# Syntactic Dependencies and Beyond: Robust Neural Architectures and Quality-Enhanced Corpora for Structured Prediction in NLP

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Philosophie (Dr. phil.) genehmigte Abhandlung

Vorgelegt von

Stefan Grünewald

aus Hagen

| | |
|---|---|
| Hauptberichter | Prof. Dr. Jonas Kuhn |
| Mitberichterin | Prof. Dr. Annemarie Friedrich |
| Mitberichter | Dr. Djamé Seddah |

Tag der mündlichen Prüfung: 29. November 2024

Institut für Maschinelle Sprachverarbeitung
der Universität Stuttgart

2025

**Erklärung (Statement of Authorship)**
Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet.

(Ort, Datum) (Stefan Grünewald)

# Contents

# Abstract

This thesis investigates **explicit structure in Natural Language Processing (NLP).** Such structure, represented by abstract linguistic objects like part-of-speech tags, syntax trees, or graph-based meaning representations, has traditionally played a central role in NLP. Historically linked to the idea of rule-based processing of human language (using tools such as formal grammars), it has also successfully been combined with statistical machine learning techniques. For practical applications, this has often taken the form of a pipeline in which the prediction of linguistic features serves as a first step towards addressing "higher-level" tasks such as text classification or information extraction. In addition, algorithmic extraction of linguistic structures is also being pursued for its own sake, i.e., as a means to deepen our understanding of human language.

Most recently, the field of NLP has been dominated by techniques leveraging artificial neural networks. In this paradigm, language data is not processed using a pipeline approach as outlined above, which is ultimately grounded in simple and interpretable features. Rather, neural networks learn internal, vector-based language representations by means of large-scale mathematical optimization based on (usually very large amounts of) raw input data, allowing for "end-to-end" language processing that does not involve any kind of explicit structure as an intermediate representation. While the successes of this paradigm are undeniable in practical terms – i.e., achieving new state-of-the-art results on a wide range of applications ranging from information extraction to machine translation –, it has also spurred controversial questions around the present and future role of explicit structure in NLP. At an overarching level, this means a general uncertainty about the role of explicit structure: Can NLP still benefit from modeling structure explicitly, or have such approaches become obsolete? Apart from this fundamental question, however, the interaction between neural networks and explicit structure in NLP also raises a number of practical challenges; and it is these challenges that form the core of this thesis and the basis for its contributions.

The first challenge relates to the **role of data in training structure-prediction systems.** As a general rule, neural networks require large amounts of (labeled) training

data for learning specific tasks, and thus the curation and annotation of suitable datasets is a common bottleneck in their development. In our contributions, we focus on the Universal Dependencies (UD) formalism for the annotation of syntactic dependencies. Evaluating the quality of existing treebanks and examining ways of improving and extending them, one of our core findings it that both rule-based and machine learning-based methods can be leveraged to reduce the need for manual annotation.

The second challenge relates to the **design and architecture of neural structure-predicting systems,** of which there exists a wide variety; often, it is not fully clear which factors are truly important in achieving the best possible performance. We study dependency parser architectures for UD parsing, finding that when using modern neural network backbones, simpler is often better, with more sophisticated setups offering little in the way of performance improvements.

The third challenge relates to **structure prediction for downstream NLP tasks.** Here, we investigate the tasks of Negation Resolution and Relation Extraction by means of framing them as graph parsing problems and utilizing neural architectures similar to those studied for dependency parsing. We find that such an approach generally yields robust results, but is not clearly superior to "shallow" sequence labeling.

In sum, we hope that our contributions serve to inform and inspire future research on the role of explicit structure in NLP, and more generally within the emerging paradigm of artificial intelligence (AI) that combines neural networks with rule-based algorithms and symbolic representations ("neuro-symbolic AI").

# Zusammenfassung

Diese Dissertation befasst sich mit **expliziten Strukturen im Kontext maschineller Sprachverarbeitung (Natural Language Processing, NLP).** Solche Strukturen, verkörpert durch abstrakte linguistische Objekte wie bspw. Part-of-Speech-Tags, Syntaxbäume oder graph-basierte Bedeutungsrepräsentationen, haben seit jeher eine zentrale Rolle in NLP inne. Sie sind historisch verwurzelt in regelbasierten Ansätzen zur Verarbeitung menschlicher Sprache (z.B. mittels formaler Grammatiken), wurden in der Folge aber auch mit statistischen Machine-Learning-Techniken kombiniert. Für praktische Anwendungen hat dies oft die Form von Pipelines angenommen, in denen die Vorhersage linguistischer Features als vorgelagerter Schritt zur Lösung von Zielaufgaben wie Textklassifikation oder Informationsextraktion dient. Darüber hinaus wird die algorithmische Extraktion linguistischer Strukturen aber auch um ihrer selbst willen verfolgt – d.h., um unser Verständnis der menschlichen Sprache zu vertiefen.

In jüngster Zeit wird das NLP-Feld von Techniken dominiert, die auf künstlichen neuronalen Netzwerken aufbauen. In diesem Paradigma wird Sprachverarbeitung nicht nach dem oben beschriebenen Pipeline-Ansatz durchgeführt, der letztlich auf einfachen und interpretierbaren Features basiert. Stattdessen lernen neuronale Netzwerke interne, vektorbasierte Sprachrepräsentationen durch mathematische Optimierungsverfahren auf Grundlage von (meist sehr großen Mengen) roher Daten, was eine „end-to-end"-Sprachverarbeitung ermöglicht, die keine explizite Struktur als Zwischenrepräsentation einbezieht. Während die Erfolge dieses Paradigmas in praktischer Hinsicht unbestreitbar sind – in einer Vielzahl von Anwendungen, von der Informationsextraktion bis hin zur maschinellen Übersetzung, wurden neue State-of-the-Art-Ergebnisse erzielt –, hat es auch kontroverse Fragen über die gegenwärtige und zukünftige Rolle expliziter Strukturen in NLP aufgeworfen. Auf einer übergeordneten Ebene bedeutet dies eine allgemeine Unsicherheit über die Rolle expliziter Strukturen: Kann NLP immer noch von der expliziten Modellierung von Strukturen profitieren, oder sind solche Ansätze inzwischen obsolet geworden? Abgesehen von dieser grundlegenden Frage wirft die Interaktion zwischen neuronalen Netzen und expliziten Strukturen in NLP aber auch eine Reihe

praktischer Herausforderungen auf; und es sind diese Herausforderungen, die den Kern dieser Dissertation bilden und die Basis für ihre Beiträge darstellen.

Die erste Herausforderung betrifft **die Rolle von Daten im Training von Systemen zur Vorhersage von Strukturen.** Grundsätzlich benötigen neuronale Netze große Mengen an (gelabelten) Trainingsdaten, um spezifische Aufgaben zu erlernen, und somit stellt die Auswahl und Annotation geeigneter Datensätze oft einen Engpass in ihrer Entwicklung dar. In unserer Arbeit konzentrieren wir uns auf den Universal-Dependencies-Formalismus (UD) zur Annotation syntaktischer Dependenzen. Wir bewerten die Qualität bestehender Baumbanken und untersuchen Methoden, wie diese verbessert und erweitert werden können. Dabei ist eines unserer zentralen Ergebnisse, dass sowohl regelbasierte Methoden als auch Machine-Learning-Ansätze genutzt werden können, um den Bedarf an manueller Annotation zu reduzieren.

Die zweite Herausforderung betrifft **das Design und die Architektur neuronaler Systeme zur Vorhersage von Strukturen.** Unter solchen Systemen herrscht eine große Vielfalt und oft ist nicht völlig klar, welche Design-Faktoren wirklich entscheidend sind, um die bestmögliche Performance zu erzielen. Wir untersuchen Parser-Architekturen für das UD-Parsing und stellen fest, dass bei Systemen, die auf modernen neuronalen Netzwerk-Architekturen aufbauen, oft „einfacher ist besser“ gilt; komplexere Systeme resultieren selten in besserer Performance.

Die dritte Herausforderung betrifft **die Vorhersage von Strukturen für nachgelagerte NLP-Aufgaben.** Hier untersuchen wir Negationsauflösung und Relationsextraktion, indem wir beide Aufgaben als Graph-Parsing-Probleme formulieren und neuronale Architekturen verwenden, die denen ähneln, die wir für das Dependenzparsing untersucht haben. Wir stellen fest, dass ein solcher Ansatz im Allgemeinen robuste Ergebnisse liefert, aber „flachen“ Sequenz-Labeling-Ansätzen nicht eindeutig überlegen ist.

Zusammenfassend hoffen wir, dass unsere Beiträge als Inspiration und Ausgangspunkt für zukünftige Forschung zur Rolle expliziter Strukturen in NLP dienen werden. Von besonderem Interesse sind hierbei Ansätze zur künstlicher Intelligenz, die neuronale Netze mit regelbasierten Algorithmen und symbolischen Repräsentationen kombinieren („neuro-symbolic AI“).

# Acknowledgements

First and foremost, I want to thank my thesis advisors, Jonas Kuhn and Annemarie Friedrich. Without their continued support, suggestions, and encouragement, completing this thesis would not have been possible. I would also like to thank Djamé Seddah and Sabine Schulte im Walde, who joined the committee for my doctoral defense.

Furthermore, I am grateful to all those colleagues and collaborators whom I have worked with (directly or indirectly) over the years, both at Bosch as well as the IMS. These include: Sophie Henning, Hendrik Schuff, Timo Schrader, Lukas Lange, Subhash Pujari, Heike Adel, Jannik Strötgen, Elizaveta Sineva, Prisca Piccirilli, Frederik Oertel, Neele Falk, Tanise Ceron, and Chris Jenkins.

Last but certainly not least, my thanks go out to everyone who supported me during my time as a doctoral student. This includes my parents, Irene Dopp-Grünewald and Klaus Grünewald; my aunt, Sabine Dopp; and all those amazing people I am fortunate to have as my friends. These include: Sophie Henning (again!), Simon Ahrendt, Leonhard Brandt, Leander Vogel, Sebastian Barszczewski, Cornelia Keller, Jan Kosina, Wenying Wu, Katie Lu, Aysun Gökçe, Martina Overbeck, Caroline Tisson, Simon Kast, and Johanne Dziewas.

Inevitably, I will have forgotten someone. For this I can only apologize; please know that the omission is only due to my bad memory, and I am grateful to you nonetheless.

**Support:** The work presented in this thesis was supported by Robert Bosch GmbH, where I was employed as a doctoral researcher in the Natural Language Processing group from November 2019 to December 2022.

# Part I.

# Introduction: Syntax, Parsing, and Structured Prediction in Natural Language Processing

# 1. Motivation and Overview

Natural Language Processing (NLP) has seen staggering process in the past decade. In virtually all areas, from Speech Recognition to tasks involving syntax and semantics, results that were once thought to be out of reach have now become the norm, with model performance on benchmark tasks often reaching or arguably even outperforming that of humans. The main driver of this remarkable progress lies in the widespread usage of artificial neural networks, whose sophisticated structures allow them to learn from raw text data with very high efficacy whilst requiring little to no manual feature engineering. In particular, one of the predominant approaches in contemporary NLP is to utilize large language models (sometimes referred to as "foundation models" for language; Bommasani et al., 2021): These large neural networks are first "pre-trained" on some auxiliary task with little human supervision and through this learn to produce information-rich representations of words within their context. In a second step, the models are then further trained ("fine-tuned") on target tasks (e.g., Named Entity Recognition, Relation Extraction, or Syntactic Parsing) in a classical supervised fashion, i.e., using human-annotated data. Taken together, this pre-training/fine-tuning paradigm has fueled large parts of the rapid progress within NLP since ca. 2018.

While undeniably impressive, these advances have also led to considerable uncertainty and disagreement about the direction of NLP as a field. The role of large language models forms a main part of these debates, with questions on their energy usage and resulting environmental impacts (Strubell et al., 2019), ethical and privacy concerns (Carlini et al., 2020; Weidinger et al., 2021), as well as philosophical disagreements about the extent to which language models can be said to "understand" natural language (Bender and Koller, 2020).

In this dissertation, we address one particularly contentious aspect of these debates: the role of **explicit structure** in the age of neural network-based NLP. By explicit structure, we here mean expert-designed formal representations such as syntax trees, graph-based meaning representations, or sequence labeling schemes. While there exists a rather broad consensus within the NLP community that language understanding cannot

be solved by simply continuing to increase model size, it is divided on the question of whether such structures are necessary for it (Michael et al., 2022). This disagreement is rooted in the trend towards fully "end-to-end" systems in recent years, which rely less and less on human-designed structured intermediate representations and instead learn mappings from input text to output structures/labels solely from the data itself.

This dissertation focuses on issues related to **using modern neural systems to produce dependency-structured outputs,** which we examine from three perspectives:

1. The first perspective is that of theoretical computational linguistics and addresses the issue of creating and maintaining **high-quality syntactic dependency data,** which also continues to be of central importance in the area of corpus linguistics. Focusing on the Universal Dependencies (UD) formalism for syntactic dependencies, we examine corpora (treebanks) annotated with syntactic dependency structures, investigate methods for improving and extending them, and monitor the effect of our corpus modifications on syntactic parser performance. A particular emphasis is placed on rule-based methods based on linguistic knowledge, whose performance we find to vary greatly depending on the exact nature of the task.

2. For the second perspective, we shift to a computational view of syntactic dependencies and operate within the bounds of a classical, linguistically motivated structured prediction task: **Syntactic Dependency Parsing.** We perform comparisons between model variants of graph-based UD parsers for a wide variety of languages built on top of modern language models in order to determine the factors most crucial for high parsing accuracy. Our main findings are that the choice of underlying language model is by far the most influential factor determining parser performance, with models with more parameters and larger amounts of pre-training data usually performing better.

3. For the third perspective, we broaden our scope and investigate methods for **incorporating dependency structure into models for other NLP applications,** building on our insights from syntactic dependency parsing. Here, we examine the tasks of Negation Resolution and Relation Extraction, which we show can be tackled successfully by framing them as graph parsing problems. Such approaches may facilitate, e.g., search applications over knowledge graphs incorporating fine-grained information from large numbers of documents.

The following sections provide further context for each perspective, introduce the associated research questions, and give previews of our findings for each.

## 1.1. High-quality Syntactic Dependency Data

*Syntax*, the study of how words combine to form larger structures such as sentences, constitutes an integral part of theoretical linguistics and has traditionally been an important focus in Natural Language Processing and Computational Linguistics. As a result, much effort has been dedicated to creating resources comprised of natural language text annotated with with syntactic structures, such as the Penn Treebank (Marcus et al., 1993) for constituency structures and the Universal Dependencies project (de Marneffe et al., 2014) for dependency structures. Such training data is of crucial importance for training machine learning systems to predict syntactic structures from raw text (i.e., syntactic parsers).

In general, hand-annotating sufficiently large amounts of high-quality data is difficult and costly. Conversely, using automated methods (either rule-based or machine learning-based) to produce such data often results in noisy annotations. A common example of this are conversions from one syntactic framework to another. Our first main area of research in this dissertation therefore looks at issues involved with evaluating and improving syntactic dependency datasets, specifically UD treebanks. We address the following research questions:

- **RQ1: What is the quality of existing UD datasets?** As of August 2023, the UD project contains nearly 200 treebanks for over 100 languages. However, many of these treebanks result from automatic conversion and have not been manually checked; this is especially true for the Enhanced UD formalism (see Sec. 3.3 for details). Therefore, the question arises how good these treebanks are (e.g., with regards to adherence to the UD guidelines and with respect to specific relevant linguistic phenomena such as coordination). Our analyses indicate that existing UD treebanks for English and German, while generally of high quality, contain systematic errors and inconsistencies for certain linguistic phenomena such as coordinate structures or preposition-determiner contractions.

