecke et al., 1998). Constraints in CDG may be expressed in very general terms (and are usually hand-crafted for specific languages), so prior work in CDG involves a constraint solver that iteratively or greedily updates hypotheses without optimality guarantees. In contrast, our work focuses on a special form of constraints — the valency patterns of syntactic dependents within a subset of relations — and we provide an efficient A*-based exact decoding algorithm.

Valency in Parsing To the best of our knowledge, there have been few attempts to utilize lexical valency information or to improve specifically on core arguments in syntactic parsing apart from CDG. Øvrelid and Nivre (2007) target parsing core relations in Swedish with specifically-designed features such as animacy and definiteness that are useful in argument realization. Jakubiček and Kovář (2013) leverage external lexicons of verb valency frames for reranking. Mirroshandel et al. (2012, 2013) and Mirroshandel and Nasr (2016) extract selectional constraints and subcategorization frames from large unannotated corpora, and enforce them through forest reranking. Our approach does not rely on external resources or lexicons, but directly extracts valency patterns from labeled dependency parse trees. Earlier works in this spirit include Collins (1997).

Semantic Dependency Parsing and Semantic Role Labeling The notion of valency is also used to describe predicate-argument structures that are adopted in semantic dependency parsing and semantic role labeling (Surdeanu et al., 2008; Hajič et al., 2009; Oepen et al., 2014, 2015). While semantic frames clearly have patterns, previous work (Punyakanok et al., 2008; Flanigan et al., 2014; Täckström et al., 2015; Peng et al., 2017; He et al., 2017) incorporates several types of constraints, including uniqueness and determinism constraints that require that certain labels appear as arguments for a particular predicate only once. They perform inference through integer linear programming, which is usually solved approximately, and cannot easily encode linear ordering constraints for the arguments.

A* parsing Best-first search uses a heuristic to expand the parsing chart instead of doing so exhaustively. It was first applied to PCFGs (Ratnaparkhi, 1997;

Caraballo and Charniak, 1998; Sagae and Lavie, 2006), and then to dependency parsing (Sagae and Tsujii, 2007; Zhao et al., 2013; Vaswani and Sagae, 2016). Our probability factorization permits a simple yet effective A* heuristic. A* parsing was introduced for parsing PCFGs (Klein and Manning, 2003; Pauls and Klein, 2009), and has been widely used for grammar formalisms and parsers with large search spaces, for example CCG (Auli and Lopez, 2011) and TAG (Waszczuk et al., 2016, 2017). Our decoder is similar to the supertag and dependency factored A* CCG parser (Yoshikawa et al., 2017), which in turn builds upon the work of Lewis and Steedman (2014) and Lewis et al. (2016). Our model additionally adds syntactic relations into the probability factorizations.

2.10 Chapter Summary and Future Work

We have presented a probability factorization and decoding process that integrates valency patterns into the parsing process. The joint decoder favors syntactic analyses with higher valency-pattern supertagging probabilities. Experiments on a large set of languages from UD show that our parsers are more precise in the subset of syntactic relations chosen for valency analysis, in addition to enjoying the benefits gained from jointly training the parsers and supertaggers in a multi-task learning setting.

Our method is not limited to a particular type of treebank annotation or a fixed subset of relations. We draw similar conclusions when we parse TAG derivation trees. Most interestingly, in a case study on PP attachment, we confirm the utility of our parsers in handling syntactic relations beyond the traditional domain of valency.

A key insight of this work that departs from prior work on automatic extraction of supertags from dependency annotations is that our definition of valency patterns is relativized to a subset of syntactic relations. This definition is closer to the linguistic notion of valency and alleviates the data sparsity problems in that the number of extracted valency patterns is small. At the same time, the patterns generalize well, and empirically, they are effective in our proposed joint decoding process.

Our findings point to a number of directions for future work. First, the choice of subsets of syntactic relations for valency analysis impacts the parsing performance in those categories. This may suggest a controllable way to address precision-recall trade-offs targeting specific relation types. Second, we experimented with a few obvious subsets of relations; characterizing what subsets can be most improved with valency augmentation is an open question. Finally, our decoder builds upon projective dependency-tree decoding algorithms. In the future, we will explore the possibility of removing the projective constraint and the tree requirement, extending the applicability of valency patterns to other tasks such as semantic role labeling.

CHAPTER 3

HEADLESS MULTI-WORD EXPRESSIONS

This chapter focuses on a representational constraint in a tree-shaped representation regarding an interesting and frequent type of multi-word expression (MWE), the headless MWE, for which there are no true internal syntactic dominance relations. Examples include many named entities (*"Wells Fargo"*) and dates (*"July 5, 2020"*) as well as certain productive constructions (*"blow for blow"*, *"day after day"*). Despite their special status and prevalence, current dependency-annotation schemes require treating such flat structures as if they had internal syntactic heads, and most current parsers handle them in the same fashion as headed constructions. Meanwhile, outside the context of parsing, taggers are typically used for identifying MWEs, but taggers might benefit from structural information. We empirically compare these two common strategies—parsing and tagging—for predicting flat MWEs. Additionally, we propose an efficient joint decoding algorithm that combines scores from both strategies. Experimental results on the MWE-Aware English Dependency Corpus and on six non-English dependency treebanks with frequent flat structures show that: (1) tagging is more accurate than parsing for identifying flat-structure MWEs, (2) our joint decoder reconciles the two different views and, for non-BERT features, leads to higher accuracies, and (3) most of the gains result from feature sharing between the parsers and taggers.

3.1 Introduction

Headless multi-word expressions (MWEs), including many named entities and certain productive constructions, are frequent in natural language and are im-

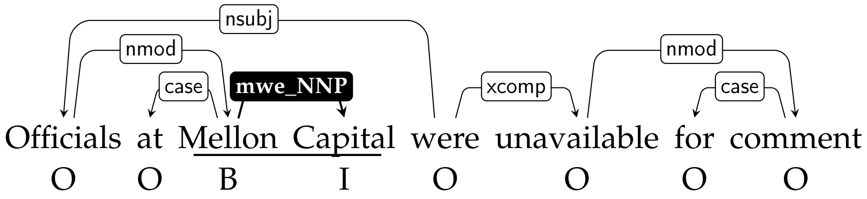


Figure 3.1: Dependency tree from the MWE-Aware English Dependency Corpus, imposing a "head" relationship between the words in the actually headless MWE Mellon Capital. Also shown are MWE BIO labels.

portant to NLP applications. In the context of dependency-based syntactic parsing, however, they pose an interesting representational challenge. Dependency-graph formalisms for syntactic structure represent lexical items as nodes and head-dominates-modifier/argument relations between lexical items as directed arcs on the corresponding pair of nodes. Most words can be assigned clear linguistically-motivated syntactic heads, but several frequently occurring phenomena do not easily fit into this framework, including punctuation, coordinating conjunctions, and "flat", or headless MWEs. While the proper treatment of headless constructions in dependency formalisms remains debated (Kahane et al., 2017; Gerdes et al., 2018), many well-known dependency treebanks handle MWEs by giving their component words a "default head", which is not indicative of a true dominance relation, but rather as *"a tree encoding of a flat structure without a syntactic head"* (de Marneffe and Nivre, 2019, pg. 213). Figure 3.1 shows an example: the headless MWE *"Mellon Capital"* has its first word, *"Mellon"*, marked as the "head" of *"Capital"*.

Despite the special status of flat structures in dependency tree annotations, most state-of-the-art dependency parsers treat all annotated relations equally, and thus do not distinguish between headed and headless constructions. When headless-span identification (e.g., as part of named-entity recognition (NER)) is the specific task at hand, begin-chunk/inside-chunk/outside-chunk (BIO) tagging (Ramshaw and Marcus, 1995) is generally adopted. It is therefore natural to ask whether *parsers* can be as accurate as *taggers* in identifying these "flat branches" in dependency trees. Additionally, since parsing and tagging represent two different views of the same underlying structures, can *joint decoding* that combines scores from the two modules and/or *joint training* under a multitask learning (MTL) framework derive more accurate models than parsing or tagging alone?

To facilitate answering these questions, we introduce a joint decoder that finds the maximum sum of scores from both BIO tagging and parsing decisions. The joint decoder incorporates a special deduction item representing continuous headless spans, while retaining the cubic-time efficiency of projective dependency parsing. The outputs are consistent structures across the tagging view and the parsing view.

We perform evaluation of the different strategies on the MWE-Aware English Dependency Corpus and treebanks for five additional languages from the Universal Dependencies 2.2 corpus that have frequent multi-word headless constructions. On average, we find taggers to be more accurate than parsers at this task, providing 0.59% (1.42%) absolute higher F1 scores with(out) pre-trained contextualized word representations. Our joint decoder combining jointly-trained taggers and parsers further improves over the tagging strategy by 0.69% (1.64%) absolute with(out) pre-trained contextualized word embeddings. This corroborates early evidence (Finkel and Manning, 2009) that joint modeling with parsing improves over NER. We also show that neural representation sharing through MTL is an effective strategy, as it accounts for a large portion of our observed improvements. Our code is publicly available at https://github.com/tzshi/flat-mwe-parsing.

3.2 Background on Headless Structures

A (multi-word) headless construction, or flat structure, is a span of lexical items that together reference a single concept and where no component is a syntactically more plausible candidate for the span's head than any other component. Examples are boldfaced in the following English sentences.

(1) *Within the scope of this work:*

a. ACL starts on **July 5, 2020**.

b. My bank is **Wells Fargo**.

c. The candidates matched each other **insult for insult**. (Jackendoff, 2008)

(1)a and (1)b show that dates and many named entities can be headless constructions, suggesting that such constructions are frequent. Indeed, in the MWE-Aware English Dependency Corpus (Kato et al., 2017), nearly half of the sentences contain headless constructions, 75% of which are named entities. For comparison, (2) shows examples of non-flat MWEs, which are also interesting and important, but are not the focus of this chapter.

(2) *Outside the scope of this work:*

a. **congressman at large** (Sag et al., 2002) [head = *"congressman"*]

b. I have **moved on**. [verb-particle construction, head = *"moved"*]

c. I **take** your argument **into account**. (Constant et al., 2017) [light-verb construction, head = *"take"*]

Returning to headless MWEs, the choice of representation for headless spans depends on the task. In *named-entity recognition*, such spans are often treated as BIO tag sequences:[1] for example, in Figure 3.1, *"Mellon"* is tagged as "B" and *"Capital"* is tagged as "I". In *dependency parsing*, where labeled dependency arcs are the only way to express a syntactic analysis (short of treating MWEs as atomic lexical items, which would result in a chicken-and-egg problem) is to impose arcs within the MWE's span. Different corpora adopt different annotation conventions. The MWE-Aware English Dependency Corpus uses the arc label `mwe_NNP`, as shown in Figure 3.1. The Universal Dependencies (UD; Nivre et al., 2018a) annotation guidelines have all following tokens in such constructions attached to the first one via arcs labeled flat, a choice that is admittedly *"in principle arbitrary"*.[2]

The frequency of flat structures across different treebanks varies according to language, genre, and even tokenization guidelines, among other factors. Table 3.1 lists the UD 2.2 treebanks with the highest and lowest percentage of flat relations. While the Korean treebank `ko_gsd` (with the highest percentage) splits up most names into multiple tokens and connects them through flat, the Japanese treebank `ja_gsd` (no flats at all) treats all names as compound nouns, and thus represents them as having internal structure without any indication that a special case has occurred.[3] Figure 3.2 shows examples from the UD par-

[1] In this work, we adopt the original BIO tagset, which cannot properly represent discontinuous MWEs. See Schneider et al. (2014) for modified tagsets providing such support.

[2] `https://universaldependencies.org/u/dep/flat.html`

[3] Some flat structures can end up using other dependency labels such as compound, as a result of the fact that many UD treebanks, including `ja_gsd`, are automatically converted from non-UD style annotations. The UD annotations depend on how detailed the original syntactic

Treebank (Language)	% of flat	
	graphs ↓	arcs
19 treebanks with highest percentages:		
`ko_gsd` (Korean)	67.84	15.35
`id_gsd` (Indonesian)	61.63	9.39
`ca_ancora` (Catalan)	41.11	3.32
`nl_lassysmall` (Dutch)	38.90	5.87
`ar_nyuad` (Arabic)	37.63	2.19
`es_ancora` (Spanish), `sr_set` (Serbian), `it_postwita` (Italian), `pt_bosque` (Portuguese), `pt_gsd` (Portuguese), `fa_seraji` (Persian), `de_gsd` (German), `hu_szeged` (Hungarian), `fr_gsd` (French), `es_gsd` (Spanish), `he_htb` (Hebrew), `kk_ktb` (Kazakh), `be_hse` (Belarusian), `nl_alpino` (Dutch)	$>$ 20.00	
…	…	
12 treebanks without flat arcs:		
`cs_cltt` (Czech), `grc_perseus` (Ancient Greek), `hi_hdtb` (Hindi), `ja_gsd` (Japanese), `ja_bccwj` (Japanese), `la_ittb` (Latin), `la_perseus` (Latin), `no_nynorsklia` (Norwegian), `swl_sslc` (Swedish Sign Language), `ta_ttb` (Tamil), `ur_udtb` (Urdu), `vi_vtb` (Vietnamese)	0.00	0.00

Table 3.1: The UD 2.2 training treebanks with highest and lowest percentage of flat arcs, out of 90 treebanks.

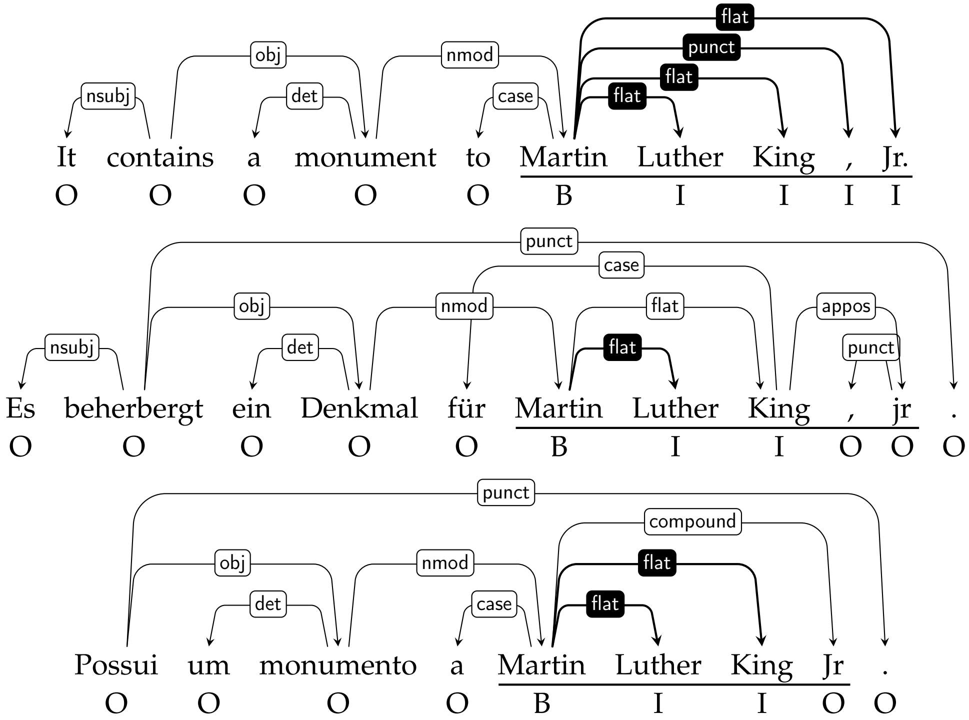


Figure 3.2: An illustration of flat-structure annotation variation across treebanks: a set of parallel sentences, all containing the conceptually headless MWE *"Martin Luther King, Jr."* (underlined), from UD 2.2 (treebank code _pud) in English, German, Portuguese, Chinese, Japanese, and Turkish (top to bottom, continued on the next page). The intent of this figure is not to critique particular annotation decisions, but to demonstrate the notation, concepts, and data extraction methods used in our work. To wit: Highlights/black-background indicate well-formed flat-MWE tree fragments according to the principles listed in §3.4. BIO sequences are induced by the longest-spanning flat arcs. When there is a mismatch between the highlighted tree fragments and the BI spans—here, in the German, Chinese and Turkish examples—it is because the dependency trees do not fully conform to the UD annotation guidelines on headless structures. For example, the word *"King"* in the German example is attached via a flat relation, so it should be but is not a leaf node (§3.4). See footnote 3 for the use of compound relations in the Portuguese and Japanese examples.

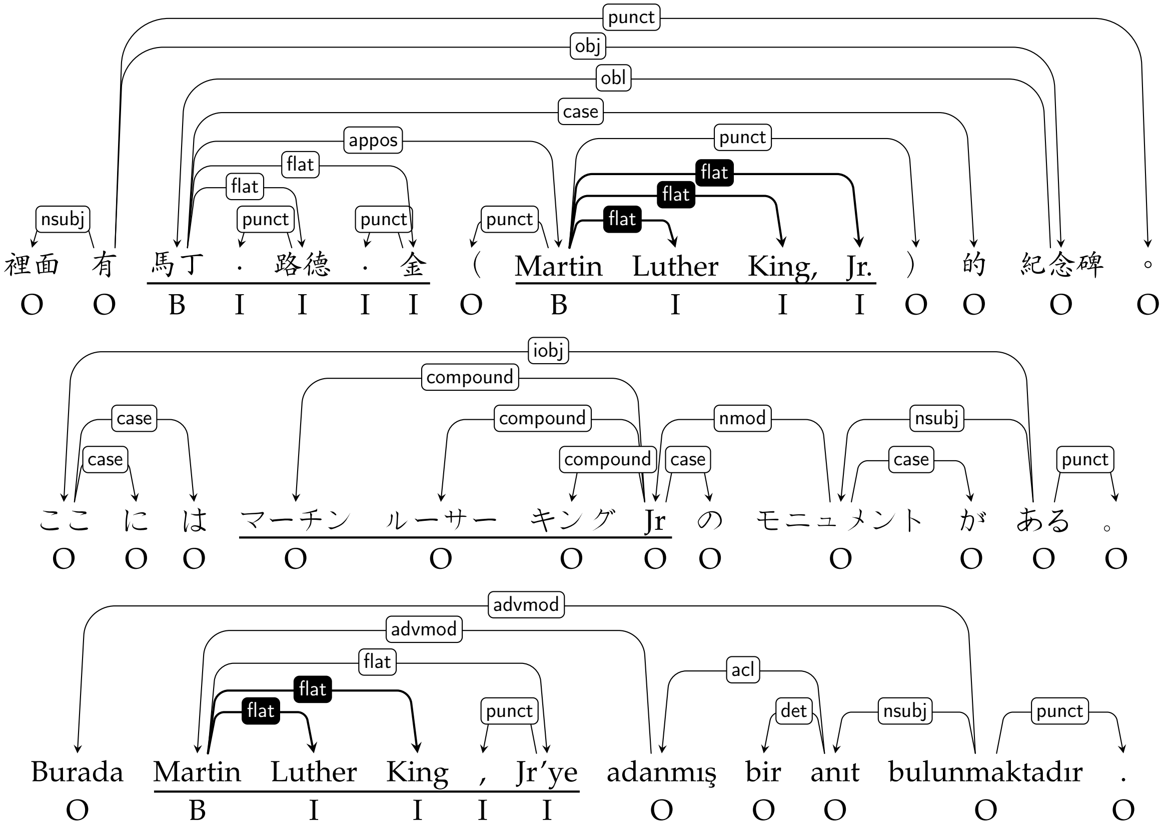

punct
obj
obl
case
appos
punct
flat
flat
flat
flat
nsubj
punct
punct
punct
flat
裡面 有 馬丁 · 路德 · 金 （ Martin Luther King, Jr. ） 的 紀念碑 。
O O B I I I I O B I I I O O O O
iobj
compound
compound
nmod
nsubj
case
case
compound
case
case
punct
ここ に は マーチン ルーサー キング Jr の モニュメント が ある 。
O O O O O O O O O O O O
advmod
advmod
flat
acl
flat
flat
punct
det
nsubj
punct
Burada Martin Luther King , Jr’ye adanmış bir anıt bulunmaktadır .
O B I I I I O O O O O

Figure 3.2: (Continued from the previous page) An illustration of flat-structure annotation variation across treebanks: a set of parallel sentences, all containing the conceptually headless MWE *“Martin Luther King, Jr.”* (underlined), from UD 2.2 (treebank code _pud) in English, German, Portuguese, Chinese, Japanese, and Turkish (top to bottom).

allel treebanks, illustrating the diversity of annotation for the same sentence rendered in different languages.

Overall, more than 20% of the treebanks in the UD 2.2 collection have flat structures in more than 20% of their training-set sentences.[4] Therefore, a parsing approach taking into account the special status of headless structural representations can potentially benefit models for a large number of languages and treebanks.

3.2.1 Notation and Definitions

Formally, given an n-word sentence $w = w_1, w_2, \ldots, w_n$, we define its dependency structure to be a graph $G = (V, E)$. Each node in V corresponds to a word in the sentence. Each (labeled) edge $(h, m, r) \in E$ denotes a syntactic relation labeled r between the head word w_h and modifier word w_m, where $h, m \in \{0, 1, \ldots, n\}$ and 0 denotes the dummy root of the sentence. Since we work with dependency treebanks, we require that the edges in E form a tree. To represent a multi-word headless span $w_i, \ldots, w_j$, all subsequent words in the span are attached to the beginning word w_i, i.e., $\forall k \in \{i+1, \ldots, j\}, (i, k, \mathsf{f}) \in E$, where f is the special syntactic relation label denoting headless structures (flat in UD annotation). Alternatively, one can also use a BIO tag sequence $T = (t_1, t_2, \ldots, t_n) \in \{\mathrm{B}, \mathrm{I}, \mathrm{O}\}^n$ to indicate the location of any headless spans within w. The headless MWE span $w_i, \ldots, w_j$ has the corresponding tags $t_i = \mathrm{B}$ and $\forall k \in \{i+1, \ldots, j\}$, $t_k = \mathrm{I}$; tokens outside any spans are assigned the tag O. We call G and T *consistent* if they indicate the same set of headless spans for w.

analyses are and the accuracies of the conversion algorithms.

[4] Measured on the 90 treebanks with training splits.

3.3 Three Approaches

We first present the standard approaches of edge-factored parsing (§3.3.2) and tagging (§3.3.3) for extracting headless spans in dependency trees, and then introduce a joint decoder (§3.3.4) that finds the global maximum among consistent (tree structure, tag sequence) pairs.