- **RQ2: How can we improve and extend treebanks in an effective way with limited manual resources?** Manual annotation of syntactic structures requires linguistic experts and is therefore time-consuming and costly. Thus, there is a need for efficient yet accurate automated methods for treebank improvement (e.g., fixing systematic annotation errors) and extension (e.g., to the more complex Enhanced UD formalism). A related question is whether such techniques for treebank improvement/extension that currently only exist for English can easily

be transferred to other languages. In experiments on English and German UD treebanks, we find that both rule-based and machine learning-based methods can be used for treebanks improvements and extensions.

- **RQ3: What is the effect of improvements to UD treebanks?** Once improvements to a given UD treebank have been performed (either manually or automatically), their impact on parsing performance must be quantified, e.g., in terms of parsing metrics such as Labeled Attachment Score (LAS). Here, we find that harmonizing annotations across treebanks can lead to improvements in parser accuracy due to less noise in the training signal.

## 1.2. Neural Graph-based Dependency Parser Architectures

*Syntactic Dependency Parsing* refers to the task of predicting, for a given sentence, the grammatical relations between its tokens. These relations may stem from different dependency formalisms such as Stanford dependencies (De Marneffe and Manning, 2008) or the Universal Dependencies framework (see above and Chapter 3). A *syntactic dependency parser* is a program that automatically retrieves, for a given input sentence, a dependency tree or graph describing the grammatical relations within it.

Syntactic parsing has traditionally been a centerpiece of Computational Linguistics and Natural Language Processing, with a large and differentiated body of research. The most obvious reason for this is that explicit syntactic information was used – and continues to be used – as an input to systems across a wide range of NLP applications, including Named Entity Recognition (Jie and Lu, 2019), Relation Extraction (Alt et al., 2020), and Open Information Extraction (Gashteovski et al., 2017). In addition, we argue that syntactic parsing is also central to NLP because it may be considered a typical example of retrieving explicit structures from textual surface realizations – a paradigm that is also present in many other NLP applications such as Event Extraction or Semantic Parsing. In fact, a variety of NLP tasks have been explicitly modelled as dependency parsing tasks, including (nested) Named Entity Recognition (Yu et al., 2020), Negation Resolution (Kurtz et al., 2020, see also Chapter 9), and Semantic Role Labeling (Zhang et al., 2021).

Assuming the availability of suitable training data, there exist numerous ways in which dependency parsers can be built using it. Architectural dimensions include the choice of underlying language model, whether to include recurrent neural network layers,

and whether to also model other linguistic dimensions (e.g., part-of-speech tags) in a multi-task setting. In light of this, we investigate the following research questions related to parser architectures:

- **RQ4: Which are the main factors determining the performance of neural graph-based dependency parsers?** In spite of a recent convergence on parser architectures built on transformer-based language models, there remains considerable variation in the concrete implementation of graph-based neural parsers. Therefore, there is a need for comparing the impact of different architectural decisions (e.g., choice of language model or whether to employ multi-task learning) on parser performance, and in the spirit of "Occam's razor," one may ask which architectural simplifications are possible without sacrificing performance. In a series of experiments, we find that the choice of language model is by far the most important factor influencing parser performance, whereas other architectural choices play a much lesser role. For example, adding additional recurrent layers on top of a transformer-based language model does not improve performance.

- **RQ5: How can neural dependency parsers be adapted for Enhanced UD?** Assuming that a state-of-the-art neural parser for basic UD is available, it is not obvious that its architecture is also suited for processing the more complex enhanced UD formalism; in general, interest in enhanced UD parsing has only recently begun to pick up. Therefore, it is necessary to evaluate ways of adapting parsing techniques to Enhanced UD parsing. We demonstrate that a state-of-the-art graph-based UD parser can be adapted to Enhanced UD with relatively minor adjustments.

## 1.3. Dependency Parsing-based Methods for NLP Applications

Finally, given the high accuracy of modern methods for syntactic dependency parsing, the question arises: Is it possible apply similar techniques to other NLP tasks – especially those that involve modeling relations between words or spans within a sentence in some way – and if so, how exactly should this be implemented? As outlined in Sec. 1.2, existing work has framed a variety of tasks as dependency parsing problems. In this context, we examine the following research question:

- **RQ6: Can we leverage the biaffine dependency parser architecture for other NLP applications?** The biaffine dependency classifier architecture is one main reason behind the strong performance of modern graph-based dependency parsers, and has been shown to be useful for other NLP tasks such as Named Entity Recognition (Yu et al., 2020). We ask how this approach stacks up against other paradigms such as sequence labeling, examining two application areas. First, building on methodology by Kurtz et al. (2020), we perform a series of experiments on Negation Resolution and find that biaffine parsing-based approaches can yield robust results, but are not always superior to other approaches such as sequence labeling. Second, we investigate Relation Extraction from scientific documents (specifically, materials science papers). Again, we find that the biaffine architecture yields good results, and extraction accuracy may be boosted further by employing a multi-task learning setup across datasets.

## 1.4. Summary

In this dissertation, we investigate a range of research questions related to the role of explicit structure in modern Natural Language Processing. Our research questions are grouped into three clusters, namely **(1) obtaining high-quality syntactic dependency data, (2) exploring architectural dimensions in syntactic dependency parsing,** and **(3) leveraging biaffine dependency parsing for downstream NLP applications.** Within the context of these clusters, we perform a wide range of experiments relating to improving and extending dependency treebanks, syntactic parser architectures, and structurally informed methods for NLP tasks.

Our results point towards explicit linguistic structure still having a role in modern NLP, albeit in relatively narrow contexts that are intrinsically related to linguistic structure to begin with – e.g., the creation and extension of syntactic dependency annotations. We also find that biaffine dependency parsing can be used to model other NLP tasks (e.g., Relation Extraction) and yield good results, however, this approach is not necessarily superior to simpler structured prediction methods such as sequence tagging. Our findings contribute to the ongoing debate about the role of explicit structure in NLP and we hope that they may spur further research in the area.

# 2. Thesis Plan and Contributions

In this chapter, we outline the contents of each of the parts of the dissertation, describing their relation to the knowledge gaps and research questions described above. We also provide a list of our contributions and their relevance to NLP as a field. Finally, we list the publications whose content forms the basis of parts of this dissertation.

## 2.1. Chapter Overviews

**Part II: Contributions to UD Data.** Part II details our **resource contributions** to the Universal Dependencies framework, addressing **RQ1–3** (see Sec. 1.1). This involves manual evaluation of existing enhanced UD treebanks, examining manual and automatic methods for improving and extending treebanks, and evaluating the effects of such improvements.

**Chapter 3** provides background on the Universal Dependencies formalism, with a special focus on Enhanced UD. This includes a brief outline of the history and motivation behind UD, a comparison and contrast with other dependency formalisms, and a description of common evaluation metrics and techniques.

**Chapter 4** contains an analysis of coordinate constructions in English Enhanced UD. We find that existing treebanks, created via automatic conversion from the basic UD format, contain systematic errors in the handling of coordination in the presence of certain linguistic phenomena. Using a manually fixed version of the English Web Treebank (EWT), we explore different methods of improving upon rule-based conversion. When using raw text as a starting point, parsing directly into Enhanced UD works best, indicating the power of end-to-end parsing-based approaches.

**Chapter 5** describes our studies on improving and extending German UD treebanks. This work involves two parts. As a first contribution, we perform an analysis of preposition-determiner contractions in German UD, finding that they are handled inconsistently between treebanks. We show that the issue can be fixed automatically with a simple rule-based algorithm, demonstrating that complex machine learning-based approaches are not

always needed, particularly in the presence of strong linguistic regularities. Furthermore, we show that our fixes improve parsing performance. As a second contribution, we evaluate rule-based methods for extending German treebanks with enhanced dependencies, finding that results perform well but not perfectly, indicating the higher complexity of this problem as compared to the handling of contractions.

**Part III: Computational Studies on Transformer-based UD Parsing.** Part III reports on our **algorithmic contributions** in the area of transformer-based dependency parser architectures for basic and Enhanced UD, addressing **RQ4 and RQ5** (see Sec. 1.2).

**Chapter 6** provides a general introduction to UD parsing. In particular, we focus on recent developments in neural (graph-based) parsing, highlighting the increasing dominance of transformer language models but also the still considerable variation in recent state-of-the-art parsers, e.g., with regards to employing Long-Short Term Memory networks (LSTMs; Hochreiter and Schmidhuber, 1997) or multi-task learning within models.

**Chapter 7** describes STEPS, our state-of-the-art modular UD parser designed to evaluate different architecture dimensions of neural graph-based parsers. In a series of experiments, we find that more complex parser architectures (relying, e.g., on LSTM layers or multi-task learning) do not generally lead to better performance, and instead, a simple model built on a high-quality transformer language model performs best. Our results support the idea that increasingly sophisticated neural word representations make structurally informed system architectures partially obsolete, at least in certain settings.

**Chapter 8** details our work on transferring STEPS's architecture to the challenge of Enhanced UD parsing. In particular, we describe novel methods for ensuring graph validity and incorporation of lexical material into dependency labels, and we show that model ensembling yields some performance gains. Our resulting Enhanced UD parser scored 1$^{st}$ on English in the IWPT 2020 Shared Task on Enhanced UD Parsing, and 3$^{rd}$ overall in the 2021 edition of the Shared Task. Our results indicate that rule-based model components can still be helpful for the syntactic parsing of Enhanced UD graphs.

**Part IV: Biaffine Dependency Parsing in NLP Applications** Part IV reports on our **transfer contributions** in applying the biaffine dependency parser architecture to downstream applications, addressing **RQ6** (see Sec. 1.3).

**Chapter 9** describes our work on employing the biaffine dependency classifier architecture for automated Negation Resolution. While our modeling yields good results, it

does not consistently outperform more traditional approaches such as sequence labeling.

**Chapter 10** describes our work on employing the biaffine dependency classifier architecture for Relation Extraction from materials science texts. Our experiments show that the biaffine architecture yields good results, and extraction accuracy may be boosted further by employing a multi-task learning setup across datasets.

**Part V: Conclusions.** Part V, consisting of **Chapter 11,** discusses and summarizes our contributions and concludes the dissertation.

## 2.2. List of Contributions

The contributions of this thesis are as follows:

- We evaluate the correctness and completeness of Enhanced Universal Dependency annotations in the context of coordinate constructions in the English Web Treebank corpus (Chapter 4).
- We propose improvements to an existing rule-based system for extracting Enhanced dependencies from basic UD trees (Chapter 4).
- We compare rule-based and machine learning-based methods for extracting Enhanced Universal Dependencies in the context of coordinate constructions, both from gold and predicted basic dependencies (Chapter 4).
- We analyze the treatment of preposition–determiner contractions in German UD treebanks and propose a rule-based method for harmonizing them. In a set of experiments, we how that consistent treatment of contractions leads to accuracy increases for parsers trained on this data (Chapter 5).
- We assess the accuracy of a rule-based system for extracting Enhanced UD from basic German treebanks (Chapter 5).
- We introduce STEPS, a new implementation of a graph-based dependency parser designed to be modular and easily extensible, and show that it achieved state-of-the-art results at the time of its release (Chapter 7).
- Using STEPS, we perform a detailed experimental study on parser architectures and identify components that are really necessary to obtain a strongly performing system (Chapter 7).

- We adapt STEPS to Enhanced UD parsing by employing a simple unfactorized biaffine classifier architecture, proposing a novel rule-based method for lexicalizing dependency labels as well as heuristics for ensuring graph validity (Chapter 8).
- We demonstrate the efficacy of STEPS for Enhanced UD parsing by participating in two editions of the IWPT Shared Task (Bouma et al., 2020, 2021), in which we score 1st (English only) and 3rd (17 languages), respectively (Chapter 8).
- We perform a comparison of approaches for automated Negation Resolution, showing that biaffine parsing-based methods perform well, but slightly worse than a traditional sequence-tagging approach (Chapter 9).
- We apply a biaffine parsing approach to Relation Extraction from materials science documents and show that it reaches good performance on multiple datasets, including in a multi-task setting (Chapter 10).

**Relevance to NLP/CL.** The contributions of this dissertation are relevant to the wider fields of Natural Language Processing and Computational Linguistics. In particular, we highlight aspects relating to the role of structured representations in multiple NLP sub-fields.

Machine learning-based NLP relies on annotated corpora. Regarding annotation and data collection, our results are relevant to the creation and curation of high-quality annotated corpora, as they show that both rule-based and machine learning-based methods can be effective at creating and improving annotation data in certain contexts. Rule-based methods are particularly appropriate for highly regular linguistic phenomena.

With regards to the study of neural architectures in the age of large pre-trained language models, our findings on parser architectures corroborate the idea that language model capacity of transformer-based systems is by far the most important factor influencing system performance. Nonetheless, there remain areas where specialized architectures and even rule-based components can be beneficial, as evidenced by the strong performance of biaffine classifiers and the efficacy of our rule-based lexicalization strategy for Enhanced UD parsing, respectively.

Finally, our findings are relevant to structured prediction in a more general sense. Predicting structures is an essential part of Natural Language Processing, as many commonly addressed tasks are concerned with mapping raw text input to structured (i.e., machine-readable) representations in some way. Our findings demonstrate that incorporating structure in the task formulation, e.g. by framing tasks like Negation

Resolution as a Dependency Parsing problem, is a valid approach, though one that does not always yield superior results to simpler methods. Overall, we find a mixed picture, which we hope will motivate further work in this direction.

## 2.3. Publications

The content of this dissertation is in large parts based on peer-reviewed publications. This work was done at Robert Bosch GmbH, where the author was employed as a doctoral researcher from November 2019 to December 2022. The following list enumerates the publications, along with the chapters their content appears in and the mode of collaboration between co-authors (where applicable).[1]

- Grünewald, S., Piccirilli, P., and Friedrich, A. (2021c). 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 [**Chapter 4**]

  I designed, implemented, and evaluated our converter and parser experiments. Prisca Piccirilli performed the annotation work for our manual corpus correction and collaborated in error analysis.

- Grünewald, S. and Friedrich, A. (2020b). Unifying the treatment of preposition-determiner contractions in German Universal Dependencies treebanks. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 94–98, Barcelona, Spain (Online). Association for Computational Linguistics [**Chapter 5**]

- Grünewald, S., Friedrich, A., and Kuhn, J. (2021a). Applying Occam's razor to transformer-based dependency parsing: What works, what doesn't, and what is really necessary. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 131–144, Online. Association for Computational Linguistics [**Chapter 7**]

- Grünewald, S. and Friedrich, A. (2020a). RobertNLP at the IWPT 2020 shared task: Surprisingly simple enhanced UD parsing for English. In *Proceedings of the*

[1] Thesis advisors are exempt here as their role was always one of high-level supervision and guidance.

*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 [**Chapter 8**]

- Grünewald, S., Oertel, F. T., and Friedrich, A. (2021b). RobertNLP at the IWPT 2021 shared task: Simple enhanced UD parsing for 17 languages. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 196–203, Online. Association for Computational Linguistics [**Chapter 8**]

  I designed and implemented the STEPS/RobertNLP parsing system, prepared our submissions, and performed further evaluations. Frederik Oertel implemented the label transducer subsystem.

Additionally, some of the content of this dissertation is based on peer-reviewed publications on which the the author has collaborated, but is not the main author. In these cases, this dissertation focuses on content which fell under the author's purview in the collaboration, which is briefly outlined in the following. The relevant papers are:

- Bürkle, T., Grünewald, S., and Friedrich, A. (2021). A corpus study of creating rule-based enhanced Universal Dependencies for German. In *Proceedings of The Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, pages 85–95, Punta Cana, Dominican Republic. Association for Computational Linguistics [**Chapter 5**]

  Teresa Bürkle implemented the rule-based Enhanced UD converter, while I collaborated in conceptual design work and performed the evaluation experiments.

- Sineva, E., Grünewald, S., Friedrich, A., and Kuhn, J. (2021). Negation-instance based evaluation of end-to-end negation resolution. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 528–543, Online. Association for Computational Linguistics [**Chapter 9**]

  Elizaveta Sineva implemented Negation Resolution evaluation scripts and biaffine parsing-based systems, while I implemented sequence-tagging models and performed the evaluation experiments.

- Schrader, T. P., Finco, M., Grünewald, S., Hildebrand, F., and Friedrich, A. (2023). MuLMS: A multi-layer annotated text corpus for information extraction in the materials science domain. In *Proceedings of the Second Workshop on Information*

*Extraction from Scientific Publications*, pages 84–100, Bali, Indonesia. Association for Computational Linguistics [**Chapter 10**]

Timo Schrader worked on entity extraction, while I was responsible for designing and implementing the parsing-based relation extraction system. Matteo Finco and Felix Hildebrand are domain experts on materials science.

# Part II.

# Contributions to (Enhanced) Universal Dependencies

# 3. Introduction to Universal Dependencies

In this part of the dissertation, we turn our focus to the analysis, improvement, and extension of Universal Dependencies corpora (*treebanks*). Treebanks are of crucial importance for syntactic dependency parsers as they provide the data used in supervised training of these systems. However, they are usually created via human annotation, which is costly and time-consuming. Additional complications can arise when treebanks are created through (usually rule-based) conversion from a different dependency framework, as is the case for multiple UD treebanks.

Against this background, we consider English and German UD treebanks and address our research questions **RQ1 (What is the quality of existing UD datasets?), RQ2 (How can we improve and extend treebanks in an effective way with limited manual resources?),** and **RQ3 (What is the effect of improvements to UD treebanks?).** From a methodological point of view, we perform both quantitative and qualitative analyses of UD treebank data, assessing to what extent they contain systematic errors or inconsistencies. In addition, we propose rule-based as well as machine learning-based methods for improving and extending UD treebanks.

As a first main contribution, we consider the linguistic phenomenon of **coordination** in English Enhanced UD treebanks. We analyze weaknesses in the representation of this phenomenon in currently existing treebanks and propose rule-based as well as machine learning-based methods for improving the annotations.

As a second main contribution, we consider the phenomenon of **preposition-determiner contractions** (e.g., "*zur* Schule" instead of "*zu der* Schule") in German UD treebanks, analyzing inconsistencies in their handling across corpora and proposing a rule-based strategy for harmonizing their representation. We find that parser accuracy increases when training on treebanks with harmonized contractions due to less noise in the training signal.

As a third main contribution, we address the issue of **enriching the representation**

**in German UD treebanks** with Enhanced Universal Dependencies. Prior to our work in this dissertation, Enhanced dependencies have not been present in German UD treebanks. Building on similar work for English, we propose a rule-based converter for creating Enhanced dependencies from the basic layer, and show that it achieves strong accuracy when used on gold basic annotations. Performance on automatically parsed sentences is lower, but may still be useful in practice.

**Outline of this part of the dissertation.** The following chapters are structured as follows:

- This chapter provides a brief introduction to syntactic dependencies in general and the Universal Dependencies formalism in particular. Aside from outlining the core ideas behind syntactic dependencies, we also provide details of the UD framework, e.g., regarding evaluation metrics. Another focus is the framework of Enhanced UD, which extends dependency trees into graph structures for better representation for certain linguistic phenomena (e.g., relative clauses).

- Chapter 4 describes our work in analyzing and improving the representation of coordinate structures in English Enhanced UD (Grünewald et al., 2021c).

- Chapter 5 describes our studies on improving and extending German UD treebanks. This covers our work on preposition-determiner contractions (Sec. 5.1; Grünewald and Friedrich, 2020b) as well as our work on extending German treebanks with Enhanced dependencies (Sec. 5.2; Bürkle et al., 2021).

## 3.1. Syntactic Dependencies

A *syntactic dependency* is an asymmetric relation between two words in a sentence that describes their grammatical relationship (Jurafsky and Martin, 2021). Dependencies describe the relation between a *head* (or *governor*) and a *dependent*. The dependent modifies the head in some way – either as an argument or as an optional modifier – which may be specified by a relation label. For example, in the sentence "Mary likes fluffy cats" (see Figure 3.1), the word "Mary" is a dependent of the word "likes" (as indicated by the direction of the corresponding arrow in the diagram). The label *subject* specifies that it is fills the subject slot. Similarly, the word "fluffy" is a dependent of the word "cats" (in this case an adjectival modifier, as indicated by the *adjmod* label).

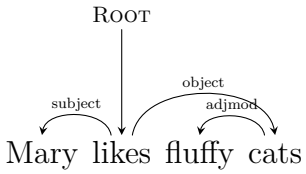



Figure 3.1.: Example dependency tree.

A *dependency tree* is a graph structure in which each token constitutes a node and is assigned exactly one parent (its syntactic head). The parent may be either one of the other words in the sentence or an additional, implicit ROOT node to signify the syntactic root of a sentence (usually its main verb). In most dependency formulations, a valid dependency tree for a sentence must have exactly one root, which all tokens must be reachable from. Figure 3.1 shows a complete labeled dependency tree for the sentence "Mary likes fluffy cats."

## 3.2. The Universal Dependencies Formalism

This section provides a more detailed description of Universal Dependencies (UD), a specific framework for syntactic dependencies.

### 3.2.1. Basic Concepts

UD is a dependency grammar framework that is driven by a large-scale multi-lingual community effort (de Marneffe et al., 2014) and aims to provide dependency annotations following a uniform annotation scheme across a wide range of human languages. As of 2024, the UD project provides annotated treebanks for almost 200 languages.[1] UD is based on Stanford Dependencies (SD; de Marneffe and Manning, 2008), but differs from it (and other dependency formalisms) on a number of dimensions, which we will outline in the following.

**Relation inventory.** UD specifies an inventory of 37 dependency relations, which are organized around the linguistic units of clauses, nominals, modifier words, and function words. Which relation is used depends on which of these units the governor and dependent belong to. For example, a nominal dependent of a nominal is connected via *nmod*, while a nominal dependent of a clause is annotated as *obl* (as illustrated in Figure 3.2 for the

[1] `https://universaldependencies.org/`


(a) (b)

Figure 3.2.: Nominal dependents in UD.


(a) (b)

Figure 3.3.: Nominal and clausal subjects in UD.

two interpretations of the sentence "I saw a man with a telescope"). Similarly, a nominal subject is attached to a verb via the *nsubj* relation, while the *csubj* relation is used for clausal subjects (Figure 3.3).

**Emphasis on content word relations.** UD aims to establish dependencies immediately between content words, rather than mediating through function words. The intention behind this is to have the dependencies reflect the predicate–argument structure of a sentence. As a result of this, function words usually do not have dependents themselves, which is in contrast to other dependency formalisms such as Stanford Dependencies.

Prepositional phrases are one linguistic phenomenon where this difference is particularly apparent. In many dependency formalisms (including Stanford Dependencies), prepositional phrases are headed by the corresponding preposition. Figure 3.4a shows this for the example sentence "Henry went to school," where in the Stanford Dependency representation, "school" is a dependent of the preposition "to" (which is itself a dependent of the verb "went"). By contrast, the UD analysis of the same sentence (Figure 3.4b) establishes a direct dependency directly between the content words "went" and "school" and demotes the preposition "to" to a dependent (case marker) of "school".

The focus on relations between content words is one of the most controversial aspects of UD, with critics arguing that it leads to inconsistent annotation of parallel constructions (Gerdes and Kahane, 2016). "Surface-syntactic Unicersal Dependencies" (SUD) have been proposed as an alternative formalism that places higher emphasis on the status of function words (Gerdes et al., 2018).

**Additional linguistic annotations.** While not immediately related to its dependency representation *per se,* UD treebanks usually also contain annotations for other linguistic


(a) (b)

Figure 3.4.: Handling of prepositional phrases in Stanford Dependencies (a) vs. Universal Dependencies (b).

dimensions: part-of-speech (POS) tags, lemmas, and morphological features. Like grammatical relations, POS tags and morphological features have label inventories aiming at cross-lingual consistency ("Universal POS tags" and "universal morphological features", respectively).

In practice, these annotations are used for various purposes, including the training of standalone part-of-speech and/or morphological taggers as well as multi-task models for joint syntactic parsing and tagging (see also Chapter 7). Additionally, morphological information and lemmas are taken into account in certain UD evaluation metrics (MLAS, BLEX; see Sec. 3.2.2).

**Tokenization.** The word constitutes the basic unit of syntactic analysis in UD. However, with the concept of "word" being rather underspecified, tokenization (word segmentation) is of crucial importance as a basic step in the creation of UD corpora. While this can be simple in many cases (e.g., simply splitting by whitespace), there are also cases where it constitutes a complex problem. Frequently, this involves cases where there is no one-to-one mapping from *orthographic tokens* to *syntactic words:* Vietnamese, for example, places whitespace at syllable boundaries, meaning that syntactic words may contain whitespace; other languages, including French and German, feature the opposite phenomenon and allow for contractions that fuse multiple syntactic words into a single token. Handling such cases consistently poses a challenge both in parsing from raw text as well as in treebank creation (see also Sec. 5.1 for an in-depth study of German contractions in UD).

### 3.2.2. Technical Aspects

We here describe two of the more technical aspects of the UD framework: The ConLL-U file format for UD treebanks, and UD evaluation metrics.

**CoNLL-U file format.** The standard representation of UD treebanks is the **CoNLL-U** file format. CoNLL-U is a plaintext format based on the underlying principle of displaying one syntactic word per line, with all information pertaining to that word separated by tabs. Sentences are separated by empty lines and start with metadata (indicated by lines starting with the # symbol) that minimally contains a sentence ID as well as the sentence's raw text.

```
# sent_id = dev-s20
# text = Ich kann dieses Geschäft nur wärmstens empfehlen.
1       Ich         ich         PRON    PPER    Case=Nom|PronType=Prs   7   nsubj   _   _
2       kann        können      AUX     VMFIN   Mood=Ind|VerbForm=Fin   7   aux     _   _
3       dieses      dies        PRON    PDAT    Case=Acc|Gender=Neut    4   det     _   _
4       Geschäft    Geschäft    NOUN    NN      Case=Acc|Gender=Neut    7   obj     _   _
5       nur         nur         ADV     ADV     _                       6   advmod  _   _
6       wärmstens   wärmstens   ADV     ADV     _                       7   advmod  _   _
7       empfehlen   empfehlen   VERB    VVINF   VerbForm=Inf            0   root    _   SpaceAfter=No
8       .           .           PUNCT   $.      _                       7   punct   _   _
```

Figure 3.5.: Example of a single sentence in CoNLL-U format (sentence taken from the German-GSD treebank; some morphological features ommitted for brevity).

For an example of a UD sentence in CoNLL-U format, see Figure 3.5. The tab-separated columns contain the following information:

1. The token's ID, counting up from 1 (the token ID 0 is reserved for a special *root* token);
2. The actual token form as present in the raw text;
3. The token's lemma;
4. The token's Universal part-of-speech (UPOS) tag;
5. The token's language-specific part-of-speech tag;
6. The token's morphological features, as key-value pairs;
7. The ID of the token's syntactic head (i.e., the ID of its parent in the dependency tree);
8. The dependency label by which the token is connected to its parent;
9. Dependency relations by which the token is connected to its parent(s) in the Enhanced UD representation (see Sec. 3.3)
10. Any other annotations (e.g., whitespace information such as `SpaceAfter=No`).

If information is unavailable (e.g., the enhanced dependency layer in Figure 3.5) or does not apply (e.g., morphological features for punctuation), the respective column contains a placeholder (underscore) instead.

**Evaluation.** The goal of Universal Dependencies evaluation is to quantify to what extent automatic systems (i.e., parsers; see Chapter 6 for further details) are able to reconstruct the correct syntactic dependencies for a given sentence, or for a collection of sentences. To this end, predicted syntax trees (i.e., system output) are usually compared against human-annotated gold-standard trees. For example, in Figure 3.6, subfigure (a) depicts the gold-standard dependencies for the sentence "I talked to my client over the phone," while subfigure (b) depicts a dependency tree for the same sentence that might be generated by a parsing system. As can be seen, in this case the parser made two errors, namely incorrect attachment of the prepositional phrase "over the phone" (*nmod* link originating at "client" rather than *obl* link originating at "talked") and wrong labeling of the syntactic relation between "client" and "my" (*det* instead of *nmod:poss*).

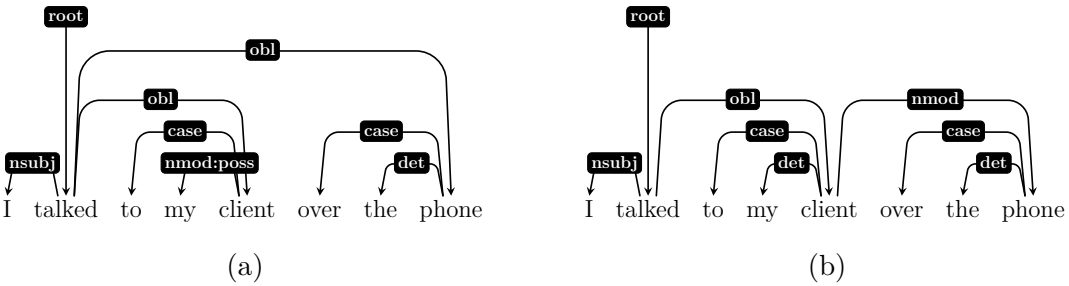



Figure 3.6.: Gold-standard dependency tree (a) and example prediction (b) for the sentence "I talked to my client over the phone." Note the discrepancies w.r.t. dependency edges.

A number of different UD evaluation metrics exist. In UD trees, every token has exactly one parent; therefore, the core idea underlying all of these metrics is to quantify the extent to which tokens are correctly attached. However, the metrics differ in the specific criteria that they apply for an attachment to count as correct.

**LAS (Labeled Attachment Score)** is the standard evaluation metric for dependency parsing (Zeman et al., 2018). A token counts as correctly attached if its parent in the predicted dependency tree is the same as in the gold-standard tree, and the dependency label matches as well. **UAS (Unlabeled Attachment Score)** is a less strict version of the LAS metric that removes the requirement of a matching dependency label, thus only evaluating the system's ability to produce correct tree structures.

Proposed by Nivre and Fang (2017), **CLAS (Content Labeled Attachment Score)** is a variant of LAS that excludes functional relations (e.g., determiners and auxiliarly verbs) and only takes into account content word relations (e.g., subject or object). The reasoning behind this is that LAS produces inflated scores for languages that tend to encode information via function words (e.g., English) while deflating scores for languages that encode information through rich morphology and use comparatively fewer function words (e.g., Finnish). This is due to the fact that in the latter case, there are fewer relations in total, meaning each mistake is weighted more heavily, and additionally, functional relations are usually easy to predict.

**MLAS (Morphology-Aware Labeled Attachment Score)** and **BLEX (Bilexical Dependency Score)** (Zeman et al., 2018) are extensions of CLAS that remain restricted to content word relations, but impose additional constraints for an attachment to count as correct. For MLAS, a relation only counts as correct if the morphological features of the token are also correctly predicted. Additionally, if the token has function word dependents in the gold standard, these must also be correctly attached in the system prediction (i.e.,

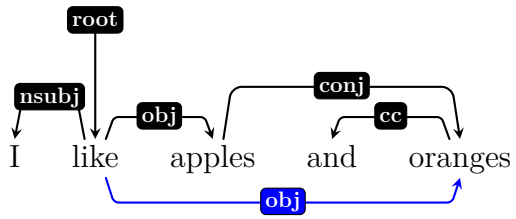



Figure 3.7.: UD **basic** (top) and **enhanced** (bottom) dependencies for a coordinate construction.

function words are treated in parallel to morphological features). For BLEX, a relation only counts as correct if the respective token is also correctly lemmatized, yielding an evaluation that is closer to the semantic content of a sentence.