3.3.1 Preliminaries

Given a length-n sentence w—which we henceforth denote with the variable x for consistency with machine-learning conventions—we first extract contextualized representations from the input to associate each word with a vector $\mathbf{x}_0$ (for the dummy word "root"), $\mathbf{x}_1, \ldots, \mathbf{x}_n$. We consider two common choices of feature extractors: (1) bi-directional long short-term memory networks (bi-LSTMs; Graves and Schmidhuber, 2005) which have been widely adopted in dependency parsing (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017) and sequence tagging (Ma and Hovy, 2016); and (2) the Transformer-based (Vaswani et al., 2017) BERT feature extractor (Devlin et al., 2019), pre-trained on large corpora and known to provide superior accuracies on both tasks (Kitaev et al., 2019; Kondratyuk and Straka, 2019). For BERT models, we fine-tune the representations from the final layer for our parsing and tagging tasks. When the BERT tokenizer renders multiple tokens from a single pre-tokenized word, we follow Kitaev et al. (2019) and use the BERT features from the last token as its representation.

3.3.2 (Edge-Factored) Parsing

Since we consider headless structures that are embedded inside parse trees, it is natural to identify them through a rule-based post-processing step after full parsing. Our parsing component replicates that of the state-of-the-art Che et al. (2018) parser, which has the same parsing model as Dozat and Manning (2017). We treat unlabelled parsing as a head selection problem (Zhang et al., 2017b) with deep biaffine attention scoring:

$$\begin{aligned}
\mathbf{h}_i^{\text{attach}} &= \text{MLP}^{\text{attach-head}}(\mathbf{x}_i) \\
\mathbf{m}_j^{\text{attach}} &= \text{MLP}^{\text{attach-mod}}(\mathbf{x}_j) \\
\mathbf{s}_{i,j} &= [\mathbf{h}_i^{\text{attach}}; 1]^\top U^{\text{attach}} [\mathbf{m}_j^{\text{attach}}; 1] \\
P(h_j = i \mid x) &= \text{softmax}_i(\mathbf{s}_{:,j}),
\end{aligned}$$

where MLP$^{\text{attach-head}}$ and MLP$^{\text{attach-mod}}$ are multi-layer perceptrons (MLPs) that project contextualized representations into a d-dimensional space; $[\cdot; 1]$ indicates appending an extra entry of 1 to the vector; $U^{\text{att}} \in \mathbb{R}^{(d+1)\times(d+1)}$ generates a score $s_{i,j}$ for w_j attaching to w_i (which we can then refer to as the head of w_j, h_j); a softmax function defines a probability distribution over all syntactic head candidates in the argument vector (we use the range operator ":" to evoke a vector); and, recall, we represent potential heads as integers, so that we may write $h_j = i \in \{0, \ldots, n\}$. The model for arc labeling employs an analogous deep biaffine scoring function:

$$\begin{aligned}
\mathbf{h}_i^{\text{rel}} &= \text{MLP}^{\text{rel-head}}(\mathbf{x}_i) \\
\mathbf{m}_j^{\text{rel}} &= \text{MLP}^{\text{rel-mod}}(\mathbf{x}_j) \\
\mathbf{v}_{i,j,r} &= [\mathbf{h}_i^{\text{rel}}; 1]^\top U_r^{\text{rel}} [\mathbf{m}_j^{\text{rel}}; 1] \\
P(r_j = r \mid x, h_j = i) &= \text{softmax}_r(\mathbf{v}_{i,j,:}),
\end{aligned}$$

where r_j is the arc label between w_{h_j} and w_j. The objective for training the parser is to minimize the cumulative negative log-likelihood

$$L^{\text{parse}} = \sum_{(i^*,j^*,r^*)\in E} [-\log P(h_{j^*} = i^* \mid x) \\ - \log P(r_i = r^* \mid x, h_{j^*} = i^*)].$$

After the model predicts a full parse, we extract headless structures as the tokens "covered" by the longest-spanning f-arcs ($f = \mathsf{flat}$ in UD).

3.3.3 Tagging

For extracting spans in texts, if one chooses to ignore the existence of parse trees, BIO tagging is a natural choice. We treat the decision for the label of each token as an individual multi-class classification problem. We let

$$P(t_i = t \mid x) = \text{softmax}_t(\text{MLP}^{\text{tag}}(\mathbf{x}_i)),$$

where MLP$^{\text{tag}}$ has 3 output units corresponding to the scores for tags B, I and O respectively.[5]

We train the tagger to minimize

$$L^{\text{tag}} = \sum_i -\log P(t_i = t_i^* \mid x),$$

where t^* corresponds to the gold BIO sequence. During inference, we predict the BIO tags independently at each token position and interpret the tag

[5] Sequence tagging is traditionally handled by conditional random fields (Lafferty et al., 2001, CRFs). However, in recent experiments using contextualized representations on tagging (Clark et al., 2018; Devlin et al., 2019), CRF-style loss functions provide little, if any, performance gains compared with simple multi-class classification solutions, at slower training speeds, to boot. Our preliminary experiments with both bi-LSTM and BERT-based encoders corroborate these findings, and thus we report results trained without CRFs.

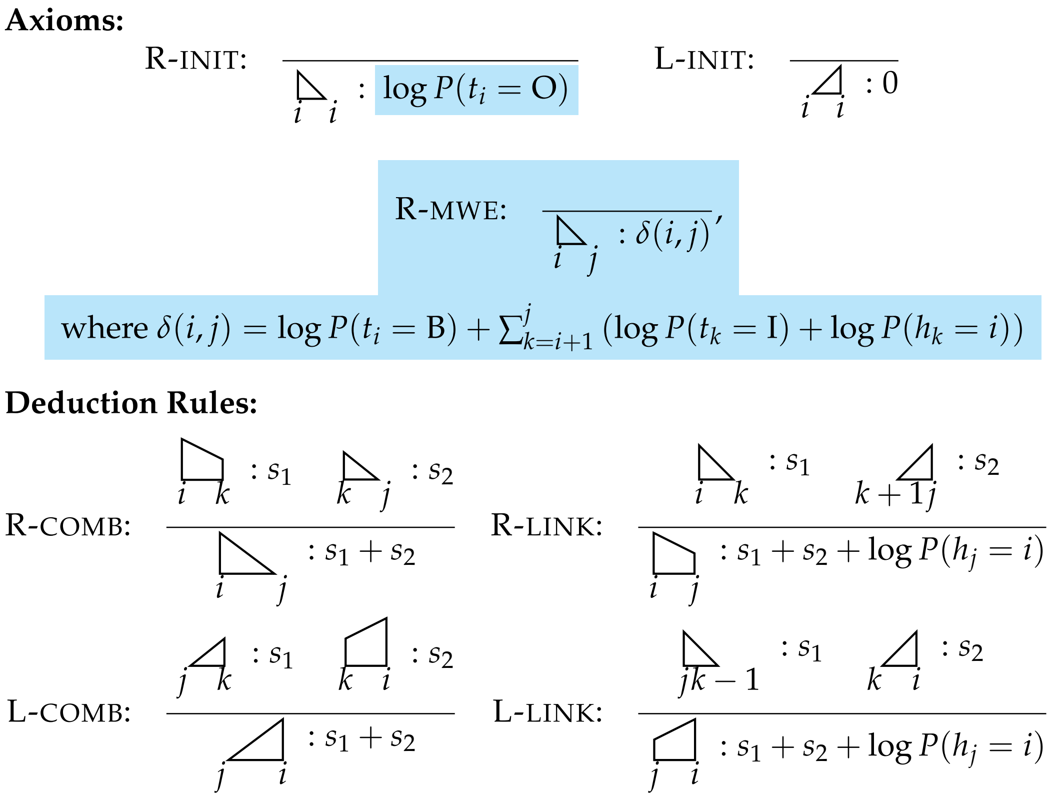

Axioms:
R-INIT:
i i : $\log P(t_i = \mathrm{O})$
L-INIT:
i i : 0
R-MWE:
i j : $\delta(i, j)$,
where $\delta(i, j) = \log P(t_i = \mathrm{B}) + \sum_{k=i+1}^{j} (\log P(t_k = \mathrm{I}) + \log P(h_k = i))$
Deduction Rules:
R-COMB:
i k : s_1
k j : s_2
i j : $s_1 + s_2$
R-LINK:
i k : s_1
$k+1$ j : s_2
i j : $s_1 + s_2 + \log P(h_j = i)$
L-COMB:
j k : s_1
k i : s_2
j i : $s_1 + s_2$
L-LINK:
j $k-1$: s_1
k i : s_2
j i : $s_1 + s_2 + \log P(h_j = i)$

Figure 3.3: Eisner's (1996) algorithm adapted to parsing headless structures (unlabeled case), our modifications highlighted in blue. All deduction items are annotated with their scores. R-MWE combines BIO tagging scores and head selection parsing scores. We need no L-MWE because of the rightward headless-structure-arc convention.

sequence as a set of MWE spans. As a post-processing step, we discard all single-token spans, since the task is to predict multi-word spans.

3.3.4 A Joint Decoder

A parser and a tagger take two different views of the same underlying data. It is thus reasonable to hypothesize that a joint decoding process that combines the scores from the two models might yield more accurate predictions. In this section, we propose such a joint decoder to find the parser+tagger-consistent structure with the highest product of probabilities. Formally, if $\mathcal{Y}$ is the output space for all consistent parse tree structures and BIO tag sequences, for $y \in \mathcal{Y}$ with components consisting of tags t_i, head assignments h_i, and relation labels r_i, our decoder aims to find $\hat{y}$ satisfying

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(y \mid x),$$

where

$$P(y \mid x) = \prod_i P(t_i \mid x) P(h_i \mid x) P(r_i \mid x, h_i).$$

Figure 3.3 illustrates our joint decoder in the unlabeled case.[6] It builds on Eisner's (1996) decoder for projective dependency parsing. In addition to having single-word spans as axioms in the deduction system, we further allow multi-word spans to enter the decoding procedures through the axiom R-MWE. Any initial single-word spans receive an O-tag score for that word, while the newly introduced MWE spans receive B-tag, I-tag, attachment and relation scores that correspond to the two consistent views of the same structure. The time complexity for this decoding algorithm remains the same $O(n^3)$ as the original Eisner algorithm.

During training, we let the parser and the tagger share the same contextualized representation $\mathbf{x}$ and optimize a linearly interpolated joint objective

$$L^{\text{joint}} = \lambda L^{\text{parse}} + (1 - \lambda) L^{\text{tag}},$$

where λ is a hyper-parameter adjusting the relative weight of each module.[7] This is an instance of multi-task learning (MTL; Caruana, 1993, 1997). MTL has

[6] In the labeled case, the parser further adds the arc-labeling scores to the R-MWE and LINK rules.

[7] The joint decoder combines tagging and parsing scores regardless of whether the two modules are jointly trained. However, since feature extraction is the most time-consuming step in our neural models, especially with BERT-based feature extractors, it is most practical to save memory and time by sharing common feature representations across modules.

proven to be a successful technique (Collobert and Weston, 2008) on its own; thus, in our experiments, we compare the joint decoder and using the MTL strategy alone.

3.4 Experiments

Data We perform experiments on the MWE-Aware English Dependency Corpus (Kato et al., 2017) and treebanks selected from Universal Dependencies 2.2 (UD; Nivre et al., 2018a) for having frequent occurrences of headless MWE structures. The MWE-Aware English Dependency Corpus provides automatically unified named-entity annotations based on OntoNotes 5.0 (Weischedel et al., 2013) and Stanford-style dependency trees (de Marneffe and Manning, 2008). We extract MWE spans according to mwe_NNP dependency relations. We choose the UD treebanks based on two basic properties that hold for flat structures conforming to the UD annotation guidelines: (1) all words that are attached via flat relations must be leaf nodes and (2) all words within a flat span should be attached to a common "head" word, and each arc label should be either flat or punct.[8] For each treebank, we compute its *compliance ratio*, defined as the percentage of its trees containing flat arc labels that satisfy both properties above; and we filter out those with compliance ratios below 90%.[9] We rank the remaining treebanks by their ratios of flat relations among all dependency arcs,

[8] punct inside a headless span is often used for hyphens and other internal punctuation in named entities. See the English sentence in Figure 3.2 for an example.

[9] The two properties defined in the UD guidelines for headless structures provide us with a common basis for uniform treatment across languages and treebanks. Unfortunately, the two properties can be violated quite often, due to issues in annotation and automatic treebank conversion into UD style. In 6 out of the top 10 treebanks containing the most flat relations, (at least one of) these properties are violated in more than 35% of the sentences with flat relations and have to be excluded from our experiments. We hope that ongoing community effort in data curation will facilitate evaluation on more diverse languages.

	Treebank	# tokens	# headless arcs	%	# headless spans	Average span length	Compliance ratio
	English	731,677	32,065	4.38%	16,997	2.89	100.00%
UD 2.2	de_gsd	263,804	6,786	2.57%	5,663	2.59	93.00%
	it_postwita	99,441	2,733	2.75%	2,277	2.26	94.89%
	nl_alpino	186,046	4,734	2.54%	3,269	2.45	100.00%
	nl_lassysmall	75,134	4,408	5.87%	3,018	2.46	99.82%
	no_nynorsk	245,330	5,578	2.27%	3,670	2.54	99.78%
	pt_bosque	206,739	5,375	2.60%	4,310	2.25	97.38%

Table 3.2: Dataset statistics. Language codes: de=German; it=Italian; nl=Dutch; no=Norwegian; pt=Portuguese.

and pick those with ratios higher than 2%. Six treebanks representing 5 languages, German (McDonald et al., 2013), Italian (Sanguinetti et al., 2018), Dutch (Bouma and van Noord, 2017), Norwegian (Solberg et al., 2014) and Portuguese (Rademaker et al., 2017), are selected for our experiments.[10] Data statistics are given in Table 3.2. To construct gold-standard BIO labels, we extract MWE spans according to the longest-spanning arcs that correspond to headless structures.

Implementation Details We use 3-layer bi-LSTMs where each layer has 400 dimensions in both directions and the inputs are concatenations of 100-dimensional randomly-initialized word embeddings with the final hidden vectors of 256-dimensional single-layer character-based bi-LSTMs; for BERT, we use pre-trained cased multi-lingual BERT models[11] and fine-tune the weights. We adopt the parameter settings of Dozat and Manning (2017) and use 500 and 100 dimensions for U^{att} and U_r^{rel}, respectively. The MLPs in the taggers have 500 hidden dimensions. We use a dropout (Srivastava et al., 2014) rate of 0.33, a single hidden layer, and a ReLU activation function (Nair and Hinton, 2010) for all MLPs. The models are trained with the Adam optimizer (Kingma and Ba, 2015) using a batch size of 16 sentences. The learning rates are set to 1×10^{-3} for bi-LSTMs and 1×10^{-5} for BERT initially and then multiplied by a factor of 0.1 if the performance on the development set stops improving within 3200 training iterations. For the parsing models, we use the projective Eisner (1996) decoder algorithm. For the joint training and joint decoding models, we tune $\lambda \in \{0.02, 0.05, 0.1, 0.3, 0.5, 0.9\}$ for each treebank independently and fix the set-

[10] It is a coincidence that all the selected languages are Indo-European (IE). Although there are some non-IE treebanks with high flat ratio, such as Korean (see Table 3.1), the annotated structures frequently break one or both of the basic properties. See Figure 3.2 for violation examples.

[11] `https://github.com/huggingface/transformers`

tings based on the best dev-set scores. We run each model with 5 different random seeds and report the mean and standard deviation for each setting.

Main Results We report F1 scores based on multi-word headless-structure extraction. Table 3.4 compares different strategies for identifying headless MWEs in parse trees. Tagging is consistently better than parsing except for two treebanks with the BERT feature extractor. Tagging beats parsing in all but two combinations of treebank and feature extractor. As hypothesized, our joint decoder improves over both strategies by 0.69% (1.64%) absolute through combined decisions from parsing and tagging with(out) BERT. We also compare the joint decoding setting with MTL training strategy alone. While joint decoding yields superior F1 scores, MTL is responsible for a large portion of the gains: it accounts for over half of the average gains with bi-LSTMs, and when we use pre-trained BERT feature extractors, the accuracies of jointly-trained taggers are essentially as good as joint decoding models.

Interestingly, the choice of feature extractors also has an effect on the performance gap between parsers and taggers. With bi-LSTMs, tagging is 1.42% absolute F1 higher than parsing, and the gap is mitigated through MTL. While pre-trained BERT reduces the performance difference dramatically down to 0.59% absolute, MTL no longer helps parsers overcome this gap. Additionally, we observe that MTL helps both parsing and tagging models, demonstrating that the two views of the same underlying structures are complementary to each other and that learning both can be beneficial to model training. By resolving such representational discrepancies, joint decoding exhibits further accuracy improvement.

Treebank	Our Parsers	CoNLL 2018 Best
de_gsd	**80.65**	80.36
it_ostwita	79.33	**79.39**
nl_alpino	**89.78**	89.56
nl_lassysmall	**87.96**	86.84
no_nynorsk	90.44	**90.99**
pt_bosque	**89.25**	87.81

Table 3.3: Comparison of our (non-MTL) parsing models with the best-performing systems (Che et al., 2018; Qi et al., 2018) from the CoNLL 2018 shared task, measured by labeled attachment scores (LAS, %).

	w/ bi-LSTM Treebank	Compl. Ratio ↓	Parsing	Tagging	MTL Parsing	MTL Tagging	Joint Decoding
	English	100.00	$91.24_{\pm0.60}$	$91.81_{\pm0.45}$	$93.00_{\pm0.83}$	**93.24**$_{\pm0.76}$	**93.49**$_{\pm0.43}$
UD 2.2	nl_alpino	100.00	$72.66_{\pm1.73}$	$74.94_{\pm1.00}$	$77.29_{\pm0.80}$	$75.58_{\pm1.18}$	**79.65**$_{\pm1.05}$
	nl_lassysmall	99.82	$76.44_{\pm1.56}$	**77.98**$_{\pm1.56}$	**78.13**$_{\pm0.98}$	$77.58_{\pm1.17}$	**78.92**$_{\pm1.00}$
	no_nynorsk	99.78	$85.34_{\pm0.81}$	$87.67_{\pm0.90}$	$86.72_{\pm0.76}$	$87.44_{\pm0.76}$	**88.40**$_{\pm0.39}$
	pt_bosque	97.38	$89.55_{\pm1.10}$	$90.97_{\pm0.46}$	**91.30**$_{\pm0.75}$	**92.07**$_{\pm1.04}$	$90.63_{\pm1.56}$
	it_postwita	94.89	$75.35_{\pm1.05}$	$76.37_{\pm1.72}$	**78.46**$_{\pm1.08}$	**77.87**$_{\pm0.57}$	**78.38**$_{\pm1.04}$
	de_gsd	93.00	$63.32_{\pm1.36}$	$64.10_{\pm1.31}$	**64.81**$_{\pm2.05}$	**65.07**$_{\pm1.35}$	**65.86**$_{\pm1.34}$
	Average		79.13	80.55	81.39	81.26	82.19

	w/ BERT Treebank	Compl. Ratio ↓	Parsing	Tagging	MTL Parsing	MTL Tagging	Joint Decoding
	English	100.00	$94.98_{\pm0.26}$	$95.45_{\pm0.23}$	$95.01_{\pm0.20}$	**95.86**$_{\pm0.19}$	$95.51_{\pm0.58}$
UD 2.2	nl_alpino	100.00	$83.87_{\pm1.61}$	$83.32_{\pm1.01}$	$84.65_{\pm1.48}$	**85.90**$_{\pm1.51}$	**86.61**$_{\pm1.52}$
	nl_lassysmall	99.82	$87.16_{\pm1.20}$	$87.52_{\pm0.59}$	**88.10**$_{\pm0.80}$	$87.68_{\pm0.78}$	**88.35**$_{\pm0.49}$
	no_nynorsk	99.78	$92.16_{\pm0.93}$	**93.48**$_{\pm0.48}$	$92.45_{\pm0.34}$	**93.11**$_{\pm0.21}$	**93.08**$_{\pm0.62}$
	pt_bosque	97.38	$92.98_{\pm0.82}$	$93.47_{\pm0.55}$	$93.42_{\pm0.65}$	**93.85**$_{\pm0.57}$	**94.01**$_{\pm0.19}$
	it_postwita	94.89	$80.80_{\pm1.51}$	$80.80_{\pm1.52}$	**80.90**$_{\pm1.78}$	**81.33**$_{\pm0.43}$	$80.83_{\pm1.20}$
	de_gsd	93.00	$68.21_{\pm1.43}$	**70.28**$_{\pm0.70}$	**70.04**$_{\pm1.14}$	**71.05**$_{\pm1.12}$	**70.72**$_{\pm0.90}$
	Average		85.74	86.33	86.37	86.97	87.02

Table 3.4: Flat-structure identification test-set F1 scores (%) with bi-LSTM (top) and BERT (bottom). The cell with the best result for each treebank has blue shading; results within one standard deviation of the best are bolded.

Evaluation of the Strengths of Our Parsing Models To confirm that we work with reasonable parsing models, we compare our parsers with those in the CoNLL 2018 shared task (Zeman et al., 2018). The shared task featured an end-to-end parsing task, requiring all levels of text processing including tokenization, POS tagging, morphological analysis, etc. We focus on the parsing task only, and predict syntactic trees based on sentences tokenized by the Qi et al. (2018) submission.[12] Table 3.3 shows that our parsing models are highly competitive with the current state-of-the-art. Indeed, on four out of the six treebanks we selected for their density of flat structures, our baseline models actually achieve higher labeled attachment scores (LAS) than the the top scorer did in the official shared task.

Do MTL and Joint Decoding Help Parsing Performance? In terms of dependency parsing accuracies, we confirm that our parsing-only models achieve state-of-the-art performance on the UD treebanks, but there are no significant differences in parsing results among parsing-only, MTL and jointly-decoded models (See Table 3.5). This fact can be explained by the relatively low ratios of flat relations and the already-high base performance: the room for improvement on the standard LAS metrics is quite small.

3.5 Further Related Work

Syntactic analysis in conjunction with MWE identification is an important line of research (Wehrli, 2000). The span-based representations that form the ba-

[12] We thank the shared task participants and the organizers for making system predictions available at `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2885`.

	w/ bi-LSTM Treebank	Compl. Ratio ↓	Parsing	MTL Parsing	Joint Decoding
	English	100.00	$\mathbf{89.30}_{\pm 0.41}$	$\mathbf{89.39}_{\pm 0.67}$	$\mathbf{89.77}_{\pm 0.52}$
UD 2.2	`nl_alpino`	100.00	$81.97_{\pm 1.27}$	$\mathbf{82.57}_{\pm 0.99}$	$\mathbf{82.79}_{\pm 0.77}$
	`nl_lassysmall`	99.82	$82.06_{\pm 1.30}$	$\mathbf{82.90}_{\pm 0.64}$	$81.55_{\pm 1.26}$
	`no_nynorsk`	99.78	$\mathbf{86.54}_{\pm 0.50}$	$\mathbf{86.35}_{\pm 0.37}$	$\mathbf{86.65}_{\pm 0.64}$
	`pt_bosque`	97.38	$84.29_{\pm 2.15}$	$84.48_{\pm 1.61}$	$\mathbf{85.28}_{\pm 0.25}$
	`it_postwita`	94.89	$\mathbf{77.39}_{\pm 0.69}$	$\mathbf{76.75}_{\pm 1.29}$	$76.59_{\pm 1.46}$
	`de_gsd`	93.00	$\mathbf{76.66}_{\pm 0.64}$	$\mathbf{76.35}_{\pm 0.83}$	$75.22_{\pm 1.98}$
Average			82.60	82.69	82.55

	w/ BERT Treebank	Compl. Ratio ↓	Parsing	MTL Parsing	Joint Decoding
	English	100.00	$\mathbf{93.73}_{\pm 0.24}$	$\mathbf{93.52}_{\pm 0.17}$	$93.38_{\pm 0.39}$
UD 2.2	`nl_alpino`	100.00	$\mathbf{89.82}_{\pm 0.55}$	$\mathbf{89.95}_{\pm 0.41}$	$\mathbf{89.86}_{\pm 0.59}$
	`nl_lassysmall`	99.82	$\mathbf{89.78}_{\pm 0.46}$	$\mathbf{89.76}_{\pm 0.17}$	$\mathbf{89.67}_{\pm 0.16}$
	`no_nynorsk`	99.78	$\mathbf{90.77}_{\pm 0.20}$	$\mathbf{90.98}_{\pm 0.38}$	$\mathbf{90.85}_{\pm 0.32}$
	`pt_bosque`	97.38	$\mathbf{89.78}_{\pm 0.32}$	$\mathbf{89.51}_{\pm 0.39}$	$\mathbf{89.79}_{\pm 0.39}$
	`it_postwita`	94.89	$\mathbf{81.61}_{\pm 0.32}$	$\mathbf{81.70}_{\pm 0.14}$	$81.53_{\pm 0.63}$
	`de_gsd`	93.00	$81.51_{\pm 0.23}$	$\mathbf{81.74}_{\pm 0.23}$	$\mathbf{81.52}_{\pm 0.17}$
Average			88.14	88.17	88.09

Table 3.5: Dependency-parsing labeled attachment scores (LAS, %) on the test sets with bi-LSTM (top) and BERT (bottom) feature extractors. The cell containing the best result for each treebank has blue shading; results within one standard deviation of the best are in boldface.

sis of phrase-structure trees (as opposed to dependency trees) are arguably directly compatible with headless spans. This motivates approaches using joint constituency-tree representations based on context-free grammars (Arun and Keller, 2005; Constant et al., 2013) and tree substitution grammars (Green et al., 2011, 2013). Finkel and Manning (2009) add new phrasal nodes to denote named entities, enabling statistical parsers trained on this modified representation to produce both parse trees and named entity spans simultaneously. Le Roux et al. (2014) use dual decomposition to develop a joint system that combines phrase-structure parsers and taggers for compound recognition. These approaches do not directly transfer to dependency-based representations since dependency trees do not explicitly represent phrases.

In the context of dependency parsing, Eryiğit et al. (2011) report that MWE annotations have a large impact on parsing. They find that the dependency parsers are more accurate when MWE spans are not unified into single lexical items. Similar to the phrase-structure case, Candito and Constant (2014) consider MWE identification as a side product of dependency parsing into joint representations. This parse-then-extract strategy is widely adopted (Vincze et al., 2013; Nasr et al., 2015; Simkó et al., 2017). Waszczuk et al. (2019) introduce additional parameterized scoring functions for the arc labelers and use global decoding to produce consistent structures during arc-labeling steps once unlabeled dependency parse trees are predicted. Our work additionally proposes a joint decoder that combines the scores from both parsers and taggers. Alternative approaches to graph-based joint parsing and MWE identification include transition-based (Constant and Nivre, 2016) and easy-first (Constant et al., 2016) dependency parsing. These approaches typically rely on greedy decoding, whereas our joint decoder finds the globally optimal solution through dynamic programming.

Our work only focuses on a subset of MWEs that do not have internal structures. There is substantial research interest in the broad area of MWEs (Sag et al., 2002; Constant et al., 2017) including recent releases of datasets (Schneider and Smith, 2015), editions of shared tasks (Savary et al., 2017; Ramisch et al., 2018) and workshops (Savary et al., 2018, 2019). We leave it to future work to extend the comparison and combination of taggers and dependency parsers to other MWE constructions.

3.6 Chapter Summary and Future Work

Our work provides an empirical comparison of different strategies for extracting headless MWEs from dependency parse trees: parsing, tagging, and joint modeling. Experiments on the MWE-Aware English Dependency Corpus and UD 2.2 across five languages show that tagging, a widely-used methodology for extracting spans from texts, is more accurate than parsing for this task. When using bi-LSTM (but not BERT) representations, our proposed joint decoder reaches higher F1 scores than either of the two other strategies, by combining scores of the two different and complementary representations of the same structures. We also show that most of the gains stem from a multi-task learning strategy that shares common neural representations between the parsers and the taggers.

An interesting additional use-case for our joint decoder is when a downstream task, e.g., relation extraction, requires output structures from both a parser and a tagger. Our joint decoder can find the highest-scoring consistent structures among all candidates, and thus has the potential to provide simpler model designs in downstream applications.

Our study has been limited to a few treebanks in UD partially due to large variations and inconsistencies across different treebanks. Future community efforts on a unified representation of flat structures for all languages would facilitate further research on linguistically-motivated treatments of headless structures in "headful" dependency treebanks.

Another limitation of our current work is that our joint decoder only produces projective dependency parse trees. To handle non-projectivity, one possible solution is pseudo-projective parsing (Nivre and Nilsson, 2005). We leave it to future work to design a non-projective decoder for joint parsing and headless structure extraction.

CHAPTER 4

ENHANCED GRAPHS

The previous two chapters focus on tree structures for dependency-based syntactic analysis, while in contrast, this chapter presents our contribution to the IWPT 2021 shared task on parsing into enhanced Universal Dependencies, which are connected graphs instead of trees. Our main system component is a hybrid tree-graph parser that integrates (a) predictions of spanning trees for the enhanced graphs with (b) additional graph edges not present in the spanning trees. We also adopt a finetuning strategy where we first train a language-generic parser on the concatenation of data from all available languages, and then, in a second step, finetune on each individual language separately. Additionally, we develop our own complete set of pre-processing modules relevant to the shared task, including tokenization, sentence segmentation, and multiword token expansion, based on pre-trained `XLM-R` models and our own pre-training of character-level language models. Our submission reaches a macro-average ELAS of 89.24 on the test set. It ranks top among all teams, with a margin of more than 2 absolute ELAS over the next best-performing submission, and best score on 16 out of 17 languages.

4.1 Introduction

The Universal Dependencies (UD; Nivre et al., 2016, 2020) initiative aims to provide cross-linguistically consistent annotations for dependency-based syntactic analysis, and includes a large collection of treebanks (202 for 114 languages in UD 2.8). Progress on the UD parsing problem has been steady (Zeman et al., 2017, 2018), but existing approaches mostly focus on parsing into *basic* UD trees, where bilexical dependency relations among surface words must form single-rooted trees. While these trees indeed contain rich syntactic information, the adherence to tree representations can be insufficient for certain constructions including coordination, gapping, relative clauses, and argument sharing through control and raising (Schuster and Manning, 2016).

The IWPT 2020 (Bouma et al., 2020) and 2021 (Bouma et al., 2021) shared tasks focus on parsing into *enhanced* UD format, where the representation is connected graphs, rather than rooted trees. The extension from trees to graphs allows direct treatment of a wider range of syntactic phenomena, but it also poses a research challenge: how to design parsers suitable for such enhanced UD graphs.

To address this setting, we propose to use a tree-graph hybrid parser leveraging the following key observation: since an enhanced UD graph must be connected, it must contain a spanning tree as a sub-graph. These spanning trees may differ from basic UD trees, but still allow us to use existing techniques developed for dependency parsing, including applying algorithms for finding maximum spanning trees to serve as accurate global decoders. Any additional dependency relations in the enhanced graphs not appearing in the spanning trees are then predicted on a per-edge basis. We find that this tree-graph hybrid approach results in more accurate predictions compared to a dependency graph parser that is combined with postprocessing steps to fix any graph connectivity issues.

Besides the enhanced graphs, the shared task setting poses two additional challenges. Firstly, the evaluation is on 17 languages from 4 language families, and not all the languages have large collections of annotated data: the lowest-resource language, Tamil, contains merely 400 training sentences — more than two magnitudes smaller than what is available for Czech, the language with the most annotations in the shared task. To facilitate knowledge sharing between high-resource and low-resource languages, we develop a two-stage finetuning strategy: we first train a language-generic model on the concatenation of all available training treebanks from all languages provided by the shared task, and then finetune on each language individually.

Secondly, the shared task demands parsing from raw text. This requires accurate text processing pipelines including modules for tokenization, sentence splitting, and multi-word token expansion, in addition to enhanced UD parsing. We build our own models for all these components; notably, we pretrain character-level masked language models on Wikipedia data, leading to improvements on tokenization, the first component in the text processing pipeline. Our multi-word token expanders combine the strengths of pre-trained learning-based models and rule-based approaches, and achieve robust results, especially on low-resource languages.

Our system submission integrates the aforementioned solutions to the three main challenges given by the shared task, and ranks top among all submissions, with a macro-average EULAS of 90.16 and ELAS of 89.24. Our system gives the best evaluation scores on all languages except for Arabic, and has large margins (more than 5 absolute ELAS) over the second-best systems on Tamil and Lithuanian, which are among languages with the smallest training treebanks.

4.2 TGIF: Tree-Graph Integrated-Format Parser for Enhanced UD

4.2.1 Tree and Graph Representations for Enhanced UD

The basic syntactic layer in UD is a single-rooted labeled dependency tree for each sentence, whereas the enhanced UD layer only requires that the set of dependency edges for each sentence form a connected graph. In these connected graphs, each word may have multiple parents, there may be multiple roots for a sentence, and the graphs may contain cycles, but there must exist one path from at least one of the roots to each node.[1]

Accompanying the increase in expressiveness of the enhanced UD representation is the challenge to produce structures that correctly satisfy graph-connectivity constraints during model inference. We summarize the existing solutions proposed for the previous run of the shared task at IWPT 2020 (Bouma et al., 2020) into four main categories:

- *Tree-based*: since the overlap between the enhanced UD graphs and the basic UD trees are typically significant, and any deviations tend to be localized and tied to one of several certain syntactic constructions (e.g, argument sharing in a control structure), one can repurpose tree-based parsers for producing enhanced UD graphs. This category of approaches include packing the additional edges from an enhanced graph into the basic tree

[1] Enhanced UD graphs additionally allow insertion of phonologically-empty nodes to recover elided elements in gapping constructions. This is currently beyond the scope our system and we use pre- and post-processing collapsing steps to handle empty nodes (§4.5).

(Kanerva et al., 2020) and using either rule-based or learning-based approaches to convert a basic UD tree into an enhanced UD graph (Heinecke, 2020; Dehouck et al., 2020; Attardi et al., 2020; Ek and Bernardy, 2020).[2]

- *Graph-based*: alternatively, one can directly focus on the enhanced UD graph with a semantic dependency graph parser that predicts the existence and label of each candidate dependency edge. But there is generally no guarantee that the set of predicted edges will form a connected graph, so a post-processing step is typically employed to fix any connectivity issues. This category of approaches includes the work of Wang et al. (2020), Barry et al. (2020), and Grünewald and Friedrich (2020).[3]
- *Transition-based*: Hershcovich et al. (2020) adapt a transition-based solution. Their system explicitly handles empty nodes through a specialized transition for inserting them; it relies on additional post-processing to ensure connectivity.
- *Tree-Graph Integrated*: He and Choi (2020) integrate a tree parser and a graph parser,[4] where the tree parser produces the basic UD tree, and the graph parser predicts any additional edges. During inference, all nodes are automatically connected through the tree parser, and the graph parser allows flexibility in producing graph structures.[5]

The tree-based approaches are prone to error propagation, since the predic-

[2] The same idea has also been applied to the task of conjunction propagation prediction (e.g., Grünewald et al., 2021).

[3] Barry et al.'s (2020) parsers use basic UD trees as features, but the output space is not restricted by the basic trees.

[4] He and Choi (2020) describe their combo as an "ensemble" but we prefer the term "integration" for both their method and ours (which is inspired by theirs), since the two components are not, strictly speaking, targeting same structures.

[5] The main difference from the tree-based approaches is that the search space for additional graph edges is unaffected by the predictions of basic UD trees in an integrated approach.

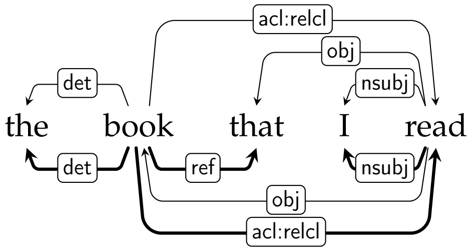


Figure 4.1: An example with basic UD and enhanced UD annotations above and below the text respectively. The extracted spanning tree (§4.2.2) is bolded and is different from the basic UD tree.

tions of the enhanced layer rely heavily on the accuracy of basic UD tree parsing. The graph-based and transition-based approaches natively produce graph structures, but they require post-processing to ensure connectivity. Our system is a tree-graph integrated-format parser that combines the strengths of the available global inference algorithms for tree parsing and the flexibility of a graph parser, without the need to use post-processing to fix connectivity issues.

4.2.2 Spanning Tree Extraction

A connected graph must contain a spanning tree, and conversely, if we first predict a spanning tree over all nodes, and subsequently add additional edges, then the resulting graph remains connected. Indeed, this property is leveraged in some previously-proposed connectivity post-processing steps (e.g., Wang et al., 2020), but extracting a spanning tree based on scores from graph-prediction models creates a mismatch between training and inference. He and Choi (2020) instead train tree parsers and graph parsers separately and combine their prediction during inference, but their tree parsers are trained on basic UD trees whose edges are not always present in the enhanced UD layer.

Our solution refines He and Choi's (2020) approach: we train tree parsers to predict spanning trees extracted from the enhanced UD graphs, instead of basic UD trees, to minimize train-test mismatch. See Figure 4.1 for an example. Spanning tree extraction is in essence assignment of unique head nodes to all nodes in a graph, subject to tree constraints. For consistent extraction, we apply the following rules:

- If a node has a unique head in the enhanced graph, there is no ambiguity in head assignment.
- If a basic UD edge is present among the set of incoming edges to a given node, include that basic UD edge in the spanning tree.
- Otherwise, there must be multiple incoming edges, none of which are present in the basic UD tree. We pick the parent node that is the "highest", i.e., the closest to the root node, in the gold-standard basic tree annotation.

The above head assignment steps do not formally guarantee that the extracted structures will be trees, but empirically, we observe that the extraction results are indeed trees for all training sentences.

4.2.3 Parameterization

Our parser architecture is adapted from that of Dozat and Manning (2017, 2018), which forms the basis for the prior graph-based approaches in the IWPT 2020 shared task. We predict unlabeled edges and labels separately, and for the unlabeled edges, we use a combination of a tree parser and a graph-edge prediction module.

Representation The first step is to extract contextual representations. For this purpose, we use the pre-trained `XLM-R` model (Conneau et al., 2020), which is trained on multilingual CommonCrawl data and supports all 17 languages in the shared task. The `XLM-R` feature extractor is finetuned along with model training. Given a length-n input sentence $x = x_1, \ldots, x_n$ and layer l, we extract

$$[\mathbf{x}_0^l, \mathbf{x}_1^l, \ldots, \mathbf{x}_n^l] = \texttt{XLM-R}^l(\texttt{<s>}, x_1, \ldots, x_n, \texttt{</s>}),$$

where inputs to the `XLM-R` model are a concatenated sequence of word pieces from each UD word, we denote the layer-l vector corresponding to the last word piece in the word x_i as $\mathbf{x}_i^l$, and the dummy root representations $\mathbf{x}_0$ are taken from the special <s> token at the beginning of the sequence.

Deep Biaffine Function All our parsing components use deep biaffine functions (DBFs), which score the interactions between pairs of words:

$$\begin{aligned} \text{DBF}(i, j) =& \mathbf{v}_i^{\text{head}\top} U \mathbf{v}_j^{\text{mod}} + \mathbf{b}^{\text{head}} \cdot \mathbf{v}_i^{\text{head}} \\ &+ \mathbf{b}^{\text{mod}} \cdot \mathbf{v}_j^{\text{mod}} + b, \end{aligned}$$

where $\mathbf{v}_i^{\text{head}}$ and $\mathbf{v}_j^{\text{mod}}$ are non-linearly transformed vectors from weighted average `XLM-R` vectors across different layers:

$$\mathbf{v}_i^{\text{head}} = \text{ReLU}\left(W^{\text{head}} \sum\nolimits_l \frac{e^{\alpha_l^{\text{head}}}}{\sum_{l'} e^{\alpha_{l'}^{\text{head}}}} \mathbf{x}_i^l \right),$$

and $\mathbf{v}_j^{\text{mod}}$ is defined similarly. Each DBF has its own trainable weight matrices U, W^{head}, and W^{mod}, vectors $\mathbf{b}^{\text{head}}$ and $\mathbf{b}^{\text{mod}}$, and scalars b, $\{\alpha_l^{\text{head}}\}$ and $\{\alpha_l^{\text{mod}}\}$.

Tree Parser To estimate the probabilities of head attachment for each token w_j, we define

$$P(\text{head}(w_j) = w_i) = \text{softmax}_i(\text{DBF}^{\text{tree}}(i, j)).$$

The tree parsing models are trained with cross-entropy loss, and we use a non-projective maximum spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967) for global inference.

Graph Parser In addition to the spanning trees, we make independent predictions on the existence of any extra edges in the enhanced UD graphs by

$$P(\exists \text{edge } w_i \rightarrow w_j) = \text{sigmoid}(\text{DBF}^{\text{graph}}(i, j)).$$

We train the graph parsing model with a cross entropy objective, and during inference, any edges with probabilities ≥ 0.5 are included in the outputs.

Relation Labeler For each edge in the unlabeled graph, we predict the relation label via

$$P(\text{lbl}(w_i \rightarrow w_j) = r) = \text{softmax}_r(\text{DBF}^{\text{rel-}r}(i, j)),$$

where we have as many deep biaffine functions as the number of candidate relation labels in the data. To reduce the large number of potential labels due to lexicalization, the relation labeler operates on a de-lexicalized version of the labels, and then a re-lexicalization step expands the predicted labels into their full forms (§4.5).

Training The above three components are separately parameterized, and during training, we optimize for the sum of their corresponding cross-entropy loss functions.

Language	Direct Training		Generic	Finetuned
	Graph+Fix	Tree-Graph		
Arabic	80.34	80.30	80.57	**80.63**
Bulgarian	91.81	92.00	91.69	**92.30**
Czech	92.93	**92.98**	92.94	**92.98**
Dutch	92.14	92.13	92.03	**92.21**
English	88.38	88.51	88.44	**88.83**
Estonian	**89.53**	89.42	89.22	89.40
Finnish	91.97	92.10	91.84	**92.48**
French	94.46	94.51	94.26	**95.52**
Italian	93.04	93.24	93.26	**93.41**
Latvian	88.47	88.42	88.38	**89.78**
Lithuanian	90.57	90.63	90.47	**90.85**
Polish	91.28	91.48	91.28	**91.63**
Russian	93.47	**93.50**	93.37	93.47
Slovak	93.70	93.83	94.00	**95.44**
Swedish	90.35	90.48	90.33	**91.57**
Tamil	66.24	66.82	67.35	**68.95**
Ukrainian	92.98	92.94	93.24	**93.89**
Average	89.51	89.61	89.57	**90.20**

Table 4.1: Dev-set ELAS (%) results, comparing graph parsers with connectivity-fixing postprocessing against tree-graph integrated models (§4.2) and comparing parsers trained directly on each language, generic-language parsers, and parsers finetuned on individual languages from the generic-language checkpoint (§4.3).

4.2.4 Empirical Comparisons

In Table 4.1, we compare our tree-graph integrated-format parser with a fully graph-based approach. The graph-based baseline uses the same feature extractor, graph parser, and relation labeler modules, but it omits the tree parser for producing spanning trees, and we apply post-processing steps to ensure connectivity of the output graphs. Our tree-graph integrated-format parser outperforms the graph-based baseline on 12 out of the 17 test languages (binomial test, $p = 0.07$).

4.3 TGIF: Two-Stage Generic- to Individual-Language Finetuning

In addition to the tree-graph integration approach, our system submission also features a two-stage finetuning strategy. We first train a language-generic model on the concatenation of all available training treebanks in the shared task data regardless of their source languages, and then finetune on each individual language in a second step.

This two-stage finetuning strategy is designed to encourage knowledge sharing across different languages, especially from high-resource languages to lower-resource ones. In our experiment results as reported in Table 4.1, we find that this strategy is indeed beneficial for the majority of languages, especially those with small training corpora (e.g., 2.13 and 1.01 absolute ELAS improvements on Tamil and French respectively), though this comes at the price of slightly decreased accuracies on high-resource languages (e.g., -0.02 on Estonian and -0.03 on Russian). Additionally, we find that the language-generic model achieves reasonably competitive performance when compared with the set of models directly trained on each individual language. This suggests that practitioners may opt to use a single model for parsing all languages if there is a need to lower disk and memory footprints, without much loss in accuracy.

4.4 Pre-TGIF: Pre-Training Grants Improvements Full-Stack

Inspired by the recent success of pre-trained language models on a wide range of NLP tasks (Peters et al., 2018; Devlin et al., 2019; Conneau et al., 2020, *inter*

Component	Pre-trained	Handwritten Rules	Lexicons
Tokenizer	✓		
Sentence splitter	✓		
MWT expander	✓	✓	✓
Lemmatizer			✓

Table 4.2: Main technologies used in our text-processing pipeline modules. The handwritten rules and lexicons are extracted from or tuned on the training data.

alia), we build our own text processing pipeline based on pre-trained language models. Due to limited time and resources, we only focus on components relevant to the shared task, which include tokenization, sentence splitting, and multi-word token (MWT) expansion. We summarize the main features of our processing pipeline in Table 4.2.

4.4.1 Tokenizers with Character-Level Masked Language Model Pre-Training

We follow state-of-the-art strategies (Qi et al., 2020; Nguyen et al., 2021) for tokenization and model the task as a tagging problem on sequences of characters. But in contrast to prior methods where tokenization and sentence segmentation are bundled into the same prediction stage, we tackle tokenization in isolation, and for each character, we make a binary prediction as to whether a token ends at the current character position or not.