## 3.3. Enhanced Universal Dependencies

Our discussion so far has focused on what is called the *basic* UD framework, which is a "standard" syntactic dependency formalism in the sense that annotation structures are classical dependency trees. However, Schuster and Manning (2016) propose **Enhanced Universal Dependencies (EUD)** as an extension of basic UD designed with shallow natural language understanding tasks in mind. Enhanced UD extends basic UD trees by including a number of additional dependencies between tokens in order to make relations between content words more explicit, especially in the presence of linguistic phenomena such as coordination, raising/control, and relative clauses. As a result, Enhanced UD allows for general dependency *graphs,* rather than just dependency *trees,* i.e., tokens may have more than one syntactic parent (incoming edge in the graph).

Figure 3.7 shows an example of this for the linguistic phenomenon of coordination. As can be seen, in the enhanced representation, the word "oranges" is not merely a dependent of "apples" (as in the basic layer), but an additional dependency link has been added, making "oranges" an immediate *obj* dependent of the main verb ("like") as well and turning the dependency structure from a tree into a graph. In a similar fashion, additional links are added for the phenomena of raising/control and relative clauses (Schuster and Manning, 2016).

The enhanced representation also includes lexical material from the sentence in certain dependency labels; this applies to prepositional modifiers (*nmod, obl*) and conjuncts (*conj*). For an example, see Figure 3.8: Here, the *obl* link between "went" and "school" is augmented with the lexical material "to," including the preposition within the dependency

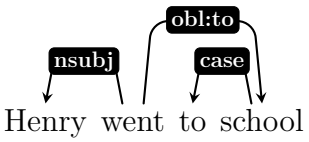



Figure 3.8.: Enhanced UD representation of the sentence "Henry went to school." Note the lexical material ("to") included as a subtype in the *obl* dependency.

label.

In addition to these main differences, Enhanced UD also introduces several other modifications of the UD annotation scheme, including for partitives and light noun constructions. For more details, see Schuster and Manning (2016).

**Enhanced UD and other graph-based meaning representations.** Aiming to move annotations closer to "meaning" on the meaning–form scale (Droganova and Zeman, 2019), Enhanced UD can be seen as part of a larger effort to annotate sentences with graph-based representations of their meaning. Other frameworks with the same goal include Abstract Meaning Representation (AMR; Banarescu et al., 2013), Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013), Universal Decompositional Semantics (UDS; White et al., 2016) and broad-coverage Semantic Dependency Graphs (SDP; Oepen et al., 2016). Enhanced UD differs from these frameworks due to its very close integration with the syntactic annotation of a sentence; at the same time, it was found to outperform the basic UD representation on a number of downstream tasks (Schuster et al., 2017).

### 3.3.1. Extracting Enhanced UD from Basic UD

Due to the effort and cost associated with manual annotation, much prior work has focused on automatically retrieving enhanced dependencies from the basic layer for various languages. Prior to the introduction of UD, Nyblom et al. (2013) used an SVM-based approach to retrieve enhanced Stanford dependencies for the Finnish-TDT treebank. They find that their approach works well for gold basic dependencies, but less so for automatic parses. A rule-based converter for English UD was first proposed by Schuster and Manning (2016) together with their initial formulation of the Enhanced UD layer. Nivre et al. (2018) compare the above two approaches for enhancing Italian and Swedish, finding that both are adequate for creating enhanced treebanks.

A body of work exists on graph-rewriting systems, which apply transformations to

graph structures based on formally defined rules/patterns (Bonfante et al., 2018). As such, they can also be used for rule-based conversion of basic to Enhanced UD. For example, Candito et al. (2017) use two graph-rewriting systems (Ogre, Ribeyre et al. (2012); and Grew, Guillaume et al. (2012)) to create Enhanced dependencies for two French treebanks, finding good accuracy in a manual evaluation.

More specific works on modeling certain phenomena within Enhanced UD further include automatically reconstructing elided predicates in sentences with gapping constructions (Schuster et al., 2018) and the modeling of coordinate constructions (Grünewald et al., 2021c, see Chapter 4).

### 3.3.2. Parsing into Enhanced UD

Another line of work examines the task of parsing raw text directly into Enhanced UD graphs. As Enhanced UD structures are not generally constrained to trees, parsing them constitutes a different problem with its own set of challenges.

Following a series of shared tasks focusing on basic UD (Zeman et al., 2017, 2018), the 2020 and 2021 IWPT Shared Tasks (Bouma et al., 2020, 2021) address the prediction of Enhanced UD from raw text. While some participating systems directly parse enhanced dependency graphs using a graph-based parser (e.g. He and Choi, 2020; Wang et al., 2020), others parsed into standard UD before applying hand-written enhancement rules (e.g., Dehouck et al., 2020; Attardi et al., 2020). Heinecke (2020) employs a rule-based approach to convert from basic to enhanced UD and then optimize the result using a classifier. The overall winner TurkuNLP (Kanerva et al., 2020) transforms enhanced UD into a tree format and then makes use of the off-the-shelf UDify parser (Kondratyuk and Straka, 2019).

With our contributions (Grünewald and Friedrich, 2020a; Grünewald et al., 2021b), we participated in both editions of the IWPT Shared Task with our STEPS dependency parser (see Chapter 7), adapted to produce Enhanced UD graphs in an end-to-end fashion. Our methodology and results are described in Chapter 8.

# 4. Coordination Constructions in English Enhanced UD: Analysis and Computational Modeling

## 4.1. Introduction

As described in Chapter 3, UD provides two levels of analysis: Basic and enhanced dependencies. One of the most frequent phenomena addressed by Enhanced UD is coordination: In the English Web Treebank (EWT), more than 15% of all sentences contain conjoined verbs; hence, a good representation of coordination clearly is crucial for downstream tasks.

Coordinate Constructions in UD are represented using the *conj* relation, with the first conjunct being the head to which all dependencies of the phrase are attached (see Figure 4.1). In the basic layer, all governors and dependents of a conjoined phrase are attached to the first conjunct. In the enhanced layer, relations are propagated to the dependent if suggested by the semantics of the sentence.[1] For example, in Figure 4.1, the enhanced layer explicitly captures that the arguments of the predicate "wrote" also fill the corresponding slots of "published," which is highly relevant for natural language understanding tasks.

In many cases, enhanced representations can be derived from the gold basic layer in a rule-based fashion. Currently available English Enhanced UD treebanks have been created by applying such a **rule-based converter (Schuster and Manning, 2016).** However, to the best of our knowledge, there are no large studies regarding their correctness and completeness.

Focusing on precision, Schuster and Manning's (2016) converter only propagates core arguments. In this chapter, we take a complementary approach, performing a large-scale annotation study in order to determine which set of links *should* be propagated from a

[1] `https://universaldependencies.org/u/overview/enhanced-syntax.html`

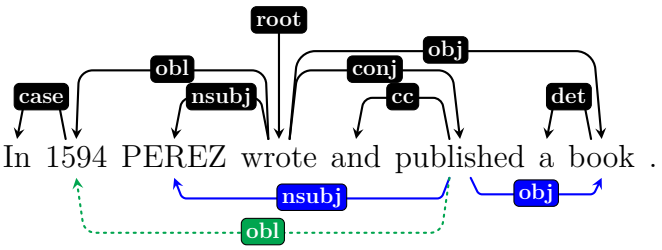



Figure 4.1.: UD **basic** (top) and **enhanced** (bottom) dependencies. **Green dotted link:** proposed addition.

semantic perspective. On a new dataset of 1,417 sentences from the EWT containing conjoined verbs, we verify and if necessary modify/extend the links involved in coordinate constructions. We argue that adjuncts such as adverbial PPs (labeled *obl* in UD) should in fact be propagated at times, e.g., in Figure 4.1, the additional (green dotted) link that we propose to add facilitates answering questions like "When was the book published?". To the best of our knowledge, our work constitutes the first large-scale annotation effort of this kind.

On the basis of our new, manually annotated dataset, we make the following contributions towards answering our main research questions. First, w.r.t. **RQ1 (What is the quality of existing UD datasets?)**, we estimate the degree of correctness and completeness of Schuster and Manning's (2016) rule-based converter and existing treebanks. We find that the converter usually generates correct graphs when applied on gold basic trees, with some notable exceptions involving non-parallel syntactic constructions (e.g., conjuncts having different voice or mood). In addition, the converter does not propagate links correctly in presence of multiple interacting conjunctions. Our inter-anntotator agreement study shows high overlap for propagation decisions, with F1 between pairs of annotators of about 0.9 on average and around 0.75 for obliques.

Second, we address **RQ2 (How can we improve and extend treebanks in an effective way with limited manual resources?)** by investigating approaches for creating high-quality treebanks for Enhanced UD from gold basic dependencies, again focusing on coordinate constructions. Based on the findings of our corpus study, we improve the rule-based converter by Schuster and Manning (2016). We also compare machine-learning (ML) based conjunction propagation classifier in the form of (a) SVM-based classifiers as previously used for Finnish, Swedish and Italian (Nyblom et al., 2013; Nivre et al., 2018), and (b) a novel neural approach integrating tree- and RoBERTa-based features. We find that all systems mostly rely on tree-based features, but contextual embeddings also provide useful information. Performance on propagation decisions has

promising F1 around 0.9, already similar to human agreement. ML-based classifiers outperform the rule-based converters on the EWT test set.

Third, touching upon **RQ5 (How can neural dependency parsers be adapted for Enhanced UD?)**, we compare methods for extracting propagated dependencies in an automatic parsing setting. The currently predominant approach is to run a basic-layer tree parser and then the same converter that has been used for gold standard construction. We propose to use a neural graph-parser based edge predictor with an architecture similar to Dozat and Manning (2018) instead, and show that this approach outperforms pipelines by around 9 points F1 on propagating links in conjunctions.

In sum, our contributions in this chapter include: (1) a manually curated large-scale dataset of 1,417 sentences addressing semantically motivated correct and complete conjunction propagation in Enhanced UD; (2) the proposal of novel neural approaches to conjunction propagation; and (3) experimental evidence that these models outperform rule- and pipeline-based approaches in both gold standard treebank enhancing and automatic parsing settings. To the best of our knowledge, our work constitutes the first principled comparison of various approaches to propagating conjunctions in Enhanced UD on manually corrected gold standard data for English.[2]

## 4.2. Related Work on Coordinate Constructions in Enhanced UD

In this section, we briefly outline related work on the handling of coordinate constructions in Enhanced UD and closely related dependency frameworks.

The task of designing **conjunction propagation classifiers** has been tackled by previous work: Nyblom et al. (2013) present an SVM-based approach for enhancing Finnish syntax trees. They observe high performance on conjunction propagation when operating on gold basic trees, but markedly worse results when using automatic parser output. Nivre et al. (2018) evaluate a similar approach for Swedish and Italian. In this chapter, we show that their approach also works well for English, and extend it with neural models and contextualized word embeddings. Simi and Montemagni (2018) de-lexicalize their rule-based converter developed for Italian, showing that their language-independent system also correctly produces most of the propagations for English. However, they

[2]The text of this chapter is based on Grünewald et al. (2021c). Prisca Piccirilli performed the annotation work for our manual corpus correction and collaborated in error analysis, while I designed, implemented, and evaluated our converter and parser experiments.

evaluate on EWT, which is itself the result of a rule-based system. In contrast, we evaluate on manually checked gold data.

Schuster and Manning (2016) present an algorithm for creating enhanced dependencies automatically based on the basic layer. While it propagates links with high precision, it propagates *only* core arguments by design (see Sec. 4.2.1). In addition, it is highly reliant on correct basic dependencies (see Sec. 4.5). As it is most closely related to our work, we describe this converter's functionality for propagating coordination links in further detail in Sec. 4.2.1.

For the related task of dealing with **gapping** constructions such as "Paul likes coffee and Mary tea," Schuster et al. (2018) reconstruct elided predicates by first parsing into an intermediate representation and then applying either a rule-based or an ML-based algorithm to copy over lexical material. We here focus on dependency propagation and operate on gold tokens as annotated in the Enhanced UD treebanks, which already include traces. Other related work exists in the area of manual and rule-based error correction on UD treebanks (Wisniewski, 2018; Alzetta et al., 2018).

The problem of predicting enhanced dependencies directly from raw text is closely related to **end-to-end Enhanced UD parsing,** which has been addressed by the IWPT 2020 and IWPT 2021 Shared Tasks (Bouma et al., 2020, 2021). From a modeling point of view, our work in this chapter is closely related to our general work on Enhanced UD parsing using our STEPS parser, with which we also participated in the aforementioned Shared Tasks (see Chapter 8).

In addition, much work exists on **semantic dependency parsing** (SDP, Oepen et al., 2014, 2015; May and Priyadarshi, 2017). These works relate to Enhanced UD as their goal is also to predict lexically anchored graph structures representing the meaning of a sentence. However, they differ from UD in that the respective formalisms represent meaning less close to syntactic structure, thus not requiring propagation.

### 4.2.1. Detailed Converter Description

After having described general related work on coordination in Enhanced UD, we next describe the converter by Schuster and Manning (2016) in futher detail. While their work introduces the Enhanced UD formalism for English and they provide high-level descriptions of propagation rules for coordination, they do not give details on the exact steps that their implementation of a rule-based system performs in order to convert basic UD trees into enhanced UD graphs. What follows is a low-level description of the

conjunction propagation functionality of their converter, based on the official code.[3]

For each *conj* relation, the converter decides whether links ending or starting at the conjunction head (`gov`) should be propagated to the conjunction dependent (`dep`):

1. Governors of `gov` are always propagated to `dep`, unless the relation belongs to a set of relation types explicitly treated as exceptions (e.g., *vocative*, *discourse*, or *root*).[4]
2. Dependents of `gov` are propagated to `dep` as follows:
   a) If the dependent is attached via *nsubj* or *csubj*, it is only propagated if `dep` does not already have a subject. If `dep` has an *aux:pass* dependent, the relation is propagated as *nsubj:pass* / *csubj:pass*.
   b) If the dependent is attached via a non-subject core relation (*obj*, *iobj*, *ccomp*, or *xcomp*), it is propagated if and only if it comes after `dep` in the linear order of the sentence.
   c) Non-core dependents (such as *obl*) are never propagated.

The algorithm is able to handle many syntactically ambiguous cases, provided the underlying basic dependencies have resolved the ambiguity correctly. Consider the sentence "She was reading or watching a movie." If "movie" is correctly attached as an object of the conjunction dependent "watching," it will not be propagated to "reading" in the enhanced representation.

## 4.3. Coordinate Constructions Dataset

In this section, we describe our creation of our manually created dataset and analyse the results.

### 4.3.1. Data

Our dataset consists of 1,417 sentences collected from EWT,[5] containing data from five genres of web media (weblogs, newsgroups, emails, reviews, and Yahoo! answers).[6] The basic dependencies of this UD gold standard have been derived from the original Stanford

[3] `https://github.com/stanfordnlp/CoreNLP/blob/master/src/edu/stanford/nlp/trees/ud/UniversalEnhancer.java`
[4] The full list of exceptions is: *dislocated*, *vocative*, *discourse*, *parataxis*, *list*, *orphan*, and *conj*.
[5] Linguistic Data Consortium LDC2012T13.
[6] `https://universaldependencies.org/treebanks/en_ewt/index.html`

| | conj. sentences | edited |
|---|---|---|
| **train** | 1,926 | 999 |
| **dev** | 222 | 222 |
| **test** | 196 | 196 |
| **total** | 2,344 | 1,417 |

Table 4.1.: Coordinate constructions dataset statistics. **conj. sentences**: sentences in EWT containing verb phrase conjunctions; we edited 60% of these.

dependencies (De Marneffe and Manning, 2008) and were then hand-corrected. The enhanced layer has been created using the automatic converter (see Sec. 4.2.1). We retrieve all sentences containing at least one instance of a *conj* link between two verbs.