An innovation in our tokenization is that we finetune character-based language models trained on Wikipedia data. In contrast, existing approaches typically use randomly-initialized models (Qi et al., 2020) or use pre-trained models on subword units instead of characters (Nguyen et al., 2021).

We follow Devlin et al. (2019) and pre-train our character-level sequence models using a masked language modeling objective: during training, we randomly replace 15% of the characters with a special mask symbol and the models are trained to predict the identity of those characters in the original texts. Due to computational resource constraints, we adopt a small-sized architecture based on simple recurrent units (Lei et al., 2018).[6] We pre-train our models on Wikipedia data[7] and each model takes roughly 2 days to complete 500k optimization steps on a single GTX 2080Ti GPU.

4.4.2 Sentence Splitters

We split texts into sentences from sequences of tokens instead of characters (Qi et al., 2020). Our approach resembles that of Nguyen et al. (2021).[8] This allows our models to condense information from a wider range of contexts while still reading the same number of input symbols. The sentence splitters are trained to make binary predictions at each token position on whether a sentence ends there. We adopt the same two-stage finetuning strategy as for our parsing modules based on pre-trained `XLM-R` feature extractors (§4.3).

[6] Simple recurrent units are a fast variant of recurrent neural networks. In our preliminary experiments, they result in lower accuracies than long-short term memory networks (LSTMs), but are 2-5 times faster, depending on sequence lengths.

[7] We extract Wikipedia texts using WikiExtractor (Attardi, 2015) from Wikipedia dumps dated 2021-04-01.

[8] An important difference is that our sentence splitters are aware of token boundaries and the models are restricted from making token-internal sentence splitting decisions.

4.4.3 Multi-Word Token (MWT) Expanders

The UD annotations distinguish between tokens and words. A word corresponds to a consecutive sequence of characters in the surface raw text and may contain one or more syntactically-functioning words (e.g., the word *Peter's* contains two words *Peter* and *'s*). We break down the MWT expansion task into first deciding whether or not to expand a given token and then performing the actual expansion. For the former, we train models to make a binary prediction on each token, and we use pre-trained `XLM-R` models as our feature extractors.

For the MWT expansion step once the tokens are identified through our classifiers, we use a combination of lexicon- and rule-based approaches. If the token form is seen in the training data, we adopt the most frequently used way to split it into multiple words. Otherwise, we invoke a set of language-specific handwritten rules developed from and tuned on the training data; a typical rule iteratively splits off an identified prefix or suffix from the remainder of the token. For example, there is a rule splitting *'s* from the token *Peter's*.

4.4.4 Lemmatizers

While the shared task requires lemmatized forms for constructing the lexicalized enhanced UD labels, we only need to predict lemmas for a small percentage of words. Empirically, these words tend to be function words and have a unique lemma per word type. Thus, we use a full lexicon-based approach to (incomplete) lemmatization. Whenever a lemma is needed during the label re-lexicalization step, we look the word up in a dictionary extracted from the training data.

Treebank	Token			Sentence			Word		
	Stanza	Trankit	Ours	Stanza	Trankit	Ours	Stanza	Trankit	Ours
Arabic-PADT	99.98	99.95	**99.99**	80.43	96.79	**96.87**	97.88	**99.39**	98.70
Bulgarian-BTB	99.93	99.78	**99.95**	97.27	98.79	**99.06**	99.93	99.78	**99.95**
Czech-FicTree	99.97	**99.98**	99.97	98.60	99.50	**99.54**	99.96	**99.98**	99.96
Czech-CAC	**99.99**	**99.99**	**99.99**	**100.00**	**100.00**	**100.00**	99.97	99.98	**99.99**
Dutch-Alpino	**99.96**	99.43	99.86	89.98	90.65	**94.45**	**99.96**	99.43	99.86
Dutch-LassySmall	99.90	99.36	**99.94**	77.95	92.60	**94.23**	99.90	99.36	**99.94**
English-EWT	**99.01**	98.67	98.79	81.13	90.49	**92.70**	**99.01**	98.67	98.79
English-GUM	**99.82**	99.52	98.88	86.35	91.60	**95.11**	**99.82**	99.52	99.18
Estonian-EDT	**99.96**	99.75	99.95	93.32	96.58	**96.60**	**99.96**	99.75	99.95
Estonian-EWT	**99.20**	97.76	98.72	67.14	82.58	**89.37**	**99.20**	97.76	98.72
Finnish-TDT	**99.77**	99.71	99.76	93.05	97.22	**98.26**	**99.73**	99.72	**99.73**
French-Sequoia	**99.90**	99.81	99.88	88.79	94.07	**96.82**	99.58	99.78	**99.84**
Italian-ISDT	**99.91**	99.88	99.90	98.76	**99.07**	**99.07**	99.76	**99.86**	99.83
Latvian-LVTB	**99.82**	99.73	99.80	99.01	98.69	**99.26**	**99.82**	99.73	99.80
Lithuanian-ALKSNIS	99.87	99.84	**99.99**	88.79	95.72	**96.22**	99.87	99.84	**99.99**
Polish-LFG	**99.95**	98.34	99.84	99.83	99.57	**99.88**	**99.95**	98.34	99.89
Polish-PDB	99.87	**99.93**	99.49	98.39	98.71	**99.66**	99.83	**99.92**	99.84
Russian-SynTagRus	99.57	99.71	**99.73**	98.86	99.45	**99.54**	99.57	99.71	**99.73**
Slovak-SNK	**99.97**	99.94	99.95	90.93	**98.49**	96.72	**99.97**	99.94	99.94
Swedish-Talbanken	**99.97**	99.91	**99.97**	98.85	99.26	**99.34**	**99.97**	99.91	**99.97**
Tamil-TTB	99.58	98.33	**99.63**	95.08	**100.00**	**100.00**	91.42	94.44	**95.34**
Ukrainian-IU	99.81	99.77	**99.86**	96.65	97.55	**98.38**	99.79	99.76	**99.84**

Table 4.3: Test-set F1 scores for tokenization, sentence segmentation, and MWT expansion, comparing Stanza (Qi et al., 2020), Trankit (Nguyen et al., 2021), and our system submission. Our system results are from the shared task official evaluations; Stanza and Trankit results are reported in the Trankit documentation with models trained on UD 2.5. *Caveat*: the results may not be strictly comparable due to treebank version mismatch.

4.4.5 Evaluation

We compare our text-processing pipeline components with two state-of-the-art toolkits, Stanza (Qi et al., 2020) and Trankit (Nguyen et al., 2021) in Table 4.3. We train our models per-language instead of per-treebank to accommodate the shared task setting, so our models are at a disadvantage when there are multiple training treebanks for a language that have different tokenization/sentence splitting conventions (e.g., English-EWT and English-GUM handle word contractions differently). Despite this, our models are highly competitive in terms of tokenization and MWT expansion, and we achieve significantly better sentence segmentation results across most treebanks. We hypothesize that a sequence-to-sequence MWT expansion approach, similar to the ones underlying Stanza and Trankit, may provide further gains to morphologically-rich languages that cannot be sufficiently modeled via handwritten rules, notably Arabic.

4.5 Other Technical Notes

Hyperparameters We report our hyperparameters in Table 4.4.

Character-level Language Model Pre-training	
Optimization:	
Optimizer	RAdam (Liu et al., 2020)
Batch size	128
Number of steps	500,000
Initial learning rate	3×10^{-4}
Weight decay	0.1
Gradient clipping	1.0
Simple Recurrent Units:	
Sequence length limit	512
Vocab size	512
Embedding size	256
Hidden size	256
Numer of layers	8
Dropout	0.3
Tokenizer	
Optimization:	
Optimizer	RAdam
Batch size	32
Initial learning rate	5×10^{-5}
Weight decay	0
Gradient clipping	1.0

Table 4.4: Hyperparameters of our submitted system. (Table continues.)

Multi-layer Perceptrons (MLPs):	
Number of layers	1
Hidden size	500
Dropout	0.5
Sentence Splitter, MWT Expander, and Parser	
Pre-trained model	`XLM-R` (Large)
Optimization:	
Optimizer	RAdam
Batch size	8
Initial learning rate	1×10^{-5}
Second-stage learning rate	1×10^{-6}
Weight decay	0
Gradient clipping	1.0
Tagger MLPs (Sentence Splitter, MWT Expander):	
Number of layers	1
Hidden size	400
Dropout	0.5
Parser MLPs (Unlabeled Tree and Graph Parsers):	
Number of layers	1
Hidden size	383
Dropout	0.33
Parser MLPs (Relation Labeler):	
Number of layers	1
Hidden size	255
Dropout	0.33

Table 4.4: Hyperparameters of our submitted system.

Empty nodes Enhanced UD graphs may contain empty nodes in addition to the words in the surface form. Our parser does not support empty nodes, so we follow the official evaluation practice and collapse relation paths with empty nodes into composite relations during training and inference.

Multiple relations In some cases, there can be multiple relations between the same pair of words. We follow Wang et al. (2020) and merge all these relations into a composite label, and re-expand them during inference.

De-lexicalization and re-lexicalization Certain types of relation labels include lexicalized information, resulting in a large relation label set. For example, nmod:in contains a lemma "in" that is taken from the modifier with a case relation. To combat this, we follow Grünewald and Friedrich's (2020) strategy and replace the lemmas[9] with placeholders consisting of their corresponding relation labels. The previous example would result in a de-lexicalized label of nmod:[case]. During inference, we apply a re-lexicalization step to reconstruct the original full relation labels given our predicted graphs. We discard the lexicalized portions of the relation labels when errors occur either in de-lexicalization (unable to locate the source child labels to match the lemmas) or re-lexicalization (unable to find corresponding placeholder relations).

Sequence length limit Pre-trained language models typically have a limit on their input sequence lengths. The XLM-R model has a limit of 512 word pieces. For a small number of sentences longer than that, we discard word-internal

[9] We find that using lemmas instead of word forms significantly improves coverage of the lexicalized labels.

word pieces, i.e., keep a prefix and a suffix of word pieces, of the longest words to fit within the limit.

Multiple Treebanks Per Language Each language in the shared task can have one or more treebanks for training and/or testing. During evaluation, there is no explicit information regarding the source treebank of the piece of input text. Instead of handpicking a training treebank for each language, we simple train and validate on the concatenation of all available data for each language.

Training on a single GPU The XLM-R model has large number of parameters, which makes it challenging to finetune on a single GPU. We use a batch size of 1 and accumulate gradients across multiple batches to lower the usage of GPU RAM. When this strategy alone is insufficient, e.g., when training the language-generic model, we additionally freeze the initial embedding layer of the model.

4.6 Official Evaluation

The shared task performs evaluation on UD treebanks that have enhanced UD annotations across 17 languages: Arabic (Hajič et al., 2009), Bulgarian (Simov et al., 2004), Czech (Hladká et al., 2010; Bejček et al., 2013; Jelínek, 2017), Dutch (van der Beek et al., 2002; Bouma and van Noord, 2017), English (Silveira et al., 2014; Zeldes, 2017), Estonian (Muischnek et al., 2014, 2019), Finnish (Haverinen et al., 2014; Pyysalo et al., 2015), French (Candito et al., 2014; Seddah and Candito, 2016), Italian (Bosco et al., 2013), Latvian (Pretkalniņa et al., 2018), Lithuanian (Bielinskienė et al., 2016), Polish (Patejuk and Przepiórkowski, 2018;

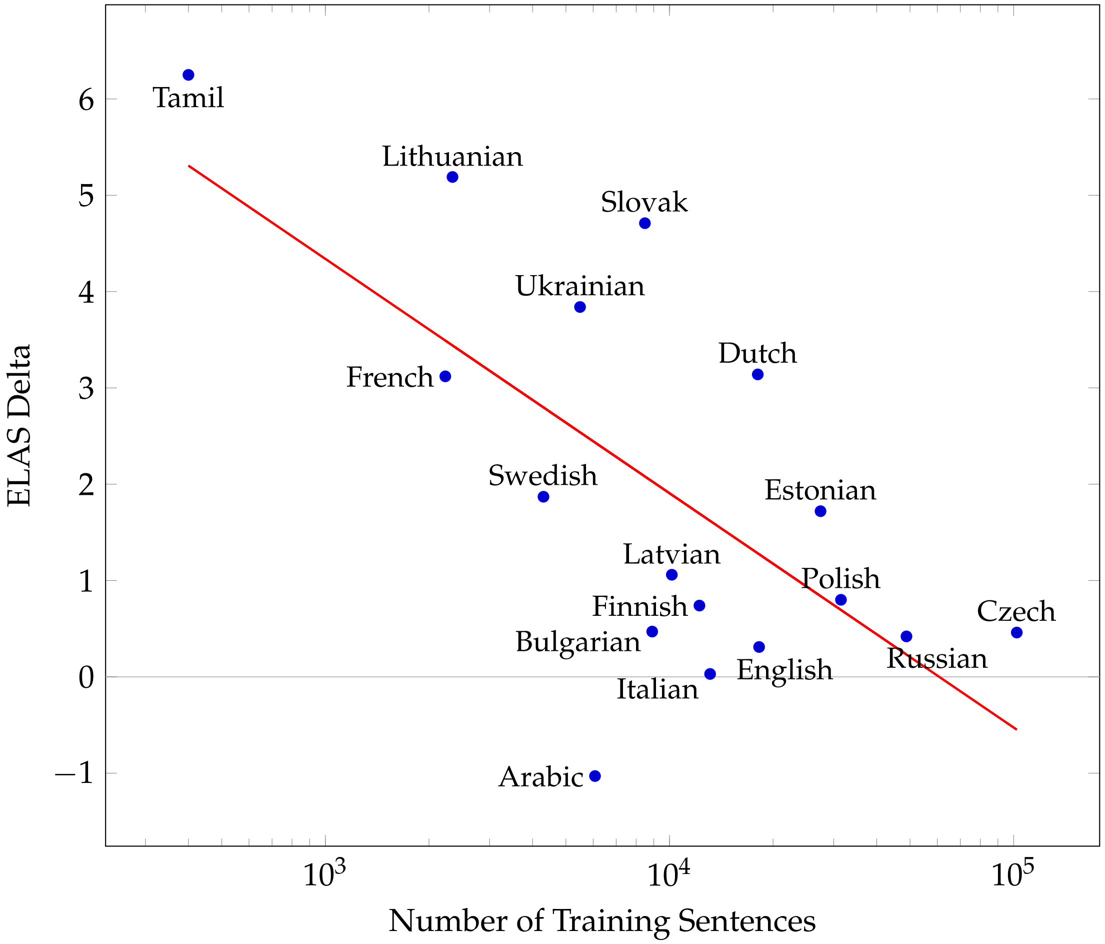


Figure 4.2: The per-language delta ELAS between our submission and the best performing system other than ours, as a function of (the log of the) number of training sentences. (For Italian, the difference is quite small but still positive.) Our models achieve larger improvements on lower-resource languages.

Language	combo	dcu_epfl	fastparse	grew	nuig	robertnlp	shanghaitech	tgif (Ours)	unipi
Arabic	76.39	71.01	53.74	71.13	–	81.58	**82.26**	81.23	77.13
Bulgarian	86.67	92.44	78.73	88.83	78.45	93.16	92.52	**93.63**	90.84
Czech	89.08	89.93	72.85	87.66	–	90.21	91.78	**92.24**	88.73
Dutch	87.07	81.89	68.89	84.09	–	88.37	88.64	**91.78**	84.14
English	84.09	85.70	73.00	85.49	65.40	87.88	87.27	**88.19**	87.11
Estonian	84.02	84.35	60.05	78.19	54.03	86.55	86.66	**88.38**	81.27
Finnish	87.28	89.02	57.71	85.20	–	91.01	90.81	**91.75**	89.62
French	87.32	86.68	73.18	83.33	–	88.51	88.40	**91.63**	87.43
Italian	90.40	92.41	78.32	90.98	–	93.28	92.88	**93.31**	91.81
Latvian	84.57	86.96	66.43	77.45	56.67	88.82	89.17	**90.23**	83.01
Lithuanian	79.75	78.04	48.27	74.62	59.13	80.76	80.87	**86.06**	71.31
Polish	87.65	89.17	71.52	78.20	–	89.78	90.66	**91.46**	88.31
Russian	90.73	92.83	78.56	90.56	66.33	92.64	93.59	**94.01**	90.90
Slovak	87.04	89.59	64.28	86.92	67.45	89.66	90.25	**94.96**	86.05
Swedish	83.20	85.20	67.26	81.54	63.12	88.03	86.62	**89.90**	84.91
Tamil	52.27	39.32	42.53	58.69	–	59.33	58.94	**65.58**	51.73
Ukrainian	86.92	86.09	63.42	83.90	–	88.86	88.94	**92.78**	87.51
Average	83.79	83.57	65.81	81.58	–	86.97	87.07	**89.24**	83.64
Rank	4	6	8	7	9	3	2	1	5

Table 4.5: Official ELAS (%) evaluation results. Our submission ranks first on 16 out of the 17 languages.

Wróblewska, 2018), Russian (Droganova et al., 2018), Slovak (Zeman, 2018), Swedish (Nivre and Megyesi, 2007), Tamil (Ramasamy and Žabokrtský, 2012), Ukrainian (Kotsyba et al., 2016), and multilingual parallel treebanks (Zeman et al., 2017).

Table 4.5 shows the official ELAS evaluation results of all 9 participating systems in the shared task.[10] Our system has the top performance on 16 out of 17 languages, and it is also the best in terms of macro-average across all languages. On average, we outperform the second best system by a margin of more than 2 ELAS points in absolute terms, or more than 15% in relative error reduction.

Figure 4.2 visualizes the "delta ELAS" between our submission and the best result other than ours on a per-language basis, plotted against the training data size for each language. Our system sees larger improvements on lower-resource languages, where we have more than 5-point leads on Tamil and Lithuanian, two languages among those with the smallest number of training sentences.

4.7 Chapter Summary and Future Work

Our submission to the IWPT 2021 shared task combines three main techniques: (1) tree-graph integrated-format parsing (graph → spanning tree → additional edges) (2) two-stage generic- to individual-language finetuning, and (3) pre-processing pipelines powered by language model pre-training. Each of the above contributes to our system performance positively,[11] and by combining

[10] Reproduced from `https://universaldependencies.org/iwpt21/results.html`.

[11] Comparing the 3 components: multilingual pre-training has a greater effect than the tree-graph parsing design. Sentence segmentation performance (SSP) doesn't necessarily translate to ELAS, so our SSP's large relative improvement at SS doesn't imply that SS is the biggest contributor to our system.

all three techniques, our system achieves the best ELAS results on 16 out of 17 languages, as well as top macro-average across all languages, among all system submissions. Additionally, our system shows more relative strengths on lower-resource languages.

Due to time and resource constraints, our system adopts the same set of techniques across all languages and we train a single set of models for our primary submission. We leave it to future work to explore language-specific methods and/or model combination and ensemble techniques to further enhance model accuracies.

CHAPTER 5

COORDINATION

This chapter focuses on a type of enriched dependency structure that is suitable for representing coordination constructions. We propose a transition-based bubble parser to perform coordination structure identification and dependency-based syntactic analysis simultaneously. Bubble representations were proposed in the formal linguistics literature decades ago; they enhance dependency trees by encoding coordination boundaries and internal relationships within coordination structures explicitly. In this work, we introduce a transition system and neural models for parsing these bubble-enhanced structures. Experimental results on the English Penn Treebank and the English GENIA corpus show that our parsers beat previous state-of-the-art approaches on the task of coordination structure prediction, especially for the subset of sentences with complex coordination structures. Our code is available at `github.com/tzshi/bubble-parser-acl21`.

5.1 Introduction

Coordination structures are prevalent in treebank data (Ficler and Goldberg, 2016a), especially in long sentences (Kurohashi and Nagao, 1994), and they are among the most challenging constructions for NLP models. Difficulties in correctly identifying coordination structures have consistently contributed to a significant portion of errors in state-of-the-art parsers (Collins, 2003; Goldberg and Elhadad, 2010; Ficler and Goldberg, 2017). These errors can further propagate to downstream NLP modules and applications, and limit their performance and utility. For example, Saha et al. (2017) report that missing conjuncts account for

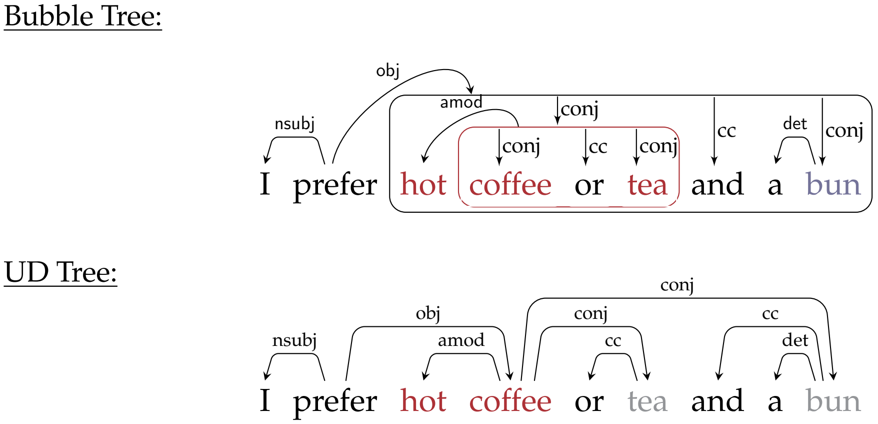


Figure 5.1: Bubble tree and (basic) UD tree for the same example sentence. (For clarity, we omit punctuation and single-word bubble boundaries.) Bubbles explicitly encode the scope of the shared modifier *"hot"* with respect to the nested coordination, whereas the UD tree gives both *"tea"* and *"bun"* identical relationships to *"hot"*.

two-thirds of the errors in recall made by their open information extraction system.

Coordination constructions are particularly challenging for the widely-adopted dependency-based paradigm of syntactic analysis, since the asymmetric definition of head-modifier dependency relations is not directly compatible with the symmetric nature of the relations among the participating conjuncts and coordinators.[1] Existing treebanks usually resort to introducing special relations to represent coordination structures. But, there remain theoretical and empirical challenges regarding how to most effectively encode information like modifier sharing relations while still permitting accurate statistical syntactic analysis.

In this work, we explore Kahane's (1997) alternative solution: extend the dependency-tree representation by introducing *bubble structures* to explicitly en-

[1] Rambow (2010) comments on other divergences between syntactic representation and syntactic phenomena.

code coordination boundaries. The co-heads within a bubble enjoy a symmetric relationship, as befits a model of conjunction. Further, bubble trees support representation of nested coordination, with the scope of shared modifiers identifiable by the attachment sites of bubble arcs. Figure 5.1 compares a bubble tree against a Universal Dependencies (UD; Nivre et al., 2016, 2020) tree for the same sentence.

Yet, despite these advantages, implementation of the formalism was not broadly pursued, for reasons unknown to us. Given its appealing and intuitive treatment of coordination phenomena, we revisit the bubble tree formalism, introducing and implementing a transition-based solution for parsing bubble trees. Our transition system, Bubble-Hybrid, extends the Arc-Hybrid transition system (Kuhlmann et al., 2011) with three bubble-specific transitions, each corresponding to opening, expanding, and closing bubbles. We show that our transition system is both sound and complete with respect to projective bubble trees (defined in §5.2.2).

Experiments on the English Penn Treebank (PTB; Marcus et al., 1993) extended with coordination annotation (Ficler and Goldberg, 2016a) and the English GENIA treebank (Kim et al., 2003) demonstrate the effectiveness of our proposed transition-based bubble parsing on the task of coordination structure prediction. Our method achieves state-of-the-art performance on both datasets and improves accuracy on the subset of sentences exhibiting complex coordination structures.

5.2 Dependency Trees and Bubble Trees

5.2.1 Dependency-based Representations for Coordination Structures

A dependency tree encodes syntactic relations via directed bilexical dependency edges. These are natural for representing argument and adjunct modification, but Popel et al. (2013) point out that *"dependency representation is at a loss when it comes to representing paratactic linguistic phenomena such as coordination, whose nature is symmetric (two or more conjuncts play the same role), as opposed to the head-modifier asymmetry of dependencies"* (pg. 517).

If one nonetheless persists in using dependency relations to annotate all syntactic structures, as is common practice in most dependency treebanks (Hajič et al., 2001; Nivre et al., 2016, *inter alia*), then one must introduce special relations to represent coordination structures and promote one element from each coordinated phrase to become the "representational head". One choice is to specify one of the conjuncts as the "head" (Mel'čuk, 1988, 2003; Järvinen and Tapanainen, 1998; Lombardo and Lesmo, 1998) (e.g., in Figure 5.1, the visually asymmetric conj relation between *"coffee"* and *"tea"* is overloaded to admit a symmetric relationship), but it is then non-trivial to distinguish shared modifiers from private ones (e.g., in the UD tree at the bottom of Figure 5.1, it is difficult to tell that *"hot"* is private to *"coffee"* and *"tea"*, which share it, but *"hot"* does not modify *"bun"*). Another choice is let one of the coordinators dominate the phrase (Hajič et al., 2001, 2020), but the coordinator does not directly capture the syntactic category of the coordinated phrase and coordinators are not obligatory in all languages and all coordination structures. Decisions on which of these dependency-based fixes is more workable are further complicated by the interaction between representation styles and their learnability in statistical parsing (Nilsson et al., 2006; Johansson and Nugues, 2007; Rehbein et al., 2017).

Enhanced UD A tactic used by many recent releases of UD treebanks is to introduce certain extra edges and non-lexical nodes (Schuster and Manning, 2016; Nivre et al., 2018b; Bouma et al., 2020). While some of the theoretical issues still persist in this approach with respect to capturing the symmetric nature of relations between conjuncts, this solution better represents shared modifiers in coordinations, and so is a promising direction. In work concurrent with our own, Grünewald et al. (2021) manually correct the coordination structure annotations in an English treebank under the enhanced UD representation format. We leave it to future work to explore the feasibility of automatic conversion of coordination structure representations between enhanced UD trees and *bubble trees*, which we discuss next.

5.2.2 Bubble Trees

An alternative solution to the coordination-in-dependency-trees dilemma is to permit certain restricted phrase-inspired constructs for such structures. Indeed, Tesnière's (1959) seminal work on dependency grammar does not describe all syntactic relations in terms of dependencies, but rather reserves a primitive relation for connecting coordinated items. Hudson (1984) further extends this idea by introducing explicit markings of coordination boundaries.

In this work, we revisit *bubble trees*, a representational device along the same vein introduced by Kahane (1997) for syntactic representation. (Kahane credits Gladkij (1968) with a formal study.) *Bubbles* are used to denote coordinated phrases; otherwise, asymmetric dependency relations are retained. Conjuncts immediately within the bubble may co-head the bubble, and the bubble itself may establish dependencies with its governor and modifiers. Figure 5.1 depicts an example bubble tree.

We now formally define bubble trees and their projective subset, which will become the focus of our transition-based parser in §5.3. The following formal descriptions are adapted from Kahane (1997), tailored to the presentation of our parser.

Formal Definition Given a dependency-relation label set L, we define a bubble tree for a length-n sentence $W = w_1, \ldots, w_n$ to be a quadruple $(V, \mathcal{B}, \phi, A)$, where $V = \{\mathsf{RT}, w_1, \ldots, w_n\}$ is the ground set of nodes (RT is the dummy root), $\mathcal{B}$ is a set of bubbles, the function $\phi : \mathcal{B} \mapsto (2^V \backslash \{\varnothing\})$ gives the content of each bubble as a non-empty[2] subset of V, and $A \subset \mathcal{B} \times L \times \mathcal{B}$ defines a labeled directed tree over $\mathcal{B}$. Given labeled directed tree A, we say $\alpha_1 \rightarrow \alpha_2$ if and only if $(\alpha_1, l, \alpha_2) \in A$ for some l. We denote the reflexive transitive closure of relation $\rightarrow$ by $\overset{*}{\rightarrow}$.

Bubble tree $(V, \mathcal{B}, \phi, A)$ is *well-formed* if and only if it satisfies the following conditions:[3]

- No partial overlap: $\forall \alpha_1, \alpha_2 \in \mathcal{B}$, either $\phi(\alpha_1) \cap \phi(\alpha_2) = \varnothing$ or $\phi(\alpha_1) \subseteq \phi(\alpha_2)$

[2] Our definition does not allow empty nodes; we leave it to future work to support them for gapping constructions.

[3] We do not use β for bubbles because we reserve the β symbol for our parser's $\underline{\text{b}}$uffer.

or $\phi(\alpha_2) \subseteq \phi(\alpha_1)$;

- Non-duplication: there exists no non-identical $\alpha_1, \alpha_2 \in \mathcal{B}$ such that $\phi(\alpha_1) = \phi(\alpha_2)$;
- Lexical coverage: for any singleton (i.e., one-element) set s in 2^V, $\exists \alpha \in \mathcal{B}$ such that $\phi(\alpha) = s$;
- Roothood: the root RT appears in exactly one bubble, a singleton that is the root of the tree defined by A.
- Containment: if $\exists \alpha_1, \alpha_2 \in \mathcal{B}$ such that $\phi(\alpha_2) \subset \phi(\alpha_1)$, then $\alpha_1 \overset{*}{\rightarrow} \alpha_2$.

Projectivity Our parser focuses on the subclass of *projective* well-formed bubble trees.[4] Visually, a projective bubble tree only contains bubbles covering a consecutive sequence of words (such that we can draw a single box (just) around the span of words to represent them) and can be drawn with all arcs arranged spatially above the sentence where no two arcs or bubble boundaries cross each other. The bubble tree in Figure 5.1 is projective.

Formally, we define the projection $\psi(\alpha) \in 2^V$ of a bubble $\alpha \in \mathcal{B}$ to be all nodes the bubble and its subtree cover, that is, $v \in \psi(\alpha)$ if and only if $\alpha \overset{*}{\rightarrow} \alpha'$ and $v \in \phi(\alpha')$ for some α'. Then, we can define a well-formed bubble tree to be *projective* if and only if it additionally satisfies the following:

- Continuous coverage: for any bubble $\alpha \in \mathcal{B}$, if $w_i, w_j \in \phi(\alpha)$ and $i < k < j$, then $w_k \in \phi(\alpha)$;

[4] While English parse trees are typically projective, some languages, such as Arabic, have high ratios of non-projective sentences. We leave it to future work to extend our parsers to non-projective scenarios (See Chapter 6).

- Continuous projections: for any bubble $\alpha \in \mathcal{B}$, if $w_i, w_j \in \psi(\alpha)$ and $i < k < j$, then $w_k \in \psi(\alpha)$;
- Contained projections: for $\alpha_1, \alpha_2 \in \mathcal{B}$, if $\alpha_1 \overset{*}{\rightarrow} \alpha_2$, then either $\psi(\alpha_2) \subset \phi(\alpha_1)$ or $\psi(\alpha_2) \cap \phi(\alpha_1) = \varnothing$.

5.3 Our Transition System for Parsing Bubble Trees

Although, as we have seen, bubble trees have theoretical benefits in representing coordination structures that interface with an overall dependency-based analysis, there has been a lack of parser implementations capable of handling such representations. In this section, we fill this gap by introducing a transition system that can incrementally build projective bubble trees.

Transition-based approaches are popular in dependency parsing (Nivre, 2008; Kübler et al., 2008). We propose to extend the Arc-Hybrid transition system (Kuhlmann et al., 2011) with transitions specific to bubble structures.[5]

5.3.1 Bubble-Hybrid Transition System

A transition system consists of a data structure describing the intermediate parser states, called *configurations*; specifications of the *initial* and *terminal configurations*; and an inventory of *transitions* that advance the parser in configuration space towards reaching a terminal configuration.

[5] Our strategy can be adapted to other transition systems as well; we focus on Arc-Hybrid here because of its comparatively small inventory of transitions, absence of spurious ambiguities (there is a one-to-one mapping between a gold tree and a valid transition sequence), and abundance of existing implementations (e.g., Kiperwasser and Goldberg, 2016).

Our transition system uses a similar configuration data structure to that of Arc-Hybrid, which consists of a stack, a buffer, and the partially-committed syntactic analysis. Initially, the stack only contains a singleton bubble corresponding to {RT}, and the buffer contains singleton bubbles, each representing a token in the sentence. Then, through taking transitions one at a time, the parser can incrementally move items from the buffer to the stack, or reduce items by attaching them to other bubbles or merging them into larger bubbles. Eventually, the parser should arrive at a terminal configuration where the stack contains the singleton bubble of {RT} again, but the buffer is empty as all the tokens are now attached to or contained in other bubbles that are now descendants of the {RT} singleton, and we can retrieve a completed bubble-tree parse.

Table 5.1 lists the available transitions in our Bubble-Hybrid system. The SHIFT, LEFTARC, and RIGHTARC transitions are as in the Arc-Hybrid system. We introduce three new transitions to handle coordination-related bubbles: BUBBLEOPEN puts the first two items on the stack into an open bubble, with the first item in the bubble, i.e., previously the second topmost item on the stack, labeled as the first conjunct of the resulting bubble; BUBBLEATTACH absorbs the topmost item on the stack into the open bubble that is at the second topmost position; and finally, BUBBLECLOSE closes the open bubble at the top of the stack and moves it to the buffer, which then allows it to take modifiers from its left through LEFTARC transitions. Figure 5.2 visualizes the stack and buffer throughout the process of parsing the example sentence in Figure 5.1. In particular, the last two steps in the left column of Figure 5.2 show the bubble corresponding to the phrase *"coffee or tea"* receiving its left modifier *"hot"* through a LEFTARC transition after it is put back on the buffer by a BUBBLECLOSE transition.

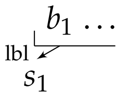
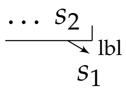
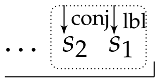
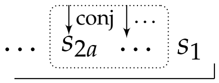
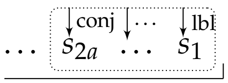
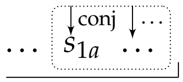
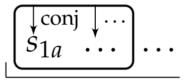
Transition	From		To	
(Pre-conditions)	Stack σ	Buffer β	Stack σ'	Buffer β'
SHIFT $(\lvert\beta\rvert \geq 1)$	$\ldots$	$b_1 \ldots$	$\ldots\ b_1$	$\ldots$
LEFTARC$_{\text{lbl}}$ $(\lvert\sigma\rvert \geq 1; \lvert\beta\rvert \geq 1; s_1, b_1 \notin \mathcal{O}; \phi(s_1) \neq \{\mathsf{RT}\})$	$\ldots\ s_1$	$b_1 \ldots$	$\ldots$	
RIGHTARC$_{\text{lbl}}$ $(\lvert\sigma\rvert \geq 2; s_1, s_2 \notin \mathcal{O})$	$\ldots\ s_2\ s_1$	$\ldots$		$\ldots$
BUBBLEOPEN$_{\text{lbl}}$ $(\lvert\sigma\rvert \geq 2; s_1, s_2 \notin \mathcal{O}; \phi(s_2) \neq \{\mathsf{RT}\})$	$\ldots\ s_2\ s_1$	$\ldots$		$\ldots$
BUBBLEATTACH$_{\text{lbl}}$ $(\lvert\sigma\rvert \geq 2; s_1 \notin \mathcal{O}; s_2 \in \mathcal{O})$	 conj $\ldots$ $\ldots$ s_{2a} $\ldots$ s_1	$\ldots$	 conj $\ldots$ lbl $\ldots$ s_{2a} $\ldots$ s_1	$\ldots$
BUBBLECLOSE $(\lvert\sigma\rvert \geq 1; s_1 \in \mathcal{O})$		$\ldots$	$\ldots$	

Table 5.1: Illustration of our Bubble-Hybrid transition system. We give the pre-conditions for each transition and visualizations of the affected stack and buffer items comparing the configurations before and after taking that transition. $\mathcal{O}$ denotes the set of currently open bubbles and RT is the dummy root symbol.

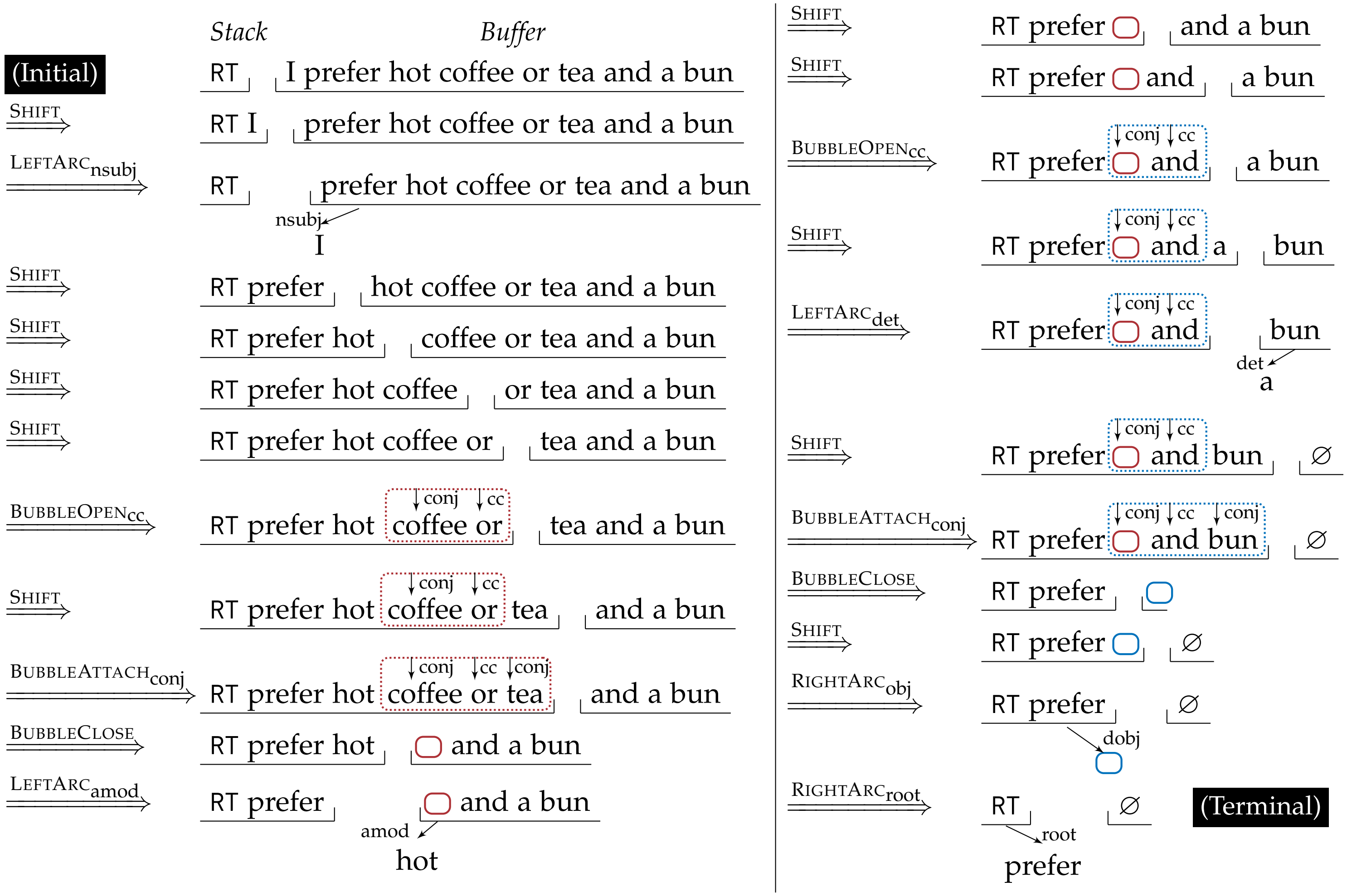


Figure 5.2: Step-by-step visualization of the stack and buffer during parsing of the example sentence in Figure 5.1. For steps following an attachment or BUBBLECLOSE transition, the detailed subtree or internal bubble structure is omitted for visual clarity. For the same reason, we omit drawing the boundaries around singleton bubbles.

Formal Definition Our transition system is a quadruple (C, T, c^i, C_τ), where C is the set of configurations to be defined shortly, T is the set of transitions with each element being a partial function $t \in T : C \rightharpoonup C$, c^i maps a sentence to its intial configuration, and $C_\tau \subset C$ is a set of terminal configurations. Each configuration $c \in C$ is a septuple $(\sigma, \beta, V, \mathcal{B}, \phi, A, \mathcal{O})$, where V, $\mathcal{B}$, ϕ, and A define a partially-recognized bubble tree, σ and β are each an (ordered) list of items in $\mathcal{B}$, and $\mathcal{O} \subset \mathcal{B}$ is a set of open bubbles. For a sentence $W = w_1, \ldots, w_n$, we let $c^i(W) = (\sigma^0, \beta^0, V, \mathcal{B}^0, \phi^0, \{\}, \{\})$, where $V = \{\mathsf{RT}, w_1, \ldots, w_n\}$, $\mathcal{B}^0$ contains $n+1$ items, $\phi^0(\mathcal{B}^0_0) = \{\mathsf{RT}\}$, $\phi^0(\mathcal{B}^0_i) = \{w_i\}$ for i from 1 to n, $\sigma^0 = [\mathcal{B}^0_0]$, and $\beta^0 = [\mathcal{B}^0_1, \ldots, \mathcal{B}^0_n]$. We write $\sigma|s_1$ and $b_1|\beta$ to denote a stack and a buffer with their topmost items being s_1 and b_1 and the remainders being σ and β respectively. We also omit the constant V in describing c when the context is clear.

For the transitions T, we have:

- $\textsc{Shift}[(\sigma, b_1|\beta, \mathcal{B}, \phi, A, \mathcal{O})] =$

  $(\sigma|b_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})$;

- $\textsc{LeftArc}_{\text{lbl}}[(\sigma|s_1, b_1|\beta, \mathcal{B}, \phi, A, \mathcal{O})] =$

  $(\sigma, b_1|\beta, \mathcal{B}, \phi, A \cup \{(b_1, \text{lbl}, s_1)\}, \mathcal{O})$;

- $\textsc{RightArc}_{\text{lbl}}[(\sigma|s_2|s_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})] =$

  $(\sigma|s_2, \beta, \mathcal{B}, \phi, A \cup \{(s_2, \text{lbl}, s_1)\}, \mathcal{O})$;

- $\textsc{BubbleOpen}_{\text{lbl}}[(\sigma|s_2|s_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})] =$

  $(\sigma|\alpha, \beta, \mathcal{B} \cup \{\alpha\}, \phi', A \cup \{(\alpha, \text{conj}, s_2), (\alpha, \text{lbl}, s_1)\}, \mathcal{O} \cup \{\alpha\})$, where α is a new bubble, and $\phi' = \phi \uplus \{\alpha \mapsto \psi(s_2) \cup \psi(s_1)\}$ (i.e., ϕ' is almost the same as ϕ, but with α added to the function's domain, mapped by the new function to cover the projections of both s_2 and s_1);

- $\textsc{BubbleAttach}_{\text{lbl}}[(\sigma|s_2|s_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})] =$

  $(\sigma|s_2, \beta, \mathcal{B}, \phi', A \cup \{s_2, \text{lbl}, s_1\}, \mathcal{O})$, where $\phi' = \phi \uplus \{s_2 \mapsto \phi(s_2) \cup \psi(s_1)\}$;
- $\textsc{BubbleClose}[(\sigma|s_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})] =$

  $(\sigma, s_1|\beta, \mathcal{B}, \phi, A, \mathcal{O}\backslash\{s_1\})$.

5.3.2 Soundness and Completeness

In this section, we show that our Bubble-Hybrid transition system is both sound and complete (defined below) with respect to the subclass of projective bubble trees.[6]

Define a *valid* transition sequence $\pi = t_1, \ldots, t_m$ for a given sentence W to be a sequence such that for the corresponding sequence of configurations $c_0, \ldots, c_m$, we have $c_0 = c^i(W)$, $c_i = t_i(c_{i-1})$, and $c_m \in C_\tau$, We can then state soundness and completeness properties, and present proof sketches below, adapted from Nivre's (2008) proof frameworks.