More than 15% of all sentences in EWT contain conjoined verbs. Out of these sentences, we edit all sentences of the dev and test sets, and 999 sentences of the training set, amounting to more than 60% of all relevant sentences in EWT (see Table 4.1). The careful curation of each sentence took around 10 minutes on average, amounting to a total annotation effort of around 240 hours.

We exclude 18 sentences when reporting our statistics: In 12 cases, the *conj* relation is annotated wrongly in the basic layer and six sentences contain syntactically highly non-standard English.[7]

### 4.3.2. Annotation Methodology

The manual corrections of the treebank were performed by a French native speaker with an extensive background in linguistics. The annotation project involved regular discussions among all authors to decide on uncertain cases and to ensure consistency. Additionally, in case of doubt, an English native speaker with an extensive linguistics background was consulted. Dependencies were checked carefully sentence-wise using the ConLL-U-Editor tool (Heinecke, 2019). If necessary, the full document was consulted to make sure interpretations were correct in context.

**Annotation Guidelines.** We verify and modify all links involved in coordinate constructions including conjoined verbs, but also noun or adjectival phrases. First, we make sure that the automatically constructed enhanced representations adhere to the

[7]For example: "i want to be able to use it in my car, out n about etc...i guess like an iphone, but thats later on and ,i know what they are so no suggestions on just goin out to buy one im talking about right now just for an ipod??" (EWT dev set)

official guidelines for Enhanced UD (see Sec. 3.3), propagating heads and dependents of conjuncts if the interpretation of the sentence suggests additional syntactic relations between words.

As each verb may also have its own complements, this task requires a semantic interpretation leveraging context and knowledge about selectional preferences. If an ambiguity has already been resolved in the basic layer,[8] we follow this interpretation unless obviously incorrect. Second, we also propagate non-core dependents such as *obl*, *advcl* and *advmod* if suggested by semantics, an annotation task similar to prepositional phrase attachment resolution. We only propagate such links if the adjunct clearly modifies each conjunct (as in Figure 4.1). Finally, we extend the attachment of relative pronouns (*ref*) to all antecedents if involved in coordinations. We focus on propagating dependencies between content words, not propagating relations such as *aux* or *cop*, which could be handled as traces.

**Inter-annotator agreement study.** We sampled 100 sentences, half of them from cases where the primary annotator had judged the original version to be correct, and half of them cases that included modifications. This sample was blindly re-annotated by two secondary annotators, both German native speakers with an extensive computational linguistics background.

For computing inter-annotator agreement, we consider only links that are part of the enhanced layer, but not the basic layer. For each annotator, we count for each label how often it occurs as an incoming or outgoing relation of a conjunct (columns labeled with the annotator's ID). Formally, the set $E^l_A$ is the set of enhanced-layer edges that are (i) not present in the basic layer and (ii) involved in conjunctions as incoming or outgoing links of the conjuncts, with label $l$ marked by annotator $A$. We also count the overlap of links for pairs of annotators. Using these counts, we then compute precision, recall and F1, treating one annotator as the system and one as the gold standard. For instance, when treating A as the gold standard and B as the system, this leads to:

$$Precision_{BA} = \frac{|E^l_A \cap E^l_B|}{E^l_B} \tag{4.1}$$

$$Recall_{BA} = \frac{|E^l_A \cap E^l_B|}{E^l_A} \tag{4.2}$$

[8] For example, in "She was reading or watching a movie," "movie" is attached to the second conjunct "watching" in the basic layer, hence resolving the syntactic ambiguity.

| | A | B | C |
|---|---|---|---|
| **A** | - | 90.1 | 94.9 |
| **B** | 95.2 | - | 97.2 |
| **C** | 80.5 | 77.9 | - |

Table 4.2.: Inter-annotator agreement on propagated links for 100 sentences: **precision** when treating the row annotator as gold standard (or, equivalently, **recall** when treating the column annotator as gold standard).

Note that when reversing this order, P and R are simply reversed, F1 stays the same.

Table 4.2 shows agreement in terms of precision and recall on the set of dependencies resulting from conjunction propagation, i.e., the links involved in conjunctions that are present in the enhanced layer but not in the basic layer. Agreement is generally high, particularly between annotators A and B. Annotator C was more conservative in propagating links, especially in generally ambiguous cases. However, the links that C does propagate are also propagated by A and B. Pairwise agreement was high on *nsubj*, *obj* and *xcomp*. Modifier clauses (*acl*, *advcl*) and adverbials (*advmod*) were common sources of disagreement, indicating the more ambiguous nature of these propagations.

### 4.3.3. Analysis and Discussion

In this section, we analyse and discuss the modifications made to the original treebanks.

**Quantitative Analysis of Changes.** Table 4.3 presents the numbers of dependency relations that have been added and removed in coordinate constructions in the enhanced layer. More specifically, we consider only the set of links not present in the basic tree and count modifications regarding links starting or ending at conjuncts. Counts for coarse-grained labels (e.g., *nmod*) include all subtypes (e.g., *nmod:for*) not explicitly listed in the table.

Furthermore, we correct some additional errors (not necessarily related to coordinations) to the original treebank. The total numbers of changes are given in Table 4.4.

During our manual correction of the treebank, around 15% of the total enhanced links involved in conjoined phrases were added and about 3% were removed. This confirms that the converter by Schuster and Manning (2016) is optimized for precision rather than recall, though our additions of course include labels that the converter does not address. Note that in these cases, removed relations in Table 4.3 are caused by fixes regarding attachment in the basic layer, whose errors had been propagated to the enhanced layer.

| label | #added | #removed | #sents | #total |
|---|---|---|---|---|
| acl | 14 | 4 | 12 | 68 |
| acl:relcl | 13 | 3 | 9 | 190 |
| advcl | 32 | 3 | 31 | 167 |
| advmod | 46 | 2 | 35 | 6 |
| amod | 19 | 2 | 14 | 7 |
| ccomp | 9 | 10 | 11 | 50 |
| nmod | 32 | 0 | 23 | 102 |
| nmod:poss | 11 | 0 | 10 | 18 |
| nsubj | 160 | 30 | 118 | 249 |
| nsubj:pass | 8 | 18 | 25 | 225 |
| nsubj:xsubj | 22 | 10 | 17 | 1688 |
| obj | 9 | 5 | 12 | 71 |
| obl | 72 | 0 | 51 | 150 |
| ref | 12 | 1 | 8 | 61 |
| xcomp | 7 | 5 | 10 | 8 |
| **all** | 466 | 93 | 386 | 3060 |

Table 4.3.: Statistics of modifications made to 1,399 sentences of the EWT. **#sents** reports the number of sentences in which the respective reported changes were made, **#total** reports the number of occurrences of the label in the enhanced layer of the original treebank.

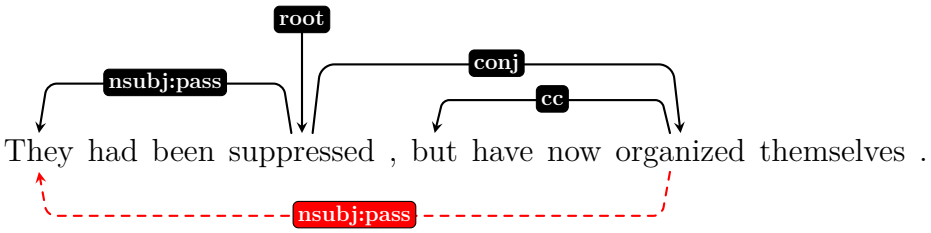
In total, we fixed errors in 57 sentences in the basic layer. In 42 of these, this led to changes in the enhanced layer.

**Linguistic Analysis of Changes.** One systematic error involves **links to subjects in passive constructions**: 18 out of 225 *nsubj:pass* links were actually wrongly propagated. All of them have been changed to *nsubj*. The reason is that the converter automatically propagates an *nsubj:pass* link if the first conjoined verb is in the passive form, as, e.g., in "These Shiite movements had been suppressed by Saddam Hussein's regime, but have now organized and armed themselves" (see Figure 4.2a). Another common error (occurring 12 times) is the propagation of the first conjoined verb's subject to the second verb, even though the latter is in **imperative mood**, as, e.g., in "I think it was the Lincoln Square area but don't quote me on that" (see Figure 4.2b).

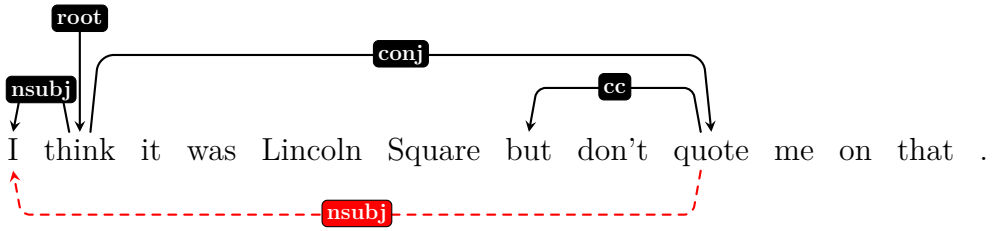
In sentences containing **multiple coordinate constructions**, such as "Dr. Fortier and his girlfriend lashed two canoes together and paddled eight kilometres along the Soper River," *nsubj* links should be present in the enhanced layer between both conjuncts of the subject noun phrase and both verbs. However, in the original treebank, the second subject conjunct was never propagated to the second verb (see Figure 4.2c). Similarly, we also added many relations in cases of **nested coordinations** as in "These Shiite

| label | #added | #removed | #sents | #total |
|---|---|---|---|---|
| acl | 18 | 5 | 14 | 749 |
| acl:relcl | 15 | 5 | 9 | 5086 |
| advcl | 39 | 11 | 36 | 2055 |
| advmod | 46 | 4 | 37 | 4426 |
| amod | 20 | 3 | 15 | 3570 |
| case | 2 | 1 | 2 | 6415 |
| ccomp | 12 | 16 | 13 | 1003 |
| det | 2 | 1 | 2 | 6448 |
| nmod | 32 | 2 | 24 | 3279 |
| nmod:poss | 12 | 1 | 10 | 7485 |
| nsubj | 194 | 41 | 144 | 7815 |
| nsubj:pass | 9 | 21 | 25 | 1381 |
| nsubj:xsubj | 35 | 20 | 31 | 8758 |
| nummod | 3 | 0 | 3 | 814 |
| obj | 10 | 10 | 14 | 4673 |
| obl | 78 | 5 | 56 | 5478 |
| punct | 3 | 3 | 3 | 9508 |
| ref | 21 | 2 | 16 | 348 |
| xcomp | 14 | 6 | 12 | 758 |
| **all** | 565 | 157 | 466 | 80049 |

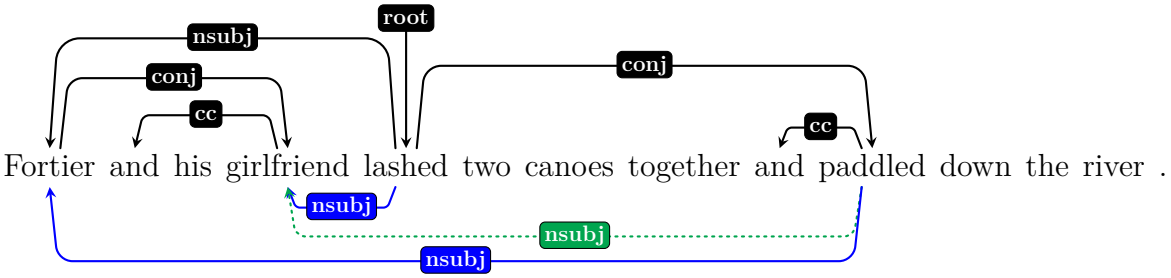
Table 4.4.: Statistics of modifications made to 1,417 sentences of the EWT, including both basic and enhanced layer. (Compare to Table 4.3, which includes only changes related to conjuncts.) **#sents** reports the number of sentences in which the respective reported changes were made, **#total** reports the number of occurrences of the label in the original treebank.




(a) Passive voice: *nsubj:pass* links should not be propagated verbatim if the second conjunct is in active voice.




(b) Imperative mood: *nsubj* links should not be propagated if the second conjunct is in imperative mood.




(c) Multiple coordinate constructions: *nsubj* links should be propagated between the second conjuncts of each coordination.

Figure 4.2.: Systematic errors found in the automatic propagation of dependencies. (Only coordination-relevant links are depicted.) **Red dashed link:** Incorrect propagation or incorrectly labeled propagation. **Green dotted link:** Missing propagation.

movements had been suppressed by Saddam Hussein's regime, but have now organized and armed themselves." The second conjunct of the conjoined verb phrase is a conjoined verb phrase itself, but the *nsubj* link to "armed" was missing. In total, 194 sentences contain several coordinations, and we modified 92 of them. This phenomenon also accounts for 45 of the added *nsubj* links.

Some originally missing propagations concern **adjectival and adverbial modifiers** (*acl, amod, advcl, advmod*), which are known to be ambiguous cases. In "Handwritten notes and files on a laptop were seized," the adjective "handwritten" clearly modifies the first conjunct "notes" only, but in "Several Indian scholars and politicians have been ready to say and endorse anything," the propagation of "several" and "Indian" was added during our modifications. These cases involve world knowledge that the converter currently does not handle.

Finally, consider the sentence "We recognize that the state may not require religious groups to officiate at, or bless, same-gender marriages." Both conjuncts take "marriages" as their argument, but as an *obl* and as an *obj* relation, respectively. The resolution of such non-parallel constructions requires detailed subcategorization information.

## 4.4. Modeling

We are now moving from manual corpus analysis to computational work and describe three approaches to generating links propagated due to coordination: (1) an improved version of an existing converter (Sec. 4.4.1); (2) ML-based propagation classification operating on basic trees (Sec. 4.4.2); and (3) a graph-parser based approach for directly predicting edges between tokens (Sec. 4.4.3). While (1) and (2) may be used to construct "silver standard" enhanced UD graphs from gold trees, (3) is applicable in the automatic parsing setting only.

### 4.4.1. Modifications to Rule-based Converter

Based on the error analysis in Sec. 4.3.3, we modify the rule-based converter by Schuster and Manning (2016) as follows. In order to fix errors related to subject propagation in passive and imperative constructions, we take the conjunction dependent's morphological features into account. In the gold standard, the *Voice* feature is considered to be *active* by default. Hence, if the conjunction dependent does not have a *Voice* feature or is explicitly marked as *active*, an *nsubj:pass* dependency will be propagated as *nsubj*. Similarly,

if it has the feature *Mood=Imp*, an *nsubj* link will not be propagated. Our second modification propagates common adjuncts of verbs as well (*obl*, *advmod*, and *advcl*). We maintain the rule from object propagation that a dependency is only propagated if the dependent comes after the potential target in the sentence. Finally, to handle multiple and nested coordinations, we iterate the converter's conjunction propagation function until the dependency graph does not change any more. This allows dependencies that result from propagation to be propagated themselves, retrieving links that would otherwise be missed.

### 4.4.2. Conjunction Propagation Classifiers

The core idea of ML-based conjunction propagation classifiers is to take a basic-layer tree and to decide for each incoming or outgoing dependency of the head of a coordinated phrase whether to propagate this dependency to the other coordinated item(s). We refer to the coordinated nodes as **conjunction head** and **conjunction dependent** and to the candidate governor/dependent of the second conjunct as the propagation **target**. In Figure 4.1, these three nodes correspond to "wrote," "published" and "1954" (or "PEREZ"/"book"), respectively. The output is a binary decision whether to propagate the given dependency or not. In addition to the features described below, we always provide the candidate dependency label and direction.

**SVM-based Classifier.** We re-implement the method proposed by Nyblom et al. (2013) using scikit-learn's SVC[9] with a polynomial kernel of degree 2. The features comprise morphological information about the tokens for the conjunction head/dependent and the target, as well as structural tree features extracted from the basic-layer tree. For a full list with descriptions, see Table 4.5. In addition to the listed features, we also experimented with including lemmas and POS tags, but did not find them to be useful in our ablation experiments.

**Neural network classifier.** We pass the sentence through the transformer-based neural language model RoBERTa (Liu et al., 2019) and extract the word embeddings for the first wordpiece tokens of the conjunction head, the conjunction dependent, and the propagation target. In addition, we use equivalents of the SVM tree features using learned embeddings or one-hot encodings (see Table 4.5). The inputs are concatenated and fed to a multi-layer perceptron, which then outputs the binary decision whether to

[9] `https://scikit-learn.org`

| Feature name | Description | SVM | NN |
|---|---|---|---|
| ***Instance features*** | | | |
| dependency label | label of candidate link | one-hot | 50-dim. embedding |
| incoming/outgoing | whether the dependency being propagated is an outgoing or incoming link at the conjunction head | one-hot | 50-dim. embedding |
| ***Token features*** | | | |
| morphological features | values of the *Number*, *Person*, *VerbForm*, and *Voice* features | one-hot | – |
| contextualized word embeddings | word embeddings as generated by the RoBERTa-base model | – | 768-dim. embedding |
| ***Tree features*** | | | |
| linear dependency direction | whether the linear direction of the candidate dependency is the same as for the dependency being propagated (both-left, both-right, or differing-directions) | one-hot | 50-dim. embedding |
| existing dependency | whether the conjunction dependent already has a dependency of this type (only relevant for outgoing links) | one-hot | 50-dim. embedding |
| outgoing dependencies (head) | set of outgoing dependencies of the conjunction head | one-hot | one-hot |
| outgoing dependencies (dep) | set of outgoing dependencies of the conjunction dependent | one-hot | one-hot |
| # coord.-items | number of items in the coordination | one-hot | scalar |

Table 4.5.: Description of feature sets used in ML-based conjunction propagation models.

| | |
|---|---|
| **RoBERTa embeddings** | |
| Embeddings dimension | 1024 |
| Token mask probability | 0.15 |
| Layer dropout | 0.1 |
| Hidden dropout | 0.2 |
| Attention dropout | 0.2 |
| Output dropout | 0.5 |
| **Biaffine classifier** | |
| Hidden size | 1024 |
| Dropout | 0.33 |
| **AdamW Optimizer** | |
| Batch size | 5 |
| Learning rate | $5e^{-6}$ |
| $\beta_1$, $\beta_2$ | 0.9, 0.999 |
| Weight decay | 0.0 |

Table 4.6.: Hyperparameters for our graph-based parser.

propagate the dependency or not. The multi-layer perceptron consists of two linear layers with hidden sizes 1500 and 500 respectively. We implement the model using Huggingface's Transformers library (Wolf et al., 2020). RoBERTa weights are not fine-tuned.

### 4.4.3. Graph-Parser Based Edge Prediction

In addition to the above approaches, we also evaluate a graph-parser based approach that predicts dependencies between tokens directly, i.e., which does not rely on a basic-layer tree. Our unfactorized architecture is similar to that of our STEPS parser (see Chapter 7), i.e., our model predicts presence of edges and the corresponding labels in a single step, treating nonexistence of an edge as simply another label (∅). As we focus on the dependencies involved in conjunctions, we do not require the parser's output to constitute valid graphs. For more details on our parser architecture, see Chapter 7.

The model is simply trained to predict all link types in enhanced UD graphs. In the training section of the EWT corpus, we replace every sentence that contains a coordinated verb phrase with our manually corrected version of that sentence, or remove it from the corpus if it is one of the 927 conjunction sentences in the training section which we did not correct.

We perform only a minimal amount of hyperparameter tuning, mostly sticking with the values used by Kondratyuk and Straka's (2019) UDify parser, which our system is architecturally related to. One notable exception is the training regime, where we found low batch size and the AdamW optimizer to yield the best results. The full hyperparameter configuration can be found in Table 4.6.

## 4.5. Experiments

In this section, we describe our experiments on creating Enhanced UD representations for coordinate constructions. Analogous to Nyblom et al. (2013), we measure precision, recall and F1 on enhanced links that are the result of propagation in coordinate constructions. For all experiments, we use gold sentence segmentation and tokenization, and evaluate on our manually corrected sentences from the dev and test sets of the EWT corpus.

### 4.5.1. Gold Standard Treebank Enhancing

We first address the research question of how to best generate enhanced representations for treebanks with gold standard basic annotations. We compare the following models: (1) an "Always" baseline, which simply propagates all incoming and outgoing links from the conjunction head to the conjunction dependent(s); (2) the rule-based converter by Schuster and Manning (2016) and the variations thereof we developed inspired by our corpus study; (3) our re-implementation of the SVM-based classifier by Nyblom et al. (2013); and (4) our neural-network (NN) based classifier. The latter uses AdamW (Loshchilov and Hutter, 2019a) with a learning rate of 5e-5, a batch size of 1 and early stopping. Table 4.7 reports the results on the development and test sets of our manually verified conjunction dataset. The recall of the "Always" baseline is not at 100% because a small number of relations change their label during propagation, e.g., *nsubj*→*nsubj:pass*.

**Rule-based conversion.** We show results for successively adding components to the original converter (**RBC**). On the test set, adding propagation of non-core dependents and allowing several iterations increases recall and improves F1 by more than 2 points. On the dev set, in contrast, we do not observe these effects.[10] Adding our suggested passive/imperative fix surprisingly decreased performance. Analysis showed that the cases that our converter got wrong were caused by erroneous morphological feature annotations in the basic layer. In sum, our suggested improvements (**RBC2**) of heuristically propagating adjuncts (*obl*, *advmod*, *acl*) and allowing several resolution passes of the converter seem to improve treebank enhancing, provided that the basic layer is correct.

**ML-based conversion.** Overall, the SVM and NN models show similar performance. As they perform already close to human agreement (cf. Table 4.2), further improvement

[10]We assume that the reason is the presence of several informal-language sentences in the dev set that include multiple conjunctions (e.g., "etc. etc. etc."), whose annotation is unclear even in the basic gold standard.

| | Dev | | | Test | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| *"Always" baseline* | *23.1* | *99.6* | *37.5* | *28.0* | *99.6* | *43.7* |
| **RBC** | **94.8** | 86.4 | **90.4** | 95.2 | 76.9 | 85.0 |
| + non-core deps | 93.7 | 86.4 | 89.9 | 94.9 | 79.7 | 86.7 |
| + iteration (**RBC2**) | 90.1 | 86.8 | 88.4 | 93.9 | 81.5 | 87.2 |
| + passive fix | 91.7 | 85.3 | 88.4 | **95.7** | 78.6 | 86.3 |
| **SVM** | 87.6 | 87.9 | 87.8 | 93.4 | 85.4 | **89.2** |
| - tree features | 75.5 | 78.0 | 76.8 | 76.5 | 63.7 | 69.5 |
| - token features | 86.3 | 87.5 | 86.9 | 92.3 | 85.1 | 88.5 |
| **NN** | 87.0 | 87.9 | 87.4 | 92.0 | **85.8** | 88.8 |
| - tree features | 87.1 | 86.4 | 86.8 | 88.0 | 78.6 | 83.1 |
| - token features | 87.3 | **88.3** | 87.8 | 92.2 | 84.3 | 88.1 |

Table 4.7.: **Predicting relation propagation for coordinate constructions on gold basic trees**. Precision, recall and F1 on propagated relations in the predicted vs. gold dependency graphs in our manually verified conjunction dataset. The gold dev and test sets contain 273 and 281 instances, respectively.

may actually indicate overfitting. On the test set, the ML-based methods outperform the heuristic rule-based methods, surpassing the original converter by over 4 points F1. We conclude that learning structural rules based on actual gold standard data is more effective than hand-designing them. Differences on the dev set are less pronounced despite models being optimized on this data, again hinting at some qualitative differences between the two sets.

In order to determine which sources of information are most relevant, we perform ablation experiments for both classifiers. The features representing the candidate dependency label and the direction of the link are essential and kept in each case. Both the SVM and the NN classifiers draw most of their information from tree-based features. This effect is particularly pronounced for the SVM classifier, where performance drops by 10 to almost 20 points F1 when ommitting these features. The NN classifier's performance does not deteriorate as strongly under the same condition, indicating that some syntactic information can also be retrieved from contextualized word embeddings (see e.g., Tenney et al., 2019a). Nonetheless, in most experiments, adding token features improves performance slightly, showing that they do contain important information for propagation decisions.

| | Dev | | | Test | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| Stanford+RBC | 70.8 | 63.0 | 66.7 | 56.5 | 47.7 | 51.7 |
| Stanford+RBC2 | 68.7 | 65.2 | 66.9 | 56.2 | 50.2 | 53.0 |
| Stanford+SVM | 64.3 | 65.2 | 64.7 | 54.7 | 49.8 | 52.1 |
| Stanford+NN | 64.3 | 65.2 | 64.7 | 54.4 | 50.2 | 52.2 |
| UDify+RBC | 72.8 | 67.8 | 70.2 | 71.8 | 58.0 | 64.2 |
| UDify+RBC2 | 71.9 | 68.5 | 70.2 | 75.0 | 61.9 | 67.8 |
| UDify+SVM | 70.6 | 68.5 | 69.5 | 70.9 | 59.1 | 64.5 |
| UDify+NN | 69.9 | 68.9 | 69.4 | 70.4 | 60.1 | 64.9 |
| GBP (orig. data) | **83.1** | 74.0 | 78.3 | **86.1** | 66.2 | 74.8 |
| GBP (our data) | 82.3 | **75.1** | **78.5** | 82.5 | **68.7** | **75.0** |

Table 4.8.: **Predicting relation propagation for coordinate constructions on parser output**. Otherwise same evaluation setup as in Table 4.7. GBP = our graph-based parser.

### 4.5.2. Propagating Conjunction Links in Automatic Parsing Setting

For the scenario of parsing from raw tokens, we compare two state-of-the-art parsers, StanfordNLP (Qi et al., 2018) and UDify (Kondratyuk and Straka, 2019), combined with the rule-based converter or ML-based conjunction propagators, and our graph-parser based edge predictor **(GBP)**. The latter is trained on the subset of training sentences that either do not contain coordinated verb phrases or that were corrected by us.

Results for these experiments can be found in Table 4.8. The impact of the quality of the parsed basic dependencies is evident: Results are much better for the UDify parser (LAS F1 of 89.4 for basic dependencies on the EWT dev set) than for StanfordNLP (LAS F1 of 87.4). In the automatic setting, our heuristic extensions improve results compared to using the original converter, and there is no decrease in F1 on dev. As in the gold standard settings, ML-based extensions improve upon RBC on test, but not dev. Of the systems based on basic-layer tree parsers, RBC2 works best. However, all pipeline systems show rather poor performance at or below an F1 of 70. Our graph-parser based edge predictor achieves by far the best results, outperforming all other models by a margin of over 7 points F1. This shows that in an automatic setting, most robust results are achieved by directly inducing dependency links between tokens, modeling conjunction only indirectly.

To estimate the impact of our corrections to the gold standard, we also train the graph parser on uncorrected data. The model trained on the corrected data has higher recall, but lower precision. This is expected to some extent as we introduce semantically

motivated propagations of adjuncts, and we suspect that they may require a larger training set.

### 4.5.3. Discussion

The main insights comparing our experiments in the gold standard vs. the automatic parsing setting are as follows. Overall, our heuristic extensions for the rule-based converter are beneficial in both settings. In the gold setting, ML-based extensions lead to higher accuracy; when applying them on noisy parser output, they do not work well. However, using *one* end-to-end machine-learning model directly to generate enhanced representations for conjunctions outperforms the pipeline version. A possible reason for this might be that these models were all developed on gold data, while the graph-based parser does not rely on potentially wrong structural tree features and is also able to use internal confidence information for edges. Another advantage of the end-to-end model may stem from the fact that its training allows to leverage semantic information from training data of a larger number of dependency links, i.e., including those not occurring in coordinate constructions. This points to a promising future research direction, i.e, generating additional semi-artificial training data for conjunction propagation.

## 4.6. Conclusion and Outlook

We have presented a large-scale manually curated **dataset for conjunction propagation** in English. In contrast to previous work focusing on high-precision rule-based propagation, we propagate links in all cases that semantically suggest argument or adjunct sharing. In the gold standard treebank enhancing setting, we found **ML-based models** to outperform the de-facto standard rule-based converter by learning to exploit mostly structural features. However, one of our main insights is that neither rule-based nor ML-based classifiers work well on noisy parser output precisely because of this reliance on structural information. We propose to use a graph-parser based edge predictor instead and show that it outperforms pipeline-based models by a large margin. Our model reaches F1 scores between 75 and 78 with a precision of more than 82 %, a level of performance that may already be useful in downstream tasks.

Our models could be used for creating high-quality enhanced-level representations of conjunctions for the remaining English data, and could thus help in a UD community effort to continuously **improve the UD treebanks**. Future work also includes the

study of conjunction propagation methods for **further languages,** which we address for the case of German in Sec. 5.2. Our in-depth study on English data provides several insights that we expect to be transferable cross-lingually. First, conjunction propagation can to some extent be addressed using heuristic rules, but capturing the full semantic nature of the task requires manual annotation. Second, given appropriate training data, our machine-learning based approaches are also applicable to other languages.

In addition, it would be interesting to see if manually annotated data for coordinate constructions may be useful in natural language understanding tasks such as natural language inference (NLI). This is especially true for "stress test" datasets such as ConjNLI (Saha et al., 2020), which are designed to specifically test models' capabilities to process coordination.

Finally, as morphological features are generally important for this task, improving their automatic prediction (see e.g., Ramm et al., 2017; Myers and Palmer, 2019) as well as UD's gold standard seems to be a promising way to go. Our work has demonstrated the value of a linguistically motivated corpus study of a syntactic-semantic phenomenon, and shown that given manually curated data, rules for conjunction propagation can be learned effectively.

# 5. Extending and Improving German UD Treebanks

The Universal Dependencies project aims to provide high-quality syntactic annotations not just for English, but for as wide a range as possible of human languages. However, treebanks for many languages – even high-resource languages with many speakers – continue to be noisy and/or incomplete (e.g., because they result from automatic conversion from a different dependency framework).

In this chapter, we widen the scope from the English language and address these issues for the case of German UD treebanks, relating them to our research questions **RQ1**–**RQ3**, i.e.:

- **What is the quality of existing German UD treebanks? (RQ1)**
- **How can we improve and extend German treebanks in an effective way with limited manual resources? (RQ2)**
- **What is the effect of improvements to German UD treebanks? (RQ3)**

The chapter is divided into two main sections. Sec. 5.1 describes our efforts in unifying the treatment of preposition-determiner contractions across German treebanks, with the overarching goal of assessing annotation consistency across treebanks and methods for improving it. To this end, we analyze their frequency and handling, propose a rule-based algorithm for canonicalizing them, and evaluate the effect of our changes to parser performance.

Sec. 5.2 outlines our work on extending German treebanks with enhanced dependencies, examining whether a rule-based extension scheme (as has been used to create enhanced dependencies for English; Schuster and Manning, 2016) also works for a different language. We propose hand-written rules for extracting enhanced dependencies from existing basic trees and evaluate their performance both when operating on gold trees as well as parser output.