Lemma 1. (Soundness) *Every valid transition sequence π produces a projective bubble tree.*

Proof Sketch. We examine the requirements for a projective bubble tree one by one. The set of edges satisfies the tree constraints since every bubble except for the singleton bubble of RT must have an in-degree of one to have been reduced from the stack, and the topological order of reductions implies acyclicness. Lexical coverage is guaranteed by c^i. Roothood is safeguarded by the transition

[6] More precisely, our transition system handles the subset where each non-singleton bubble has ≥ 2 internal children.

pre-conditions. Non-duplication is ensured because newly-created bubbles are strictly larger. All the other properties can be proved by induction over the lengths of transition sequence prefixes since each of our transitions preserves zero partial overlap, containment, and projectivity constraints. □

Lemma 2. (Completeness) *For every projective bubble tree over any given sentence W, there exists a corresponding valid transition sequence π.*

Proof Sketch. The proof proceeds by strong induction on sentence length. We omit relation labels without loss of generality. The base case of $|W| = 1$ is trivial. For the inductive step, we enumerate how to decompose the tree's top-level structure. (1) When the root has multiple children: Due to projectivity, each child bubble tree τ_i covers a consecutive span of words $w_{x_i}, \ldots, w_{y_i}$ that are shorter than $|W|$. Based on the induction hypothesis, there exisits a valid transition sequence π_i to construct the child tree over $\texttt{RT}, w_{x_i}, \ldots, w_{y_i}$. Here we let π_i to denote the transition sequence excluding the always-present final RIGHTARC transition that attaches the subtree to RT; this is for explicit illustration of what transitions to take once the subtrees are constructed. The full tree can be constructed by $\pi = \pi_1$, RIGHTARC, π_2, RIGHTARC, …(expanding each π_i sequence into its component transitions), where we simply attach each subtree to RT immediately after it is constructed. (2) When the root has a single child bubble α, we cannot directly use the induction hypothesis since α covers the same number of words as W. Thus we need to further enumerate the top-level structure of α. (2a) If α has children with their projections outside of $\phi(\alpha)$, then we can find a sequence π_0 for constructing the shorter-length bubble α and placing it on the buffer (this corresponds to an empty transition sequence if α is a singleton; otherwise, π_0 ends with a BUBBLECLOSE transition) and π_is for α's outside children; say it has l children left of its contents. We construct the entire tree via $\pi = \pi_1, \ldots, \pi_l$, π_0, LEFTARC, . . . , LEFTARC, SHIFT, π_{l+1}, RIGHTARC, . . . , RIGHTARC, where we first construct all the left outside children and leave them on the stack, next build the bubble α and use LEFTARC transitions to attach its left children while it is on the buffer, then shift α to the stack before finally continuing on building its right children subtrees, each immediately followed by a RIGHTARC transition. (2b) If α is a non-singleton bubble without any outside children, but each of its inside children can be parsed through π_i based on the inductive hypothesis, then we can define $\pi = \pi_1, \pi_2$, BUBBLEOPEN, π_3, BUBBLEATTACH, . . . , BUBBLECLOSE, SHIFT, RIGHTARC, where we use a BUBBLEOPEN transition once the first two bubble-internal children are built, each subsequent child is attached via BUBBLEATTACH immediately after construction, and the final three transitions ensure proper closing of the bubble and its attachment to RT. □

5.4 Models

Our model architecture largely follows that of Kiperwasser and Goldberg's (2016) neural Arc-Hybrid parser, but we additionally introduce feature composition for non-singleton bubbles, and a rescoring module to reduce frequent coordination-boundary prediction errors. Our model has five components: feature extraction, bubble-feature composition, transition scoring, label scoring, and boundary subtree rescoring.

Feature Extraction We first extract contextualized features for each token using a bidirectional LSTM (Graves and Schmidhuber, 2005):

$$[\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_n] = \text{bi-LSTM}([\mathsf{RT}, w_1, \ldots, w_n]),$$

where the inputs to the bi-LSTM are concatenations of word embeddings, POS-tag embeddings, and character-level LSTM embeddings. We also report experiments replacing the bi-LSTM with pre-trained BERT features (Devlin et al., 2019).

Bubble-Feature Composition We initialize the features[7] for each singleton bubble $\mathcal{B}_i$ in the initial configuration to be $\mathbf{v}_{\mathcal{B}_i} = \mathbf{w}_i$. For a non-singleton bubble α, we use recursively composed features

$$\mathbf{v}_\alpha = g(\{\mathbf{v}_{\alpha'} | (\alpha, \text{conj}, \alpha') \in A\}),$$

where g is a composition function combining features from the co-heads (conjuncts) immediately inside the bubble.[8] For our model, for any $V' = \{\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_N}\}$, we set

$$g(V') = \tanh(\mathbf{W}^g \cdot \text{mean}(V')),$$

where mean() computes element-wise averages and $\mathbf{W}^g$ is a learnable square matrix. We also experiment with a parameter-free version: $g = \text{mean}$. Neither of the feature functions distinguishes between open and closed bubbles, so we append to each $\mathbf{v}$ vector an indicator-feature embedding based on whether the bubble is open, closed, or singleton.

[7] We adopt the convenient abuse of notation of allowing indexing by arbitrary objects.

[8] Comparing with the subtree-feature composition functions in dependency parsing that are motivated by asymmetric headed constructions (Dyer et al., 2015; de Lhoneux et al., 2019; Basirat and Nivre, 2021), our definition focuses on composing features from an unordered set of vectors representing the conjuncts in a bubble. The composition function is recursively applied when there are nested bubbles.

Transition Scoring Given the current parser configuration c, the model predicts the best unlabeled transition to take among all valid transitions valid(c) whose pre-conditions are satisfied. We model the log-linear probability of taking an action with a multi-layer perceptron (MLP):

$$P(t|c) \propto \exp(\mathrm{MLP}_t^{\mathrm{trans}}([\mathbf{v}_{s_3} \circ \mathbf{v}_{s_2} \circ \mathbf{v}_{s_1} \circ \mathbf{v}_{b_1}])),$$

where $\circ$ denotes vector concatenation, s_1 through s_3 are the first through third topmost items on the stack, and b_1 is the immediately accessible buffer item. We experiment with varying the number of stack items to extract features from.

Label Scoring We separate edge-label prediction from (unlabeled) transition prediction, but the scoring function takes a similar form:

$$P(l|c,t) \propto \exp(\mathrm{MLP}_l^{\mathrm{lbl}}([\mathbf{v}_{h(c,t)} \circ \mathbf{v}_{d(c,t)}])),$$

where $(h(c,t), l, d(c,t))$ is the edge to be added into the partial bubble tree in $t(c)$.

Boundary Subtree Rescoring In our preliminary error analysis, we find that our models tend to make more mistakes at the boundaries of full coordination phrases than at the internal conjunct boundaries, due to incorrect attachments of children choosing between the phrasal bubble and the first/last conjunct. For example, our initial model predicts "if you *owned it* and *liked it Friday*" instead of the annotated "if you *owned it* and *liked it* Friday" (the predicted and gold conjuncts are both italicized and underlined), incorrectly attaching *"Friday"* to *"liked"*. We attribute this problem to the greedy nature of our first formulation of the parser, and propose to mitigate the issue through rescoring. To rescore boundary attachments of a non-singleton bubble α, for each of the left dependents d of α and its first conjunct α_f, we (re)-decide the attachment via

$$P(\alpha \rightarrow d|\alpha_f) = \text{logistic}(\text{MLP}^{\text{re}}([\mathbf{v}_d \circ \mathbf{v}_\alpha \circ \mathbf{v}_{\alpha_f}])),$$

and similarly for the last conjunct α_l and a potential right dependent.

Training and Inference Our parser is a locally-trained greedy parser. In training, we optimize the model parameters to maximize the log-likelihoods of predicting the target transitions and labels along the paths generating the gold bubble trees, and the log-likelihoods of the correct attachments in rescoring;[9] during inference, the parser greedily commits to the highest-scoring transition and label for each of its current parser configurations, and after reaching a terminal configuration, it rescores and readjusts all boundary subtree attachments.

5.5 Empirical Results

Task and Evaluation We validate the utility of our transition-based parser using the task of coordination structure prediction. Given an input sentence, the task is to identify all coordination structures and the spans for all their conjuncts within that sentence. We mainly evaluate based on *exact* metrics which count a prediction of a coordination structure as correct if and only if all of its conjunct spans are correct. To facilitate comparison with pre-existing systems that do not attempt to identify all conjunct boundaries, following Teranishi et al. (2017, 2019), we also consider *inner* (=only consider the correctness of the two con-

[9] We leave the definition of dynamic oracles (Goldberg and Nivre, 2013) for bubble tree parsing to future work.

juncts adjacent to the coordinator) and *whole* (=only consider the boundary of the whole coordinated phrase) metrics.

Dataset Processing and Statistics We follow Teranishi et al. (2019) and use the same dataset splits and pre-processing steps. For the Penn Treebank (PTB; Marcus et al., 1993) data with added coordination annotations (Ficler and Goldberg, 2016a), we use WSJ sections 02-21 for training, section 22 for development, and section 23 for test sets respectively. We also use Teranishi et al.'s (2019) pre-processing steps in stripping quotation marks surrounding PTB coordinated phrases to normalize irregular coordinations. This results in 39,832/1,700/2,416 sentences and 19,890/848/1,099 coordination structures in train/dev/test splits respectively. For the GENIA treebank (Kim et al., 2003), we use the beta version of the corpus and follow the same 5-fold cross-validation splits as Teranishi et al. (2019). In total, GENIA contains 2,508 sentences and 3,598 coordination structures.

To derive bubble tree representations, we first convert the PTB-style phrase-structure trees in both treebanks with the conversion tool (Schuster and Manning, 2016) provided by the Stanford CoreNLP toolkit version 4.2.0 into Universal Dependencies (UD; Nivre et al., 2016) style.[10] We then merge the UD trees with the bubbles formed by the coordination boundaries. We define the boundaries to be from the beginning of the first conjunct to the end of the last conjunct for each coordinated phrase. We attach all conjuncts to their corresponding bubbles with a `conj` label, and map any `conj`-labeled dependencies outside an annotated coordination to `dep`. We resolve modifier scope ambiguities according to

[10] Code for data conversion is available at `https://github.com/tzshi/bubble-parser-acl21`.

conjunct annotations: if the modifier is within the span of a conjunct, then it is a private modifier; otherwise, it is a shared modifier to the entire coordinated phrase and we attach it to the phrasal bubble. Since our transition system targets projective bubble trees, we filter out any non-projective trees during training (but still evaluate on them during testing). We retain 39,678 sentences, or 99.6% of the PTB training set, and 2,429 sentences, or 96.9% of the GENIA dataset.

Implementation Details We train our models by using the Adam optimizer (Kingma and Ba, 2015). After a fixed number of optimization steps (3,200 steps for PTB and 800 steps for GENIA, based on their training set sizes), we perform an evaluation on the dev set. If the dev set performance fails to improve within 5 consecutive evaluation rounds, we multiply the learning rate by 0.1. We terminate model training when the learning rate has dropped three times, and select the best model checkpoint based on dev set F1 scores according to the "exact" metrics.[11] For the BERT feature extractor, we finetune the pretrained case-sensitive $\text{BERT}_{\text{base}}$ model through the `transformers` package.[12] For the non-BERT model, we use pre-trained GloVe embeddings (Pennington et al., 2014).

Following prior practice, we embed gold POS tags as input features when using bi-LSTM for the models trained on the GENIA dataset, but we omit the POS tag embeddings for the PTB dataset.

The training process for each model takes roughly 10 hours using an RTX 2080 Ti GPU; model inference speed is 41.9 sentences per second.[13]

[11] Even though we report recall on GENIA, model selection is still performed using F1.

[12] `https://github.com/huggingface/transformers`

[13] We have not yet done extensive optimization regarding GPU batching for greedy transition-based parsers.

Adam Optimizer:	
Initial learning rate for bi-LSTM	10^{-3}
Initial learning rate for BERT	10^{-5}
β_1	0.9
β_2	0.999
ϵ	10^{-8}
Minibatch size	8
Linear warmup steps	800
Gradient clipping L_2 norm	5.0
Inputs to bi-LSTM:	
Word-embedding dimensionality	100
POS tag-embedding dimensionality	32
Character bi-LSTM layers	1
Character bi-LSTM dimensionality	128
Bi-LSTM:	
Number of layers	3
Dimensionality	800
Dropout	0.3
MLPs (same for all MLPs):	
Number of hidden layers	1
Hidden layer dimensionality	400
Activation function	ReLU
Dropout	0.3

Table 5.2: Hyperparameters of our models.

		Dev						Test					
		All			NP			All			NP		
		P	R	F	P	R	F	P	R	F	P	R	F
Exact	TSM17	74.13	73.34	73.74	76.21	75.51	75.86	71.48	70.70	71.08	75.20	74.84	75.01
	TSM19	76.95	76.76	76.85	78.11	77.57	77.84	75.33	75.61	75.47	77.95	77.70	77.83
	Ours	77.19	77.00	77.10	80.00	79.63	79.82	76.62	76.34	76.48	81.89	81.37	81.63
	+BERT	84.25	84.55	84.40	88.53	88.33	88.43	83.85	83.62	83.74	85.33	85.19	85.26
Inner	FG16	72.34	72.25	72.29	75.17	74.82	74.99	72.81	72.61	72.70	76.91	75.31	76.10
	TSM17	76.04	75.23	75.63	77.82	77.11	77.47	74.14	73.33	73.74	77.44	77.07	77.25
	TSM19	79.19	79.00	79.10	80.64	80.09	80.36	77.60	77.88	77.74	80.19	79.93	80.06
	Ours	78.61	78.42	78.51	82.07	81.69	81.88	78.45	78.16	78.30	84.29	83.76	84.03
	+BERT	85.19	85.50	85.34	89.45	89.24	89.35	84.58	84.35	84.46	86.28	86.15	86.22

Table 5.3: Precision, recall, and F1 scores on the PTB dev and test sets.

		NP	VP	ADJP	S	PP	UCP	SBAR	ADVP	Others	All
	Count	2,317	465	321	188	167	60	56	21	3	3,598
Exact	TSM17	57.14	54.83	72.27	8.51	55.68	28.33	57.14	85.71	0.00	55.22
	TSM19	59.21	64.94	78.19	53.19	55.68	48.33	66.07	90.47	0.00	61.22
	Ours	68.28	58.71	86.29	56.38	55.09	51.67	58.93	95.24	0.00	67.09
	+BERT	79.41	76.34	88.79	77.13	73.05	61.67	76.79	100.0	33.33	79.18
Whole	HSOM09	64.2	54.2	80.4	22.9	59.9	36.7	51.8	85.7	66.7	61.5
	FG16	65.08	71.82	74.76	17.02	56.28	51.67	91.07	80.95	33.33	64.14
	TSM17	67.19	63.65	76.63	53.19	61.67	35.00	78.57	85.71	33.33	66.31
	TSM19	59.30	65.16	78.19	53.19	55.68	48.33	66.07	90.47	0.00	61.31
	Ours	69.40	59.35	87.85	57.45	56.89	51.67	62.50	95.24	0.00	68.23
	+BERT	80.58	76.77	90.03	78.19	76.65	61.67	82.14	100.0	33.33	80.41

Table 5.4: Recall on the GENIA dataset.

We select our hyperparameters by hand. Due to computational constraints, our hand-tuning has been limited to setting the dropout rates, and from the candidates set of $\{0.0, 0.1, 0.3, 0.5\}$ we chose 0.3 based on dev-set performance. Our hyperparameters are listed in Table 5.2.

Baseline Systems We compare our models with several baseline systems. Hara et al. (2009, HSOM09) use edit graphs to explicitly align coordinated conjuncts based on the idea that they are usually similar; Ficler and Goldberg (2016b, FG16) score candidate coordinations extracted from a phrase-structure parser by modeling their symmetry and replaceability properties; Teranishi et al. (2017, TSM17) directly predict boundaries of coordinated phrases and then split them into conjuncts;[14] Teranishi et al. (2019, TSM19) use separate neural models to score the inner and outer boundaries of conjuncts relative to the coordinators, and then use a chart parser to find the globally-optimal coordination structures.

Main Results Table 5.3 and Table 5.4 show the main evaluation results on the PTB and GENIA datasets. Our models surpass all prior results on both datasets. While the BERT improvements may not seem surprising, we note that Teranishi et al. (2019) report that their pre-trained language models — specifically, static ELMo embeddings — fail to improve their model performance.

General Parsing Results We also evaluate our models on standard parsing metrics by converting the predicted bubble trees to UD-style dependency trees. On PTB, our parsers reach unlabeled and labeled attachment scores (UAS/LAS)

[14] We report results for the extended model of TSM17 as described by Teranishi et al. (2019).

	Bubble-Hybrid (Ours)		Edge-Factored	
	Prec.	Rec.	Prec.	Rec.
punct	92.56	92.52	92.92	92.85
case	97.46	98.14	97.71	98.26
compound	94.12	95.18	94.24	95.02
det	98.85	99.13	98.70	99.06
nsubj	97.69	97.72	98.01	97.92
nmod	93.04	93.45	93.20	93.62
amod	94.52	93.43	94.61	93.95
...	...	...	...	...
conj	92.68	93.20	92.52	93.04
UAS	95.81		95.99	
LAS	94.46		94.56	

Table 5.5: PTB test-set results, comparing our transition-based bubble parser and an edge-factored graph-based parser, both using a BERT-based feature encoder. The relation labels are ordered by decreasing frequency. While our transition-based bubble parser slightly underperforms the graph-based dependency parser generally, perhaps due to the disadvantage of greedy decoding, it gives slightly better precision and recall on the “conj” relation type.

Rescoring	$+$	$-$
Ours (bi-LSTM)	77.10	76.27
• $g = \text{mean}$	75.51	74.16
• $\{\mathbf{v}_{s_2}, \mathbf{v}_{s_1}, \mathbf{v}_{b_1}\}$	76.05	74.87
• $\{\mathbf{v}_{s_1}, \mathbf{v}_{b_1}\}$	76.33	73.85
• $\{\mathbf{v}_{b_1}\}$	50.27	35.14
• +BERT	84.40	83.70

Table 5.6: Exact F1 scores of different model variations on the PTB dev set, w/ and w/o the rescoring module.

of 95.81/94.46 with BERT and 94.49/92.88 with bi-LSTM, which are similar to the scores of prior transition-based parsers equipped with similar feature extractors (Kiperwasser and Goldberg, 2016; Mohammadshahi and Henderson, 2020).[15] Table 5.5 compares the general parsing results of our bubble parser and an edge-factored graph-based dependency parser based on Dozat and Manning’s (2017)

[15] Results are not strictly comparable with previous PTB evaluations that mostly focus on non-UD dependency conversions. Table 5.5 makes a self-contained comparison using the same UD-based and coordination-merged data conversions.

Complexity	All	Simple	Complex
TSM17	66.09	72.90	50.37
TSM19	70.90	78.16	54.16
Ours	72.97	79.97	56.82
+BERT	80.07	83.74	71.59

Table 5.7: Per-sentence exact match on the PTB test set. *Simple* includes sentences with only one two-conjunct coordination, and *complex* contains the other cases.

parser architecture and the same feature encoder as our parser and trained on the same data. Our bubble parser shows a slight improvement on identifying the conj relations, despite having a lower overall accuracy due to the greedy nature of our transition-based decoder. Additionally, our bubble parser simultaneously predicts the boundaries of each coordinated phrase and conjunct, while a typical dependency parser cannot produce such structures.

Model Analysis Table 5.6 shows results of our models with alternative bubble-feature composition functions and varying feature-set sizes. We find that the parameterized form of composition function g performs better, and the F1 scores mostly degrade as we use fewer features from the stack. Interestingly, the importance of our rescoring module becomes more prominent when we use fewer features. Our results resonate with Shi et al.'s (2017a) findings on Arc-Hybrid that we need at least one stack item but not necessarily two. Table 5.7 shows that our model performs better than previous methods on complex sentences with multiple coordination structures and/or more than two conjuncts, especially when we use BERT as feature extractor.

5.6 Further Related Work

Coordination Structure Prediction Very early work with heuristic, non-learning-based approaches (Agarwal and Boggess, 1992; Kurohashi and Nagao, 1994) typically report difficulties in distinguishing shared modifiers from private ones, although such heuristics have been recently incorporated in unsupervised work (Sawada et al., 2020). Generally, researchers have focused on symmetry principles, seeking to align conjuncts (Kurohashi and Nagao, 1994; Shimbo and Hara, 2007; Hara et al., 2009; Hanamoto et al., 2012), since coordinated conjuncts tend to be semantically and syntactically similar (Hogan, 2007), as attested to by psycholinguistic evidence of structural parallelism (Frazier et al., 1984, 2000; Dubey et al., 2005). Ficler and Goldberg (2016a) and Teranishi et al. (2017) additionally leverage the linguistic principle of replaceability — one can typically replace a coordinated phrase with one of its conjuncts without the sentence becoming incoherent; this idea has resulted in improved open information extraction (Saha and Mausam, 2018). Using these principles may further improve our parser.

Coordination in Constituency Grammar While our work mainly focuses on enhancing dependency-based syntactic analysis with coordination structures, coordination is a well-studied topic in constituency-based syntax (Zhang, 2009), including proposals and treatments under lexical functional grammar (Kaplan and Maxwell III, 1988), tree-adjoining grammar (Sarkar and Joshi, 1996; Han and Sarkar, 2017), and combinatory categorial grammar (Steedman, 1996, 2000).

Tesnière Dependency Structure Sangati and Mazza (2009) propose a representation that is faithful to Tesnière's (1959) original framework. Similar to bubble trees, their structures include special attention to coordination structures respecting conjunct symmetry, but they also include constructs to handle other syntactic notions currently beyond our parser's scope.[16] Such representations have been used for re-ranking (Sangati, 2010), but not for (direct) parsing. Perhaps our work can inspire a future Tesnière Dependency Structure parser.

Non-constituent Coordination Seemingly incomplete (non-constituent) conjuncts are particularly challenging (Milward, 1994), and our bubble parser currently has no special mechanism for them. Dependency-based analyses have adapted by extending to a graph structure (Gerdes and Kahane, 2015) or explicitly representing elided elements (Schuster et al., 2017). It may be straightforward to integrate the latter into our parser, à la Kahane's (1997) proposal of phonologically-empty bubbles.

5.7 Chapter Summary and Future Work

We revisit Kahane's (1997) bubble tree representations for explicitly encoding coordination boundaries as a viable alternative to existing mechanisms in dependency-based analysis of coordination structures. We introduce a transition system that is both sound and complete with respect to the subclass of projective bubble trees. Empirically, our bubble parsers achieve state-of-the-art results on the task of coordination structure prediction on two English datasets.

[16] For example, differentiating content and function words which has recently been explored by Basirat and Nivre (2021).

Future work may extend the research scope to other languages, graph-based, and non-projective parsing methods.

CHAPTER 6

CONCLUDING REMARKS

This thesis makes progress towards the broad goal of (re)introducing, enforcing, and/or reinforcing structures in dependency-based syntactic analysis. These structures are described in terms of linguistic notions, such as valency patterns, and representational constraints and conventions, such as graph connectivity and flat structures in headless MWEs. Through the development of four parsers focusing on different syntactic constructions, this thesis demonstrates that structure-augmented dependency parsers are more accurate at identifying the targeted syntactic constructions than those without construction-specific emphasis during training and inference. These results have implications for future research in two avenues:

- *Evaluation:* Dependency parsers have been traditionally evaluated by summing up individual evaluations on all edge predictions, i.e., each correctly predicted dependency edge is acknowledged and each incorrect prediction is penalized independently in most standard evaluation metrics. Recently, there are proposals to evaluate parsers on a subset of relations instead of treating all dependency relations equally. For example, Nivre and Fang (2017) introduce a content-word-focused measure, and the CoNLL 2018 shared task (Zeman et al., 2018) extends the idea into multiple main evaluation metrics that additionally consider morphological tags and lemmas. However, these measures are still edge-based. In contrast, this thesis estimates parser performance on specific constructions based on several customized metrics, including valency pattern accuracy, MWE span identification F1 score, and coordination structure prediction F1 score. A more construction-focused evaluation can incentivize future parsers to recognize syntactic constructions as a whole, in addition to learning to predict each dependency edge independently.

- *Linguistic Resources:* As is the case with other supervised learning tasks, availability of annotated treebank data is crucial to parser development. This thesis utilizes existing data resources in multiple ways: Chapter 4 directly trains parsers on treebanks that are annotated with enhanced Universal Dependencies; Chapter 2 and Chapter 3 extract structures based on dependency trees, and then create training labels for the valency pattern taggers and the MWE taggers respectively; Chapter 5 merges syntactic trees with coordination structure annotations and convert them into bubble tree formats. During automatic conversion and label creation, Chapter 3 reveals inconsistent annotations, different interpretations of the same dependency relation labels, and in some cases, violations of the representational conventions across different treebanks regarding the headless MWE structures. Future annotation projects may consider requesting labels from human annotators on syntactic constructions as a whole instead of requiring the annotators to specify full details of each dependency edge independently; this may reduce annotation effort and chances for errors (Schneider et al., 2013). Further, Chapter 5 demonstrates the usefulness of bubble tree representations based on experiments using converted data concerning coordination structures. The bubble tree formalism has potential to address issues surrounding other syntactic phenomena, including disambiguating scopes of modification. Future creation of natively-annotated bubble treebanks will facilitate further investigation into the utility and adequacy of bubble trees and development of bubble tree parsers.

Additionally, syntactic structures are mostly used as intermediate representations in NLP. While this thesis focuses on parser improvements, it also invites future research on how to best incorporate the proposed enriched dependency structures into downstream tasks and NLP systems:

- *Applications:* A construction-centric representational format contains, in theory, more "regularized" structures and provides stronger specifications for heuristics-based downstream NLP modules and applications to work with. For example, the open information extraction system developed by Zhang et al. (2017a) operates on dependency trees based on handwritten rules. Conformance to valency patterns and MWE annotation conventions may relieve the burden of extra handling of any unexpected predictions and facilitate development of better information extraction systems. Another way to use syntactic parses is to integrate them into neural model architectures. For instance, the self-attention mechanism in the currently-popular Transformer models (Vaswani et al., 2017) calculates normalized attention weights between all possible pairs of tokens in the input sequence, which resembles the formulation of dependency parsing as a head selection problem (Zhang et al., 2017b) and in turn motivates the design of the multi-task learning framework of Strubell et al. (2018) to improve semantic role labelers by supervising one of the self-attention modules in a Transformer model to perform dependency parsing. It is less obvious how to integrate the enriched structures, such as bubble trees, into neural networks, which can be an important direction for future research.

Finally, the parsers presented in this thesis have some limitations, which are left for future work to address:

- *Modeling:* Firstly, the parsers presented in Chapter 2, Chapter 3, and Chapter 5 are limited to projective trees and projective bubble trees. While the targeted syntactic constructions, such as flat MWEs, are projective locally, the parsers' reliance on projective decoders limits their coverage of other syntactic phenomena on languages with non-negligible non-projectivity ratios. An extension to mildly non-projective decoder, may be feasible. For example, the deduction steps in the non-projective decoder of Gómez-Rodríguez et al. (2018) resemble those of Eisner (1996) and Eisner and Satta (1999), which are used in Chapter 2 and Chapter 3, so techniques introduced in this thesis may be adaptable to Gómez-Rodríguez et al.'s (2018) decoder. Secondly, this thesis only explores some decoding solutions, e.g., transition-based parsing for bubble trees, to validate the utility of the augmented and enhanced representations. Future work that further explores alternative decoding solutions, e.g., exact decoding for bubble trees, may lead to additional accuracy improvement. Last but not least, this thesis mostly considers each syntactic construction in isolation in parser designs. All the targeted structures do not conflict with each other in theory, but assembling them together in an unified representation in a naïve manner results in increased parsing time complexity and necessitates further research into more efficient decoding solutions and effective training regimes.

BIBLIOGRAPHY

Rajeev Agarwal and Lois Boggess. 1992. A simple but useful approach to conjunct identification. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 15–21, Newark, Delaware, USA. Association for Computational Linguistics.

Vilmos Ágel, Ludwig M. Eichinger, Hans Werner Eroms, Peter Hellwig, Hans Jürgen Heringer, and Henning Lobin, editors. 2003. *Dependency and valency: An international handbook of contemporary research*, volume 1. De Gruyter Mouton, Berlin, Germany.

Vilmos Ágel, Ludwig M. Eichinger, Hans Werner Eroms, Peter Hellwig, Hans Jürgen Heringer, and Henning Lobin, editors. 2006. *Dependency and valency: An international handbook of contemporary research*, volume 2. De Gruyter Mouton, Berlin, Germany.

Vilmos Ágel and Klaus Fischer. 2015. Dependency grammar and valency theory. In Bernd Heine and Heiko Narrog, editors, *The Oxford Handbook of Linguistic Analysis*, 2nd edition. Oxford University Press, Oxford, UK.

Wasi Uddin Ahmad, Nanyun Peng, and Kai-Wei Chang. 2021. GATE: Graph attention Transformer encoder for cross-lingual relation and event extraction. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 12462–12470, Online. AAAI Press.

Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. 2019. Syntactically supervised Transformers for faster neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1269–1281, Florence, Italy. Association for Computational Linguistics.

Hiyan Alshawi. 1996. Head automata and bilingual tiling: Translation with minimal representations (invited talk). In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 167–176, Santa Cruz, California, USA. Association for Computational Linguistics.

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2013. Using CCG categories to improve Hindi dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 604–609, Sofia, Bulgaria. Association for Computational Linguistics.

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2014. Improving dependency parsers using combinatory categorial grammar. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 159–163, Gothenburg, Sweden. Association for Computational Linguistics.

Abhishek Arun and Frank Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of French. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 306–313, Ann Arbor, Michigan, USA. Association for Computational Linguistics.

Giuseppe Attardi, Daniele Sartiano, and Maria Simi. 2020. Linear neural parsing and hybrid enhancement for enhanced Universal Dependencies. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 206–214, Online. Association for Computational Linguistics.

Giuseppppe Attardi. 2015. WikiExtractor. `https://github.com/attardi/wikiextractor`.

Michael Auli and Adam Lopez. 2011. Efficient CCG parsing: A* versus adaptive supertagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1577–1585, Portland, Oregon, USA. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.

Srinivas Bangalore, Pierre Boullier, Alexis Nasr, Owen Rambow, and Benoît Sagot. 2009. MICA: A probabilistic dependency parser based on tree insertion grammars (application note). In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 185–188, Boulder, Colorado, USA. Association for Computational Linguistics.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics.*, 25(2):237–265.

Srinivas Bangalore and Aravind K. Joshi. 2010. *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, USA.

James Barry, Joachim Wagner, and Jennifer Foster. 2020. The ADAPT enhanced dependency parser at the IWPT 2020 shared task. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 227–235, Online. Association for Computational Linguistics.

Ali Basirat and Joakim Nivre. 2021. Syntactic nuclei in dependency parsing – A multilingual exploration. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1376–1387, Online. Association for Computational Linguistics.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. 2013. Prague dependency treebank 3.0. `https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0023-1AAF-3`.

Akshar Bharati, Samar Husain, Dipti Misra, and Rajeev Sangal. 2009. Two stage constraint based hybrid approach to free word order language dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 77–80, Paris, France. Association for Computational Linguistics.

Akshar Bharati, Rajeev Sangal, and T. Papi Reddy. 2002. A constraint based parser using integer programming. In *Proceedings of the International Conference on Natural Language Processing*, pages 18–21, Mumbai, India.

Agnė Bielinskienė, Loïc Boizou, Jolanta Kovalevskaitė, and Erika Rimkutė. 2016. Lithuanian dependency treebank ALKSNIS. In *Proceedings of the 7th International Conference: Human Language Technologies — The Baltic Perspective*, pages 107–114, Riga, Latvia. IOS Press.

Anders Björkelund, Agnieszka Falenska, Xiang Yu, and Jonas Kuhn. 2017. IMS at the CoNLL 2017 UD shared task: CRFs and perceptrons meet neural networks. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 40–51, Vancouver, Canada. Association for Computational Linguistics.

Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 164–171, Sydney, Australia. Association for Computational Linguistics.

Rens Bod. 2007. Is the end of supervised parsing in sight? In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 400–407, Prague, Czech Republic. Association for Computational Linguistics.

Cristina Bosco, Simonetta Montemagni, and Maria Simi. 2013. Converting Italian treebanks: Towards an Italian Stanford dependency treebank. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 61–69, Sofia, Bulgaria. Association for Computational Linguistics.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2020. Overview of the IWPT 2020 shared task on parsing into enhanced Universal Dependencies. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 151–161, Online. Association for Computational Linguistics.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2021. From raw text to enhanced Universal Dependencies: The parsing shared task at IWPT 2021. In *Proceedings of the 17th International Conference on Parsing Technologies*, pages 146–157, Online. Association for Computational Linguistics.

Gosse Bouma and Gertjan van Noord. 2017. Increasing return on annotation investment: The automatic construction of a Universal Dependency treebank for Dutch. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies*, pages 19–26, Gothenburg, Sweden. Association for Computational Linguistics.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th Conference on Computational Linguistics – Volume 2*, pages 1198–1204, Kyoto, Japan. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, Online.

Marie Candito and Matthieu Constant. 2014. Strategies for contiguous multiword expression analysis and dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 743–753, Baltimore, Maryland, USA. Association for Computational Linguistics.

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric de la Clergerie. 2014. Deep syntax annotation of the

Sequoia French treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 2298–2305, Reykjavik, Iceland. European Languages Resources Association.

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, pages 41–48, San Francisco, California, USA. Morgan Kaufmann Publishers Inc.

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

Radek Čech, Petr Pajas, and Ján Majčutek. 2010. Full valency. Verb valency without distinguishing complements and adjuncts. *Journal of Quantitative Linguistics*, 17(4):291–302.

Wanxiang Che, Jiang Guo, Yuxuan Wang, Bo Zheng, Huaipeng Zhao, Yang Liu, Dechuan Teng, and Ting Liu. 2017. The HIT-SCIR system for end-to-end parsing of Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 52–62, Vancouver, Canada. Association for Computational Linguistics.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

John Chen. 2001. *Towards efficient statistical parsing using lexicalized grammatical information*. Ph.D. thesis, University of Delaware.

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding universal grammatical relations in multilingual BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5577, Online. Association for Computational Linguistics.

Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-based information extraction is dead! Long live rule-based information extraction systems! In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 827–832, Seattle, Washington, USA. Association for Computational Linguistics.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Wonchang Chung, Suhas Siddhesh Mhatre, Alexis Nasr, Owen Rambow, and Srinivas Bangalore. 2016. Revisiting supertagging and parsing: How to use supertags in transition-based parsing. In *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 85–92, Düsseldorf, Germany.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

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

Michael Collins and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proceedings of the 3rd Workshop on Very Large Corpora*, Cambridge, Massachusetts, USA.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167, New York City, New York, USA. Association for Computing Machinery.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 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. Association for Computational Linguistics.

Mathieu Constant, Gülşen Eryiğit, Johanna Monti, Lonneke van der Plas, Carlos Ramisch, Michael Rosner, and Amalia Todirascu. 2017. Multiword expression processing: A survey. *Computational Linguistics*, 43(4):837–892.

Matthieu Constant, Joseph Le Roux, and Anthony Sigogne. 2013. Combining compound recognition and PCFG-LA parsing with word lattices and conditional random fields. *ACM Transactions on Speech and Language Processing*, 10(3):8:1–8:24.

Matthieu Constant, Joseph Le Roux, and Nadi Tomeh. 2016. Deep lexical segmentation and syntactic parsing in the easy-first dependency framework. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1095–1101, San Diego, California, USA. Association for Computational Linguistics.

Matthieu Constant and Joakim Nivre. 2016. A transition-based system for joint lexical and syntactic analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 161–171, Berlin, Germany. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany. Association for Computational Linguistics.

James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics – Volume 1*, pages 91–98, Budapest, Hungary. Association for Computational Linguistics.

James R. Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 697–704, Sydney, Australia. Association for Computational Linguistics.

Miryam de Lhoneux, Miguel Ballesteros, and Joakim Nivre. 2019. Recursive subtree composition in LSTM-based dependency parsing. 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 1566–1576, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Marie-Catherine de Marneffe and Joakim Nivre. 2019. Dependency grammar. *Annual Review of Linguistics*, 5(1):197–218.

Mathieu Dehouck, Mark Anderson, and Carlos Gómez-Rodríguez. 2020. Efficient EUD parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 192–205, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, USA. Association for Computational Linguistics.

Kaustubh Dhole and Christopher D. Manning. 2020. Syn-QG: Syntactic and shallow semantic rules for question generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 752–765, Online. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France. OpenReview.net.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Kira Droganova, Olga Lyashevskaya, and Daniel Zeman. 2018. Data conversion and consistency of monolingual corpora: Russian UD treebanks. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories*, Oslo, Norway. Linköping University Electronic Press.

Amit Dubey, Patrick Sturt, and Frank Keller. 2005. Parallelism in coordination as an instance of syntactic priming: Evidence from corpus-based modeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 827–834, Vancouver, Canada. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 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. Association for Computational Linguistics.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communations of the ACM*, 13(2):94–102.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B(4):233–240.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345, Copenhagen, Denmark. The COLING 1996 Organizing Committee.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–464, College Park, Maryland, USA. Association for Computational Linguistics.

Adam Ek and Jean-Philippe Bernardy. 2020. How much of enhanced UD is contained in UD? In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 221–226, Online. Association for Computational Linguistics.

Gülşen Eryiğit, Tugay İlbay, and Ozan Arkan Can. 2011. Multiword expressions in statistical dependency parsing. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 45–55, Dublin, Ireland. Association for Computational Linguistics.

Agnieszka Faleńska, Anders Björkelund, Özlem Çetinoğlu, and Wolfgang Seeker. 2015. Stacking or supertagging for dependency parsing – what's the difference? In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 118–129, Bilbao, Spain. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2016a. Coordination annotation extension in the Penn tree bank. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 834–842, Berlin, Germany. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2016b. A neural network for coordination boundary prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 23–32, Austin, Texas, USA. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Improving a strong neural parser with conjunction-specific features. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 343–348, Valencia, Spain. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D. Manning. 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, USA. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning

Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland, USA. Association for Computational Linguistics.

Kilian A. Foth, Tomas By, and Wolfgang Menzel. 2006. Guiding a constraint dependency parser with supertags. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 289–296, Sydney, Australia. Association for Computational Linguistics.

Kilian A. Foth and Wolfgang Menzel. 2006. Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 321–328, Sydney, Australia. Association for Computational Linguistics.

Lyn Frazier, Alan Munn, and Charles Clifton. 2000. Processing coordinate structures. *Journal of Psycholinguistic Research*, 29(4):343–370.

Lyn Frazier, Lori Taft, Tom Roeper, Charles Clifton, and Kate Ehrlich. 1984. Parallel structure: A source of facilitation in sentence comprehension. *Memory & Cognition*, 12(5):421–430.

Dan Friedman, Jungo Kasai, R. Thomas McCoy, Robert Frank, Forrest Davis, and Owen Rambow. 2017. Linguistically rich vector representations of supertags for TAG parsing. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 122–131, Umeå, Sweden. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 30, pages 1019–1027, Barcelona, Spain.

Kim Gerdes and Sylvain Kahane. 2015. Non-constituent coordination and other coordinative constructions as dependency graphs. In *Proceedings of the Third International Conference on Dependency Linguistics*, pages 101–110, Uppsala, Sweden. Uppsala University.

Kim Gerdes, Joakim Nivre, Agata Savary, and Nathan Schneider. 2018. Working group on multiword expressions. `https://universaldependencies.org/workgroups/mwe.html`. Webpage accessed May 5, 2020.

Aleksej V. Gladkij. 1968. "On describing the syntactic structure of a sentence" (in Russian with English summary). *Computational Linguistics*, 7:21–44.

Goran Glavaš and Ivan Vulić. 2021. Is supervised syntactic parsing beneficial for language understanding tasks? An empirical investigation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3090–3104, Online. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256, Sardinia, Italy.

Yoav Goldberg and Michael Elhadad. 2010. Inspecting the structural biases of dependency parsing algorithms. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 234–242, Uppsala, Sweden. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414.

Carlos Gómez-Rodríguez, Tianze Shi, and Lillian Lee. 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. Association for Computational Linguistics.

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

Spence Green, Marie-Catherine de Marneffe, John Bauer, and Christopher D. Manning. 2011. Multiword expression identification with tree substitution grammars: A parsing *tour de force* with French. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 725–735, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Spence Green, Marie-Catherine de Marneffe, and Christopher D. Manning. 2013. Parsing models for identifying multiword expressions. *Computational Linguistics*, 39(1):195–227.

Stefan Grünewald and Annemarie Friedrich. 2020. RobertNLP at the IWPT 2020 shared task: Surprisingly simple enhanced UD parsing for English. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 245–252, Online. Association for Computational Linguistics.

Stefan Grünewald, Prisca Piccirilli, and Annemarie Friedrich. 2021. Coordinate constructions in English enhanced Universal Dependencies: Analysis and computational modeling. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 795–809, Online. Association for Computational Linguistics.

Jetic Gū, Hassan S. Shavarani, and Anoop Sarkar. 2018. Top-down tree structured decoding with syntactic connections for neural machine translation and parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 401–413, Brussels, Belgium. Association for Computational Linguistics.

Jan Hajič, Eduard Bejček, Jaroslava Hlavacova, Marie Mikulová, Milan Straka, Jan Štěpánek, and Barbora Štěpánková. 2020. Prague dependency treebank - consolidated 1.0. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5208–5218, Marseille, France. European Language Resources Association.

Jan Hajič, Barbora Vidová Hladká, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague dependency treebank 1.0 (LDC2001T10). https://catalog.ldc.upenn.edu/LDC2001T10.

Jan Hajič, Otakar Smrž, Petr Zemánek, Petr Pajas, Jan Šnaidauf, Emanuel Beška, Jakub Kracmar, and Kamila Hassanová. 2009. Prague Arabic dependency treebank 1.0. https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0001-4872-3.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and

Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18, Boulder, Colorado. Association for Computational Linguistics.

Chung-hye Han and Anoop Sarkar. 2017. Coordination in TAG without the Conjoin Operation. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 43–52, Umeå, Sweden. Association for Computational Linguistics.

Atsushi Hanamoto, Takuya Matsuzaki, and Jun'ichi Tsujii. 2012. Coordination structure analysis using dual decomposition. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 430–438, Avignon, France. Association for Computational Linguistics.

Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. 2009. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 967–975, Suntec, Singapore. Association for Computational Linguistics.

Katri Haverinen, Jenna Nyblom, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Anna Missilä, Stina Ojala, Tapio Salakoski, and Filip Ginter. 2014. Building the essential resources for Finnish: The Turku Dependency Treebank. *Language Resources and Evaluation*, 48(3):493–531.

Han He and Jinho D. Choi. 2020. Adaptation of multilingual transformer encoder for robust enhanced universal dependency parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared*

Task on Parsing into Enhanced Universal Dependencies, pages 181–191, Online. Association for Computational Linguistics.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Johannes Heinecke. 2020. Hybrid enhanced Universal Dependencies parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 174–180, Online. Association for Computational Linguistics.

Johannes Heinecke, Jürgen Kunze, Wolfgang Menzel, and Ingo Schröder. 1998. Eliminative parsing with graded constraints. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics – Volume 1*, pages 526–530, Montreal, Canada. Association for Computational Linguistics.

Gerhard Helbig and Wolfgang Schenkel. 1959. *Wörterbuch zur Valenz und Distribution deutscher Verben*. Bibliographisches Institut, Leipzig, Germany.

Thomas Herbst, David Heath, Ian F. Roe, and Dieter Götz. 2004. *A Valency Dictionary of English: A Corpus-Based Analysis of the Complementation Patterns of English Verbs, Nouns and Adjectives*, volume 40 of *Topics in English Linguistics*. De Gruyter Mouton, Berlin, Germany.

Daniel Hershcovich, Miryam de Lhoneux, Artur Kulmizev, Elham Pejhan, and Joakim Nivre. 2020. Køpsala: Transition-based graph parsing via efficient training and effective encoding. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 236–244, Online. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 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, USA. Association for Computational Linguistics.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

Barbora Vidová Hladká, Jan Hajič, Jiří Hana, Jaroslava Hlaváčová, Jiří Mírovský, and Jan Raab. 2010. The Czech academic corpus 2.0 guide. *The Prague Bulletin of Mathematical Linguistics*, 89(2008):41–96.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Deirdre Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 680–687, Prague, Czech Republic. Association for Computational Linguistics.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

Richard A. Hudson. 1984. *Word Grammar*. Blackwell, Oxford, UK.

Samar Husain, Raghu Pujitha Gade, and Rajeev Sangal. 2011. Linguistically rich graph based data driven parsing for Hindi. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 56–61, Dublin, Ireland. Association for Computational Linguistics.

Ray Jackendoff. 2008. *Construction after construction* and its theoretical challenges. *Language*, 84(1):8–28.

Miloš Jakubıček and Vojtěch Kovář. 2013. Enhancing Czech parsing with verb valency frames. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 282–293, Samos, Greece. Springer.

Timo Järvinen and Pasi Tapanainen. 1998. Towards an implementable dependency grammar. In *Processing of the Workshop on Dependency-Based Grammars*, pages 1–10, Montreal, Canada.

Tomáš Jelínek. 2017. FicTree: A manually annotated treebank of Czech fiction. In *Proceedings of the 17th Conference on Information Technologies - Applications and Theory*, pages 181–185, Martinské Hole, Slovakia.

Fan Jiang and Trevor Cohn. 2021. Incorporating syntax and semantics in coreference resolution with heterogeneous graph attention network. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1584–1591, Online. Association for Computational Linguistics.

Anders Johannsen, Dirk Hovy, and Anders Søgaard. 2015. Cross-lingual syntactic variation over age and gender. In *Proceedings of the Nineteenth Conference on*

Computational Natural Language Learning, pages 103–112, Beijing, China. Association for Computational Linguistics.

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

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages, Volume 3: Beyond Words*, pages 69–124. Springer, New York, USA.

Aravind K. Joshi and B. Srinivas. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 154–160, Kyoto, Japan. International Committee on Computational Linguistics.