Our main findings are that rule-based extensions and improvements of UD treebanks are generally feasible, but their quality – and thus their effect on downstream performance – depend on the nature of the problem that is being addressed. For a highly regular phenomenon like preposition-determiner contractions, a rule-based solution is straightforward and results in consistent (if modest) parsing accuracy gains. By contrast, extending an existing basic treebank with enhanced dependency via hand-written rules – which necessitates handling of complex phenomena such as coordination and relative clauses – yields more mixed results, particularly when trying to use the resulting relations to train a machine-learning based parser. This highlights the higher complexity and bigger scope of the issue.

## 5.1. Unifying the Treatment of Preposition-Determiner Contractions in German Universal Dependencies Treebanks

### 5.1.1. Introduction

The treatment of function words in Universal Dependencies, being a rather language-specific issue, is to date sometimes inconsistent even across the UD treebanks of a single language. Function words including prepositions or negation words are often contracted with other words, which requires to decide whether to keep them as a fused unit or split them up during tokenization – a non-trivial problem.[1]

The German language allows to fuse certain combinations of preposition+determiner into a single token, resulting in what UD refers to as a *multiword token*, i.e., a single token that contains more than one *syntactic word*. Examples include *zum* (= *zu dem, "to the"*) and *ins* (= *in das, "into the"*). These constructions can even be regarded as lexicalized, i.e., as belonging to the inventory of the language's lexicon (Lehmann, 2002). Expanding such contractions into several tokens does not depend on the context, hence, treating them as multi-word tokens is rather straightforward. The only caveat is assigning the correct morphological features to the determiner, but these can easily be retrieved from the head noun.

The current UD annotation guidelines for German[2] suggest treating preposition-

[1] See, e.g., `https://github.com/UniversalDependencies/docs/issues/641`
[2] `https://universaldependencies.org/de/index.html`

| Contract. | Expansion | count | % sents |
|---|---|---|---|
| im | in dem | 26236 | 12.8 |
| am | an dem | 7764 | 3.9 |
| zum | zu dem | 7584 | 3.9 |
| zur | zu der | 6149 | 3.1 |
| vom | von dem | 3404 | 1.8 |
| beim | bei dem | 2795 | 1.4 |
| ins | in das | 1422 | 0.7 |
| fürs | für das | 233 | 0.1 |
| ans | an das | 160 | 0.1 |
| übers | über das | 147 | 0.1 |
| TOTAL | | 56150 | 25.0 |

(a) German-HDT-UD

| Contract. | Expansion | count | % sents |
|---|---|---|---|
| im | in dem | 89 | 4.2 |
| zur | zu der | 44 | 2.0 |
| zum | zu dem | 27 | 1.4 |
| vom | von dem | 17 | 0.9 |
| am | an dem | 17 | 0.9 |
| ins | in das | 8 | 0.4 |
| aufs | auf das | 7 | 0.3 |
| beim | bei dem | 5 | 0.3 |
| fürs | für das | 5 | 0.3 |
| beym | bey dem | 3 | 0.1 |
| TOTAL | | 222 | 9.8 |

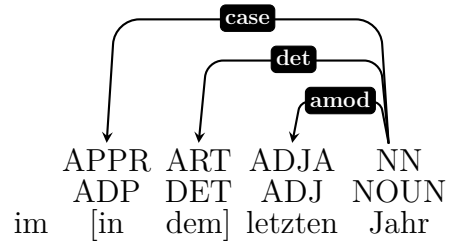
(b) German-LIT

Table 5.1.: The 10 most common contractions in two German UD corpora, as well as the total count. (The computation of "% sents" in the TOTAL row considers the fact that one sentence may contain several contractions.)

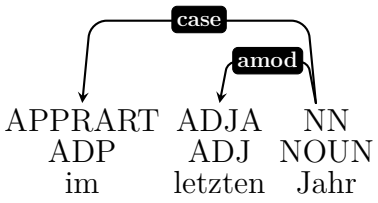
determiner contractions as multi-word tokens in the way outlined above, and their treatment is implemented accordingly in the German GSD treebank (292k tokens) as well as in German PUD (21k tokens). However, HDT-UD (Borges Völker et al., 2019), the largest German UD treebank (with 190k sentences and 3.4 million tokens currently the largest available UD treebank overall), as well as the small German-LIT treebank (40k tokens; Salomoni, 2017), do not expand these tokens. This may lead to inconsistency-based parsing errors in cross-treebank experiments or when training a parser on several treebanks.

In this section, we analyze the extent of the problem **(RQ1),** finding that in HDT-UD, 25% of all sentences contain such contractions. Our further contributions are (i) the development of a simple lookup-based script for splitting German preposition-determiner contractions into several tokens **(RQ2),** and (ii) a set of experiments showing that on the relevant sentences, the increased consistency leads to an increase in parsing accuracy by up to 0.8 points in terms of LAS F1 **(RQ3).** Hence, this chapter constitutes a case study of the improvements we can expect from careful linguistic data analysis and corpus harmonization.[3]

[3]The text of this section is based on Grünewald and Friedrich (2020b)




(a) Expanded contraction (according to guidelines).




(b) HDT-UD / German-LIT original version.

Figure 5.1.: Annotation of contractions in the phrase *im letzten Jahr* (*"in the last year"*).

### 5.1.2. Preposition–Determiner Contractions in German UD corpora

The German HDT-UD treebank (Borges Völker et al., 2019) has been automatically converted from the Hamburg Dependency Treebank (HDT; Foth et al., 2014) using a tree transducer. The text data stems from the German technical website heise.de, which contains, among others, reports about new software and hardware as well as technology-related politics. HDT uses its own dependency annotation scheme (Foth, 2006), in which, in contrast to UD, relations are headed by function words. Preposition-determiner contractions are simply marked as prepositions and indicate their complement (i.e., what would be a preposition's head noun in UD) using the relation PN.

Table 5.1(a) reports corpus statistics for the occurrences of the most frequent preposition-determiner contractions in HDT-UD. Contractions occur in 25% of all sentences, showing that treating the phenomenon in a consistent way is non-negligible. Some contractions, including *im* (= *in dem*, "*in the*"), *am* (= *an dem*, "*at the*") and *zum* (= *zu dem*, "*to the*") are extremely common; others are more colloquial and rarer in written German (e.g., *übers* = *über das*, "*over the*").

Figure 5.1 illustrates the two different ways of treating preposition-determiner contractions in German UD corpora. Version (a), introducing two trace-like tokens, is suggested by the official guidelines, and applied in GSD and German-PUD. The commonly used Stanza tokenizer (Qi et al., 2020) also employs this strategy. HDT-UD currently applies a single-token analysis (b). The contractions have their own language-specific XPOS tag `APPRART` ("preposition with article") taken from the STTS tagset (Schiller et al., 1995). Their UPOS tag in HDT is `ADP`, which further shows that the single-token analysis is inadequate: `ADP` stands for adpositions (a cover term for prepositions and postpositions), but the contractions also include a determiner in addition to a preposition. This information is lost in the single-token analysis, which attaches the contraction to its head via

*case*, circumventing *det* altogether.

The same issue exists in the small German-LIT treebank (Salomoni, 2017), which consists of short fragments of 18th century literary essays about aesthetical issues by Schlegel and Novalis, written in the then-young Hochdeutsch (modern German). Corpus statistics for German-LIT are given in Table 5.1(b).

### 5.1.3. Expanding Contractions

Based on the above discussion, we propose a simple lookup-based method for expanding preposition-determiner contractions in German UD corpora into multi-word tokens in order to make the data consistent with the UD annotation guidelines. Figure 5.2 shows our algorithm, which operates on files of the CoNLL-U format.[4] Note that we do not make changes for the token *z.* from *z. B.* = *zum Beispiel* = "*for example*," as well as in several other infrequent special cases resulting from annotation errors.

Since prepositions and determiners are closed word classes and the mapping from contractions to their expansions is unambiguous in German, we use a simple lookup table (see Table 5.1) for expansion. The full table was constructed by extracting all word forms labeled `APPRART` from the HDT-UD and LIT corpora and then using the authors' knowledge of German to assign the correct expansions. Morphological features of the syntactic words of the expansion may be ambiguous: for example, *zum* may be expanded into *zu* $dem_{Gender=Neut}$ or *zu* $dem_{Gender=Masc}$. However, we can easily derive the correct features by simply copying over the *Case*, *Number*, *Gender* features from the syntactic head of the contraction. (Note, however, that potential annotation errors are also propagated this way.)

We ensure the correctness of the output by manually inspecting some of the resulting annotated sentences as well as running the official UD validation script.[5]

### 5.1.4. Parser Evaluation

To evaluate how our changes to the corpora affect parsing accuracy, we train the state-of-the-art UDify parser (Kondratyuk and Straka, 2019) on the existing GSD corpus as well as the original and modified versions of the HDT-UD corpus, and report scores on various versions of the test sets. (LIT and PUD only provide test data.) We keep all hyperparameters the same as in the original setup, except that (a) we use the German

[4] `https://universaldependencies.org/format.html`
[5] `https://github.com/UniversalDependencies/tools/blob/master/validate.py`

For each token with index $i$: if the token has the `APPRART` XPOS tag (and no exception applies*):

1. Increase the indices of all tokens after index $i$ by 1.
2. Insert the contraction as a new multiword token with index $(i, i+1)$
3. Replace the contraction at $i$ with two new tokens: a preposition and a determiner as specified by the lookup table.
4. Attach the preposition and determiner to the contractions's head via the *case* and *det* relations, respectively.
5. Copy the contraction's head's *Case* feature to the preposition and its *Case*, *Number*, *Gender* features to the determiner.

*Exceptions include tokens that are clearly incorrectly tagged and tokens attached via the *reparandum* relation (i.e., disfluencies).

```
1-2 Im      _    _      _ _
1   In      ADP  APPR   3 case
2   dem     DET  ART    4 det
3   letzten ADJ  ADJA   3 amod
4   Jahr    NOUN NN     4 obl
5   stieg   VERB VVFIN  0 root
6   der     DET  ART    6 det
7   Umsatz  NOUN NN     4 nsubj
```

Example CoNLL-U file (output of algorithm):
*"In the last year increased the revenue"*

Figure 5.2.: Algorithm for lookup-table based preposition-determiner expansion in German UD.

BERT model by deepset[6] to initialize BERT weights; and (b) we only train on HDT-UD for 25 epochs due to the extremely large size of the corpus. We report results for gold tokens.

Table 5.2 shows parser performance on sentences containing contractions in the test sets of GSD, HDT-UD, and LIT (the latter two in both their original and modified versions, indicated by *+/-exp*). As can be seen, for the parser that is trained on the GSD corpus (in which contractions are split up), performance is higher on the modified versions of the HDT and LIT corpora than on the original versions, as would be expected due to the unified treatment of contractions. Interestingly, the same holds when training on the original version of HDT-UD. The reason for this is that the original corpus also contains many cases of non-contracted preposition-determiner combinations, enabling the parser to get expanded contractions right as well. Furthermore, scores increase on the fixed test sets because the expansion of contractions leads to more *det* relations, which are generally very easy to predict. However, training on the fixed data is still beneficial, as comparing the last two rows for each of the *+exp* test sets shows.

An analysis by label type confirms that increases in accuracy are mainly caused by *case* and *det*. For example, when training on GSD, LAS F1 of *case* increases from 96.28 ($\text{HDT}_{\text{-exp}}$) to 96.60 ($\text{HDT}_{\text{+exp}}$); *det* increases from 96.74 ($\text{HDT}_{\text{-exp}}$) to 97.72 ($\text{HDT}_{\text{+exp}}$). We also observe modest improvements in parsing accuracy on a number of other dependency labels such as *obl*, *nmod*, and *ccomp*, indicating that surrounding

[6] `https://deepset.ai/german-bert`

| | **test** | | | | | |
|---|---|---|---|---|---|---|
| ↓ **train** | $GSD_{+exp}$ | $PUD_{+exp}$ | $HDT_{-exp}$ | $HDT_{+exp}$ | $LIT_{-exp}$ | $LIT_{+exp}$ |
| $GSD_{+exp}$ | 85.78 | 85.34 | 85.70 | **86.50** | 79.76 | **80.45** |
| $HDT_{-exp}$ | 79.15 | 81.32 | 95.57 | **95.72** | 76.33 | **76.91** |
| $HDT_{+exp}$ | 79.23 | 81.41 | 95.45 | **95.73** | 76.34 | **77.11** |

Table 5.2.: Parsing performance (**LAS F1**) on test set sentences that contain contractions. *+exp* indicates that contractions are analysed as multi-word tokens in the respective corpus, *-exp* indicates that they are analysed as single tokens.

syntactic constructions also benefit from the consistent handling of contractions.

Interestingly, our results also show that the parser trained on GSD, despite not having encountered contractions during training, still attaches the vast majority of contractions in $HDT_{-exp}$/$LIT_{-exp}$ correctly. We suspect that this may be owed to BERT's ability to generalize from simple prepositions to contractions because of their similar distribution in the pre-training data.

### 5.1.5. Related Work

**Discussions within UD community.** It is an on-going discussion within the UD community how to best achieve a standardized treatment of tokenization, e.g., how to treat multi-word tokens such as *gonna* (= *going to*) in English. A current proposal[7] suggests breaking up these tokens for formal English as was done in the original Penn TreeBank annotations, but allowing different solutions for informal language such as Twitter posts. We follow this suggestion in some sense as we also create multi-word tokens for highly frequent and lexicalized preposition-determiner contractions in German.

Within the context of Surface-Syntactic Universal Dependencies (SUD; Gerdes et al., 2018), it has been proposed to treat prepositions as heads, moving away from UD's focus on content words and applying distributional criteria for the units instead. In fact, in SUD, the question of how to deal with these contractions is even more pressing: Because the preposition part of the contraction would be the head of a noun, but the determiner part would be a dependent, it is not quite clear what the overall syntactic relation of the contraction to the noun should be if left as one syntactic word.

**French amalgames.** A similar issue arises in the word segmentation of French *amalgames* (contractions). Here, the situation is slightly more complicated due to the ambiguity of

[7] `https://github.com/UniversalDependencies/docs/issues/641`

*du/des*, which may occur either as indefinite determiners as in *des enfants jouent* ("(some) kids play"), in partitive constructions such as *je bois du lait* (= *de le*, "I drink (some) milk"), or in possessive constructions such as *la lettre des filles* (*de les*, "the letter of the girls"). Currently, the major French treebanks annotate the infinite determiner *des* as DET, while they split the other two cases into two tokens. However, a context-dependent treatment of tokenization bears technical difficulties for automatic processing; a discussion by the treebank maintainers suggests to always split these contractions and annotate the indefinite determiner case using a *fixed* relation between the two components.[8] To date, this does not seem to have been implemented.

### 5.1.6. Summary and Outlook

In this section, we have carefully analysed the treatment of preposition-determiner contractions in German UD corpora. Harmonizing representations lead to increases in LAS F1 of up to 0.8, indicating that a unified treatment of these frequent construction is essential.

We here have presented a case study of how to unify word segmentation for the relatively simple case of German preposition-determiner contractions. Future work includes addressing similar phenomena in more difficult situations such as the context-dependent interpretation of French amalgames or, more generally, finding good guidelines for word segmentation and harmonizing corpora accordingly. The latter is especially tricky for informal genres where segmentation decisions seem to be a continuum.