Sylvain Kahane. 1997. Bubble trees and syntactic representations. In *Proceedings of the 5th Meeting of Mathematics of Language*, pages 70–76, Saarbrücken, Germany.

Sylvain Kahane, Marine Courtin, and Kim Gerdes. 2017. Multi-word annotation in syntactic treebanks – Propositions for Universal Dependencies. In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*, pages 181–189, Prague, Czech Republic.

Jenna Kanerva, Filip Ginter, and Sampo Pyysalo. 2020. Turku enhanced parser pipeline: From raw text to enhanced graphs in the IWPT 2020 shared task. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 162–173, Online. Association for Computational Linguistics.

Ronald M. Kaplan and John T. Maxwell III. 1988. Constituent coordination in lexical-functional grammar. In *Proceedings of International Conference on Computational Linguistics*, pages 303–305, Budapest, Hungary. International Committee on Computational Linguistics.

Jungo Kasai, Robert Frank, R. Thomas McCoy, Owen Rambow, and Alexis Nasr. 2017. TAG parsing with neural networks and vector representations of supertags. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722, Copenhagen, Denmark. Association for Computational Linguistics.

Jungo Kasai, Robert Frank, Pauli Xu, William Merrill, and Owen Rambow. 2018. End-to-end graph-based TAG parsing with neural networks. 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 1181–1194. Association for Computational Linguistics.

Akihiko Kato, Hiroyuki Shindo, and Yuji Matsumoto. 2017. English multiword expression-aware dependency parsing including named entities. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 427–432, Vancouver, Canada. Association for Computational Linguistics.

Jin Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA. OpenReview.net.

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

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology – Volume 1*, pages 40–47, Edmonton, Canada. Association for Computational Linguistics.

Daniël de Kok, Jianqiang Ma, Corina Dima, and Erhard Hinrichs. 2017. PP attachment: Where do we stand? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 311–317, Valencia, Spain. Association for Computational Linguistics.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Natalia Kotsyba, Bohdan Moskalevskyi, and Mykhailo Romanenko. 2016. Gold standard Universal Dependencies corpus for Ukrainian. `https://github.com/UniversalDependencies/UD_Ukrainian-IU`.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2008. *Dependency Parsing*, volume 2 of *Synthesis Lectures on Human Language Technologies*. Morgan and Claypool.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682, Portland, Oregon, USA. Association for Computational Linguistics.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street Corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 8:776–794.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine*

Learning, pages 282–289, San Francisco, California, USA. Morgan Kaufmann Publishers Inc.

Joseph Le Roux, Antoine Rozenknop, and Matthieu Constant. 2014. Syntactic parsing and compound recognition via dual decomposition: Application to French. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1875–1885, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4470–4481, Brussels, Belgium. Association for Computational Linguistics.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, San Diego, California, USA. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 990–1000, Doha, Qatar. Association for Computational Linguistics.

KyungTae Lim and Thierry Poibeau. 2017. A system for multilingual dependency parsing based on bidirectional LSTM feature representations. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 63–70, Vancouver, Canada. Association for Computational Linguistics.

Yuri Lin, Jean-Baptiste Michel, Erez Aiden Lieberman, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the Google Books Ngram Corpus. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 169–174, Jeju Island, Korea. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2019. Neural cross-lingual event detection with minimal parallel resources. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 738–748, Hong Kong, China. Association for Computational Linguistics.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *Proceedings of the Eighth International Conference on Learning Representations*, Online. OpenReview.net.

Vincenzo Lombardo and Leonardo Lesmo. 1998. Unit coordination and gapping in dependency theory. In *Processing of the Workshop on Dependency-Based Grammars*, pages 11–20, Montreal, Canada.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University. https://nlp.stanford.edu/software/dependencies_manual.pdf.

Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*, pages 31–38, Pittsburgh, Pennsylvania, USA. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York, Albany, USA.

Igor Mel'čuk. 2003. Levels of dependency in linguistic description: Concepts and problems. In Vilmos Ágel, Ludwig M. Eichinger, Hans Werner Eroms, Peter Hellwig, Hans Jürgen Heringer, and Henning Lobin, editors, *Dependency and Valency: An International Handbook of Contemporary Research*, volume 1, pages 188–229. Walter de Gruyter, Berlin, Germany.

David Milward. 1994. Non-constituent coordination: Theory and practice. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume 2, pages 935–941, Kyoto, Japan. International Committee on Computational Linguistics.

Seyed Abolghasem Mirroshandel and Alexis Nasr. 2016. Integrating selectional constraints and subcategorization frames in a dependency parser. *Computational Linguistics*, 42(1):55–90.

Seyed Abolghasem Mirroshandel, Alexis Nasr, and Joseph Le Roux. 2012. Semi-supervised dependency parsing using lexical affinities. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 777–785. Association for Computational Linguistics.

Seyed Abolghasem Mirroshandel, Alexis Nasr, and Benoît Sagot. 2013. Enforcing subcategorization constraints in a parser using sub-parses recombining. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 239–247. Association for Computational Linguistics.

Alireza Mohammadshahi and James Henderson. 2020. Graph-to-graph Transformer for transition-based dependency parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3278–3289, Online. Association for Computational Linguistics.

Kadri Muischnek, Kaili Müürisep, Tiina Puolakainen, Eleri Aedmaa, Riin Kirt, and Dage Särg. 2014. Estonian dependency treebank and its annotation scheme. In *Proceedings of the 13th Workshop on Treebanks and Linguistic Theories*, pages 285–291, Tübingen, Germany.

Kadri Muischnek, Kaili Müürisep, and Dage Särg. 2019. CG roots of UD treebank of Estonian web language. In *Proceedings of the NoDaLiDa 2019 Workshop on Constraint Grammar-Methods, Tools and Applications*, pages 23–26, Turku, Finland.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814, Haifa, Israel. Omnipress.

Alexis Nasr and Owen Rambow. 2004. Supertagging and full parsing. In *Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 56–63, Vancouver, Canada. Simon Fraser University.

Alexis Nasr, Carlos Ramisch, José Deulofeu, and André Valli. 2015. Joint dependency parsing and multiword expression tokenization. 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 1116–1126, Beijing, China. Association for Computational Linguistics.

Mitchell G. Newberry, Christopher A. Ahern, Robin Clark, and Joshua B. Plotkin. 2017. Detecting evolutionary forces in language change. *Nature*, 551(7679):223–226.

Dominick Ng and James R. Curran. 2015. Identifying cascading errors using constraints in dependency parsing. 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 1148–1158, Beijing, China. Association for Computational Linguistics.

Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. 2021. Trankit: A light-weight Transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the*

European Chapter of the Association for Computational Linguistics: System Demonstrations, pages 80–90, Online. Association for Computational Linguistics.

Jens Nilsson, Joakim Nivre, and Johan Hall. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 257–264, Sydney, Australia. Association for Computational Linguistics.

Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely lexicalized models for accurate and fast HPSG parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 155–163, Sydney, Australia. Association for Computational Linguistics.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. Technical Report MSI 05133, Växjö University, School of Mathematics and Systems Engineering.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Sandra Bellato, Kepa Bengoetxea, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit,

Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan

Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Shoval Sadde, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel

Zeman, Manying Zhang, and Hanzhi Zhu. 2018a. Universal Dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phương Lê Hồng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Joakim Nivre and Chiao-Ting Fang. 2017. Universal dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies*, pages 86–95, Gothenburg, Sweden. Association for Computational Linguistics.

Joakim Nivre, Paola Marongiu, Filip Ginter, Jenna Kanerva, Simonetta Montemagni, Sebastian Schuster, and Maria Simi. 2018b. Enhancing Universal Dependency treebanks: A case study. In *Proceedings of the Second Workshop on Universal Dependencies*, pages 102–107, Brussels, Belgium. Association for Computational Linguistics.

Joakim Nivre and Beata Megyesi. 2007. Bootstrapping a Swedish treebank using cross-corpus harmonization and annotation projection. In *Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories*, pages 97–102, Bergen, Norway.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 99–106, Ann Arbor, Michigan, USA. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th*

International Workshop on Semantic Evaluation, pages 63–72, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Hiroki Ouchi, Kevin Duh, and Yuji Matsumoto. 2014. Improving dependency parsers with supertags. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 154–158, Gothenburg, Sweden. Association for Computational Linguistics.

Hiroki Ouchi, Kevin Duh, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Transition-based dependency parsing exploiting supertags. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2059–2068.

Lilja Øvrelid and Joakim Nivre. 2007. When word order and part-of-speech tags are not enough – Swedish dependency parsing with rich linguistic features. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 447–451, Borovets, Bulgaria.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. *Semantic Role Labeling*, volume 6 of *Synthesis Lectures on Human Language Technologies*. Morgan and Claypool.

Agnieszka Patejuk and Adam Przepiórkowski. 2018. *From Lexical Functional Grammar to Enhanced Universal Dependencies: Linguistically Informed Treebanks of Polish*. Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland.

Adam Pauls and Dan Klein. 2009. K-best A* parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint*

Conference on Natural Language Processing of the AFNLP, pages 958–966, Suntec, Singapore. Association for Computational Linguistics.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 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. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. 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, USA. Association for Computational Linguistics.

Martin Popel, David Mareček, Jan Štěpánek, Daniel Zeman, and Zděněk Žabokrtský. 2013. Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 517–527, Sofia, Bulgaria. Association for Computational Linguistics.

Lauma Pretkalniņa, Laura Rituma, and Baiba Saulīte. 2018. Deriving enhanced Universal Dependencies from a hybrid dependency-constituency treebank. In *Proceedings of the 21sh International Conference on Text, Speech, and Dialogue*, pages 95–105, Brno, Czech Republic. Springer.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Sampo Pyysalo, Jenna Kanerva, Anna Missilä, Veronika Laippala, and Filip Ginter. 2015. Universal Dependencies for Finnish. In *Proceedings of the 20th Nordic Conference of Computational Linguistics*, pages 163–172, Vilnius, Lithuania. Linköping University Electronic Press.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Alexandre Rademaker, Fabricio Chalub, Livy Real, Cláudia Freitas, Eckhard Bick, and Valeria de Paiva. 2017. Universal Dependencies for Portuguese. In *Proceedings of the Fourth International Conference on Dependency Linguistics*, pages 197–206, Pisa, Italy. Linköping University Electronic Press.

Loganathan Ramasamy and Zdeněk Žabokrtský. 2012. Prague dependency style treebank for Tamil. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 1888–1894, Istanbul, Turkey. European Language Resources Association.

Owen Rambow. 2010. The simple truth about dependency and phrase structure representations: An opinion piece. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 337–340, Los Angeles, California, USA. Association for Computational Linguistics.

Owen Rambow and Aravind Joshi. 1997. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Leo Wanner, editor, *Recent Trends in Meaning-Text Theory*, volume 39 of *Studies in Language Companion Series*, pages 167–190. John Benjamins, Philadelphia, USA.

Carlos Ramisch, Silvio Ricardo Cordeiro, Agata Savary, Veronika Vincze, Verginica Barbu Mititelu, Archna Bhatia, Maja Buljan, Marie Candito, Polona Gantar, Voula Giouli, Tunga Güngör, Abdelati Hawwari, Uxoa Iñurrieta, Jolanta Kovalevskaitė, Simon Krek, Timm Lichte, Chaya Liebeskind, Johanna Monti, Carla Parra Escartín, Behrang QasemiZadeh, Renata Ramisch, Nathan Schneider, Ivelina Stoyanova, Ashwini Vaidya, and Abigail Walsh. 2018. Edition 1.1 of the PARSEME shared task on automatic identification of verbal multiword expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions*, pages 222–240, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Cambridge, Massachusetts.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical*

Methods in Natural Language Processing, pages 1–10, Providence, Rhode Island, USA.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of Adam and beyond. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada. OpenReview.net.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark. Association for Computational Linguistics.

Ines Rehbein, Julius Steen, Bich-Ngoc Do, and Anette Frank. 2017. Universal Dependencies are hard to parse — or are they? In *Proceedings of the Fourth International Conference on Dependency Linguistics*, pages 218–228, Pisa, Italy. Linköping University Electronic Press.

Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15, Mexico City, Mexico. Springer.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL Main Conference Poster Sessions*, pages 691–698, Sydney, Australia. Association for Computational Linguistics.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050, Prague, Czech Republic. Association for Computational Linguistics.

Swarnadeep Saha and Mausam. 2018. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for numerical Open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323, Vancouver, Canada. Association for Computational Linguistics.

Federico Sangati. 2010. A probabilistic generative model for an intermediate constituency-dependency representation. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 19–24, Uppsala, Sweden. Association for Computational Linguistics.

Federico Sangati and Chiara Mazza. 2009. An English dependency treebank à la Tesnière. In *Proceedings of the 8th International Workshop on Treebanks and Linguistic Theories*, pages 173–184, Milan, Italy.

Manuela Sanguinetti, Cristina Bosco, Alberto Lavelli, Alessandro Mazzei, Oronzo Antonelli, and Fabio Tamburini. 2018. PoSTWITA-UD: An Italian Twitter treebank in Universal Dependencies. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, pages 1768–1775, Miyazaki, Japan. European Language Resources Association.

Anoop Sarkar and Aravind Joshi. 1996. Coordination in tree adjoining grammars: Formalization and implementation. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 610–615, Copenhagen, Denmark. International Committee on Computational Linguistics.

Agata Savary, Carla Parra Escartín, Francis Bond, Jelena Mitrović, and Verginica Barbu Mititelu, editors. 2019. *Proceedings of the Joint Workshop on Multiword Expressions and WordNet)*. Association for Computational Linguistics, Florence, Italy.

Agata Savary, Carlos Ramisch, Silvio Cordeiro, Federico Sangati, Veronika Vincze, Behrang QasemiZadeh, Marie Candito, Fabienne Cap, Voula Giouli, Ivelina Stoyanova, and Antoine Doucet. 2017. The PARSEME shared task on automatic identification of verbal multiword expressions. In *Proceedings of the 13th Workshop on Multiword Expressions*, pages 31–47, Valencia, Spain. Association for Computational Linguistics.

Agata Savary, Carlos Ramisch, Jena D. Hwang, Nathan Schneider, Melanie Andresen, Sameer Pradhan, and Miriam R. L. Petruck, editors. 2018. *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions*. Association for Computational Linguistics, Santa Fe, New Mexico, USA.

Yuya Sawada, Takashi Wada, Takayoshi Shibahara, Hiroki Teranishi, Shuhei Kondo, Hiroyuki Shindo, Taro Watanabe, and Yuji Matsumoto. 2020. Coordination boundary identification without labeled data for compound terms disambiguation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3043–3049, Online. International Committee on Computational Linguistics.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A. Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under)specifying dependency syntax without overloading annotators. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 51–60, Sofia, Bulgaria. Association for Computational Linguistics.

Nathan Schneider and Noah A. Smith. 2015. A corpus and model integrating multiword expressions and supersenses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1537–1547, Denver, Colorado, USA. Association for Computational Linguistics.

Sebastian Schuster, Matthew Lamm, and Christopher D. Manning. 2017. Gapping constructions in Universal Dependencies v2. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies)*, pages 123–132, Gothenburg, Sweden. Association for Computational Linguistics.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association.

Roy Schwartz, Omri Abend, and Ari Rappoport. 2012. Learnability-based syntactic annotation design. In *Proceedings of COLING 2012, the 24th Interna-*

tional Conference on Computational Linguistics: Technical Papers, pages 2405–2422, Mumbai, India. The COLING 2012 Organizing Committee.

Djamé Seddah and Marie Candito. 2016. Hard time parsing questions: Building a QuestionBank for French. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, pages 2366–2370, Portorož, Slovenia. European Language Resources Association.

Yikang Shen, Shawn Tan, Alessandro Sordoni, Siva Reddy, and Aaron Courville. 2021. Explicitly modeling syntax in language models with incremental parsing and a dynamic oracle. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1660–1672, Online. Association for Computational Linguistics.

Tianze Shi, Liang Huang, and Lillian Lee. 2017a. 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. Association for Computational Linguistics.

Tianze Shi and Lillian Lee. 2018. Valency-augmented dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1291, Brussels, Belgium. Association for Computational Linguistics.

Tianze Shi and Lillian Lee. 2020. Extracting headless MWEs from dependency parse trees: Parsing, tagging, and joint modeling approaches. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8780–8794, Online. Association for Computational Linguistics.

Tianze Shi and Lillian Lee. 2021a. TGIF: Tree-graph integrated-format parser for enhanced UD with two-stage generic- to individual-language finetuning. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies*, Online. Association for Computational Linguistics.

Tianze Shi and Lillian Lee. 2021b. Transition-based bubble parsing: Improvements on coordination structure prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Tianze Shi, Igor Malioutov, and Ozan İrsoy. 2020. Semantic role labeling as syntactic dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7551–7571, Online. Association for Computational Linguistics.

Tianze Shi, Felix G. Wu, Xilun Chen, and Yao Cheng. 2017b. Combining global models for parsing Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 31–39, Vancouver, Canada. Association for Computational Linguistics.

Masashi Shimbo and Kazuo Hara. 2007. A discriminative learning model for coordinate conjunctions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 610–619, Prague, Czech Republic. Association for Computational Linguistics.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International*

Conference on Language Resources and Evaluation, pages 2897–2904, Reykjavik, Iceland. European Languages Resources Association.

Katalin Ilona Simkó, Viktória Kovács, and Veronika Vincze. 2017. USzeged: Identifying verbal multiword expressions with POS tagging and parsing techniques. In *Proceedings of the 13th Workshop on Multiword Expressions*, pages 48–53, Valencia, Spain. Association for Computational Linguistics.

Kiril Simov, Petya Osenova, Alexander Simov, and Milen Kouylekov. 2004. Design and implementation of the Bulgarian HPSG-based treebank. *Research on Language and Computation*, 2(4):495–522.

Per Erik Solberg, Arne Skjærholt, Lilja Øvrelid, Kristin Hagen, and Janne Bondi Johannessen. 2014. The Norwegian dependency treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 789–795, Reykjavik, Iceland. European Language Resources Association.

Kaiqiang Song, Lin Zhao, and Fei Liu. 2018. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1717–1729, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Mark Steedman. 1996. *Surface Structure and Interpretation*. Number 30 in Linguistic Inquiry Monograph. The MIT Press, Cambridge, USA.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, USA.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, UK. The COLING 2008 Organizing Committee.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782, Brussels, Belgium. Association for Computational Linguistics.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Coordination boundary identification with similarity and replaceability. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume*

1: Long Papers), pages 264–272, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2019. Decomposed local models for coordinate structure parsing. 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 3394–3403, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Lucien Tesnière. 1959. *Eléments de Syntaxe Structurale*. Librairie C. Klincksieck, Paris, France.

Damon Tutunjian and Julie E. Boland. 2008. Do we need a distinction between arguments and adjuncts? Evidence from psycholinguistic studies of comprehension. *Language and Linguistics Compass*, 2(4):631–646.

Marco A. Valenzuela-Escárcega, Gus Hahn-Powell, and Dane Bell. 2020. Odinson: A fast rule-based information extraction framework. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2183–2191, Marseille, France. European Language Resources Association.

Marco A. Valenzuela-Escárcega, Gus Hahn-Powell, Mihai Surdeanu, and Thomas Hicks. 2015. A domain-independent rule-based framework for event extraction. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 127–132, Beijing, China. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.

Leonoor van der Beek, Gosse Bouma, Rob Malouf, and Gertjan Van Noord. 2002.

The Alpino dependency treebank. In *Proceedings of Computational Linguistics in the Netherlands*, pages 8–22, Twente, Netherlands. Rodopi.

Ashish Vaswani and Kenji Sagae. 2016. Efficient structured inference for transition-based parsing with neural networks and error states. *Transactions of the Association for Computational Linguistics*, 4:183–196.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, Long Beach, California, USA. Curran Associates, Inc.

Veronika Vincze, János Zsibrita, and István Nagy T. 2013. Dependency parsing for identifying Hungarian light verb constructions. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 207–215, Nagoya, Japan. Asian Federation of Natural Language Processing.

Wen Wang and Mary P. Harper. 2004. A statistical constraint dependency grammar (CDG) parser. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 42–49, Barcelona, Spain. Association for Computational Linguistics.

Xinyu Wang, Yong Jiang, and Kewei Tu. 2020. Enhanced Universal Dependency parsing with second-order inference and mixture of training data. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 215–220, Online. Association for Computational Linguistics.

Jakub Waszczuk, Rafael Ehren, Regina Stodden, and Laura Kallmeyer. 2019. A neural graph-based approach to verbal MWE identification. In *Proceedings* *of the Joint Workshop on Multiword Expressions and WordNet*, pages 114–124, Florence, Italy. Association for Computational Linguistics.

Jakub Waszczuk, Agata Savary, and Yannick Parmentier. 2016. Promoting multiword expressions in A* TAG parsing. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 429–439, Osaka, Japan. The COLING 2016 Organizing Committee.

Jakub Waszczuk, Agata Savary, and Yannick Parmentier. 2017. Multiword expression-aware A* TAG parsing revisited. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 84–93, Umeå, Sweden. Association for Computational Linguistics.

Eric Wehrli. 2000. Parsing and collocations. In *Proceedings of the Second International Conference on Natural Language Processing*, pages 272–282, London, UK. Springer-Verlag.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. Ontonotes release 5.0 LDC2013T19. `https://catalog.ldc.upenn.edu/LDC2013T19`.

Alina Wróblewska. 2018. Extended and enhanced Polish dependency bank in Universal Dependencies format. In *Proceedings of the Second Workshop on Universal Dependencies*, pages 173–182, Brussels, Belgium. Association for Computational Linguistics.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 277–287, Vancouver, Canada. Association for Computational Linguistics.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Daniel Zeman. 2018. Slovak dependency treebank in Universal Dependencies. *Jazykovedný casopis/Journal of Linguistics*, 68(2):385–395.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Meishan Zhang, Zhenghua Li, Guohong Fu, and Min Zhang. 2019a. Syntax-enhanced neural machine translation with syntax-aware 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 1151–1161, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Niina Ning Zhang. 2009. *Coordination in Syntax*. Cambridge Studies in Linguistics. Cambridge University Press, New York.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017a. An evaluation of PredPatt and open IE via stage 1 semantic role labeling. In *Proceedings of the 12th International Conference on Computational Semantics — Short Papers*, Montpellier, France.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017b. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Xinyuan Zhang, Yi Yang, Siyang Yuan, Dinghan Shen, and Lawrence Carin. 2019b. Syntax-infused variational autoencoder for text generation. In *Proceed-*

ings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2069–2078, Florence, Italy. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566, Berlin, Germany. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Kai Zhao, James Cross, and Liang Huang. 2013. Optimal incremental parsing via best-first dynamic programming. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 758–768, Seattle, Washington, USA. Association for Computational Linguistics.