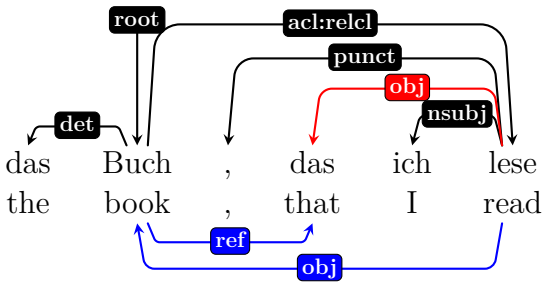
## 5.2. A Corpus Study of Using Rules to Create Enhanced Universal Dependencies for German

### 5.2.1. Introduction

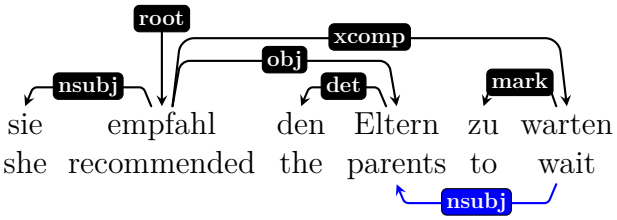
As of today, annotations for the Enhanced UD representation are still only available for a small subset of UD treebanks.[9] Many of these existing annotations have been created using rule-based systems operating on the basic layer with little to no human

[8] `https://github.com/bguil/UD-French-discussion/issues/1`

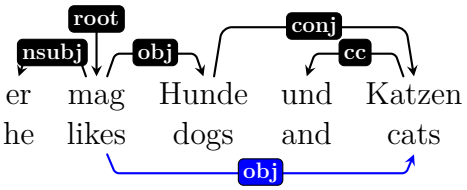
[9] The official UD website lists 41 treebanks as providing enhanced dependencies, however, manual inspection shows that in some of them the enhanced layer is identical to the basic layer. 29 treebanks for 17 languages were used for Enhanced UD parsing in the 2020 and 2021 IWPT Shared Tasks.




(a) Relative clause




(b) Raising/control




(c) Coordination

Figure 5.3.: Linguistic structures in German enhanced UD. **Blue:** Relations added in enhanced representation. **Red:** Relations removed in enhanced representation.

supervision/evaluation, or via conversion from other syntactic representations (Droganova and Zeman, 2019, see also Chapter 4).

Addressing **RQ2 (How can we improve and extend treebanks in an effective way with limited manual resources?),** in this section we take a first step towards extending Enhanced UD coverage to German treebanks. No enhanced annotations for German existed prior to our work, making this a difficult problem that cannot be tackled with standard supervised machine learning techniques. We give details on our findings when generating Enhanced UD representations from the basic layer of German UD treebanks in a rule-based way, similarly to the work of Schuster and Manning (2016, see also Sec. 4.2.1) for English. We focus on three linguistic phenomena (relative clauses, raising/control, and coordination; see Figure 5.3) and present rule-based algorithms for each phenomenon that take into account the peculiarities of the German language, which has less strict rules for word order compared to English.

We assess the accuracy of the extractions made by our system with the help of expert human annotators, evaluating system performance on both gold as well as automatically parsed basic dependencies. In the gold setting, we find that around 90 % of added relations are correct, outperforming the English-focused system by Schuster and Manning (2016). On automatically parsed text, performance drops by around 10 %, indicating a reliance on correct basic dependencies. Addressing **RQ3 (What is the effect of improvements to UD treebanks?),** we find that a graph parser trained on automatically extracted enhanced dependencies achieves low accuracy, indicating that more work remains to be done to automatically create high-quality enhanced German treebanks.

To the best of our knowledge, our work presents the first investigation into creating Enhanced UD annotations for German treebanks. Our system achieves accuracies which we believe may already be useful in downstream applications (e.g., to increase recall in information extraction systems relying on syntactic dependencies). Furthermore, we hope that our findings may serve as a starting point for the creation of high-quality Enhanced UD corpora for the German language.[10]

## 5.2.2. Basic-to-Enhanced Converter

In this section, we describe the design of our basic-to-enhanced UD converter for German. Along the way, we discuss important differences w.r.t. the converter for English (Schuster and Manning, 2016).

### Enhancing Relative Clauses

As shown in Figure 5.4, in basic UD, relative clauses are represented as sub-trees, with the root of this subtree being linked to the noun modified by the clause. The basic UD representation provides no direct link between a relative pronoun and its antecedent. As the relative pronoun and the antecedent can be interpreted as co-referring, the referent of the antecedent can also be considered as an argument of the relative pronoun's head (here: "kenne"). Enhanced UD marks this by means of a `ref` relation linking the antecedent and the relative pronouns, as well as adding an argument link (here: `obj`) from the relative clause's main verb to the antecedent.

**Algorithm.** If a relative clause is indicated by an `acl:relcl` relation, we add a `ref` relation from the clauses' antecedent (which is head of the `acl:relcl` relation) to the

[10]The text of this section is based on Bürkle et al. (2021). Implementation of the rule-based Enhanced UD converter was done by Teresa Bürkle.

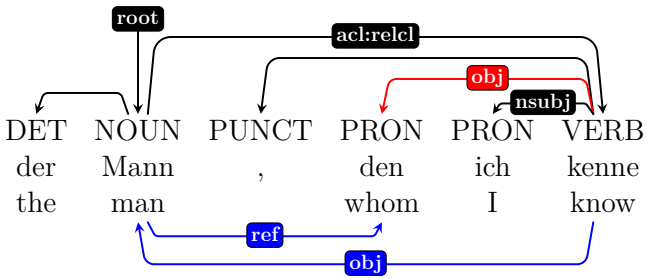



Figure 5.4.: Basic and enhanced representation of a relative clause.

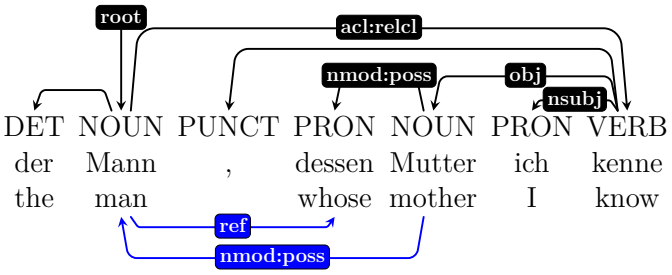



Figure 5.5.: Relative clause with possessive relative pronoun.

relative pronoun or relativizer. Because not all treebanks use the subtype `relcl` for this construction, an `acl` relation is also considered if the token that might be the relative pronoun has the UPOS `PRON` or the lemma "wo" ("where"). The relative pronoun is found by selecting the first child of the `acl` relation's dependent according to surface order which is linked to the relative clause's root via any relation except `punct`. The relation between the relative pronoun or relativizer and its head is deleted, and a relation from the relative pronoun's / relativizer's former head to the antecedent is added, using the relation type of the deleted relation, as in Figure 5.4.

The English converter (Schuster and Manning, 2016) treats relative clauses the same way, but their way of identifying the relative pronoun is slightly different to ours. The leftmost word contained in a word list of possible relative pronouns is taken as the dependent of the ref relation if any child or grandchild of the `acl:relcl` relation depends on a possible relative pronoun. In addition, only the specific subtype `acl:relcl` is considered for the identification of relative clauses.

We treat the special case of possessive relative pronouns as follows. As shown in Figure 5.5, if the relative pronoun is possessive, it is not a direct child of the relative clause's root, but rather attaches to a child node thereof. In German, the only two possessive relative pronouns are "deren" or "dessen." These word forms can be detected

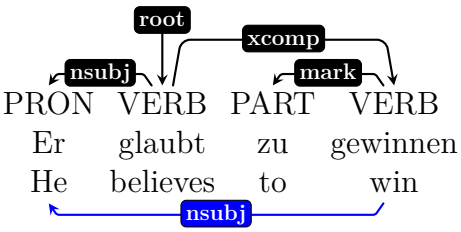



Figure 5.6.: Raising construction example.

easily and unambigously. In such constructions, the relative pronoun or relativizer (here: "dessen"), refers to "Mann." Accordingly, a `ref` relation is inserted between "Mann" and "dessen." In contrast to the case of Figure 5.4, the object of "kenne" is not the pronoun's antecedent "Mann," but "Mutter" within the relative clause. Hence, an `nmod:poss` relation is added between "Mutter" and "Mann" instead.

In English, this type of relative clause exists with the possessive pronoun "whose" functioning as a relativizer. When analysing the differences of our converter to the output of the converter of Schuster and Manning (2016), we noted the following. In English treebanks, the relevant relation is always `nmod:poss`, while in some German treebanks, the relative pronoun is linked via `det` or `det:poss`. Our converter hence covers these cases as well. Further, Schuster and Manning (2016) delete the `nmod:poss` relation between the possessive relative pronoun and its former governor in the enhanced layer, which our converter as used in Sec. 5.2.3 does not.[11]

### Enhancing Raising and Control

The linguistic phenomena called raising and control describe sentences with embedded verbs whose subjects have been "raised" to some higher position in the syntactic structure (Polinsky, 2013). In the constructions "Susan decided to start" and "Susan seemed to start," the subject of the embedded clause is shared with that of the matrix clause. In the control construction "Susan told them to start," the object "them" is also the subject of the embedded clause. While raising verbs (e.g., "seem") do not select their arguments, control verbs such as "tell" or "decide" do. In enhanced UD, as shown in Figure 5.6 and Figure 5.7, the two phenomena are represented in the same way, i.e., by making the `nsubj` relation explicit. The embedded clause is linked to the matrix verb via `xcomp` ("open clausal complement"), hence, the construction can be detected easily in the basic layer. However, resolving is non-trivial, as we discuss in Sec. 5.2.3.

[11]This oversight was fixed in a subsequent version of our converter.

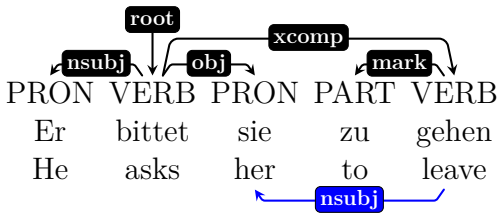



Figure 5.7.: Object control example.

**Algorithm.** We first identify `xcomp` relations whose head node is a verb or adjective and whose dependent is a verb, noun, or adjective node. For these, we add `nsubj` links starting at the `xcomp` relation's child node according to the following criteria. We check the outgoing links of the `xcomp` relation's head node (the embedding predicate) for the relations `iobj`, `obj`, and `nsubj` in this fixed order. For the first match, we assume that child node linked using this relation is the endpoint of the newly added `nsubj` relation.

Raising and control constructions where the controlling or raising predicate is a noun, such as "der Versuch, etwas zu tun" ("the attempt to do something") or "die Möglichkeit, etwas zu tun" ("the opportunity to do something") often lead to the insertion of wrong links. This is the reason for only addressing cases where the contolling or raising predicate is a verb or an adjective and the controlled or embedded predicate is a verb, an adjective or a noun.

### Enhancing Coordinate Constructions

As described in Sec. 3.3, coordinate constructions in UD are represented by a `conj` relation between the coordinated tokens (see, e.g., Figure 5.11). Generally, the first conjunct of a coordinate construction according to surface order is the head, with all other conjuncts attached to it via `conj` relations. Other dependency relations that are semantically shared by all conjuncts (e. g., `nsubj` or `obj` relations) are also attached to the first one. Since the first conjunct can also have dependents that are **not** shared by the other conjunct(s), this means that the dependency structures do not capture the information whether a dependent of the first conjunct is also a dependent of the other conjuncts.

For enhancing conjunctions, we distinguish two cases: (a) conjoined verbs, and (b) other types of conjunctions.

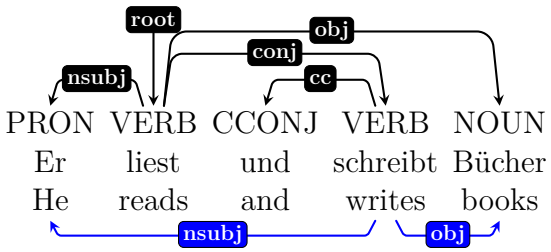



Figure 5.8.: Conjoined verbs that unambiguously share a subject and an object.

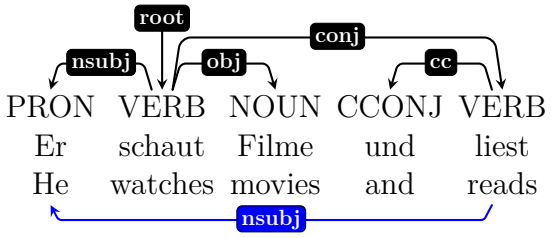



Figure 5.9.: Conjoined verbs that unambiguously share a subject, but no object.

**Conjoined Verbs.** When two verbs are conjoined, the basic UD representation lacks an explicit link between the second verb and complements that both verbs share semantically, such as subjects or objects.

**Algorithm.** If the second conjunct does not have an outgoing `nsubj` link, we add an `nsubj` link between the second conjunct and all nodes that are linked to the first conjunct via `nsubj` links.

Regarding shared objects, we distinguish three cases. If the second conjunct ("schreibt" ("writes")) comes after the first conjunct ("liest" ("reads")) and before the first conjunct's object ("Bücher" ("books")) according to surface order, as in Figure 5.8, we propagate an `obj` link between the second conjunct and this object. If the first conjunct's object comes after the first conjunct but before the second conjunct in surface order, as exemplified by Figure 5.9, no `obj` link is inserted. In these scenarios, it is possible to decide with certainty whether the first conjunct's object is semantically shared by both conjuncts, as it is always the case in English due to the strict surface order.

However, if the object comes before both conjuncts according to surface order, as in Figure 5.10, we face an ambiguity. In spite of both sentences in Figure 5.10 having the same dependency relations and POS tags, in the first sentence, the first conjunct's object ("Film") is semantically not shared with the second conjunct, while in the second sentence, both conjuncts share the object ("Bücher"). Therefore, in the second case, the

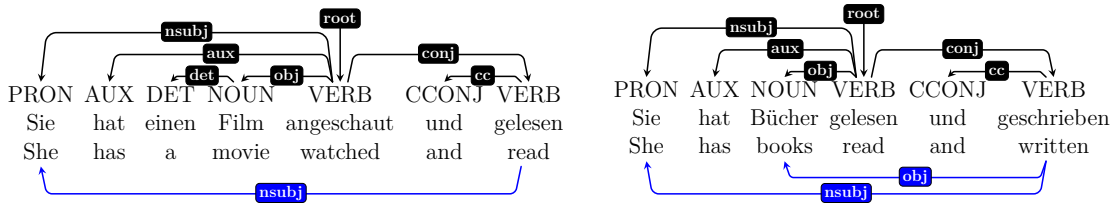



Figure 5.10.: Conjoined verb constructions that are ambiguous regarding the question whether they semantically share the object or not.

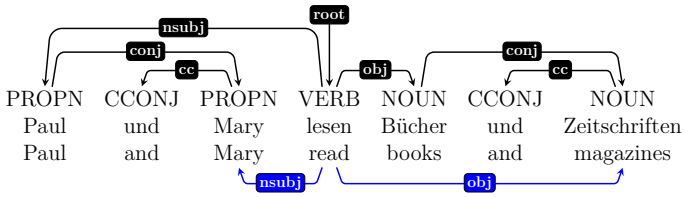



Figure 5.11.: Conjunction of noun phrases and objects, with both conjuncts sharing the verb.

enhanced UD representation should contain an additional `obj` link. However, we do not propagate in such cases in order to maximize precision rather than recall.

**Other types of conjunctions.** In case of coordinated noun phrases, adjectives, or modifiers, the token with an incoming `conj` relation (the "second conjunct"), is not explicitly linked to the governor or dependents of the first conjunct. That the two conjuncts fill the same roles in the sentence's argument structure is made explicit in the enhanced layer.

**Algorithm.** First, we identify the `conj` relation and the tokens it interlinks. We add a direct link between the second conjunct and the governor of the first conjunct. The type of the added relation is copied over from the relation linking the first conjunct to its governor.

### 5.2.3. Analysis and Evaluation

In this section, we first describe our manual corpus study to measure the performance of our converter. We compute precision, recall and F1 on enhanced links and compare the results to using the English converter by Schuster and Manning (2016) on German text. In addition, we then evaluate the performance of the converter in automatic parsing

settings.

**Manually Annotated Gold Standard.** We create a test set containing sentences from the two German treebanks GSD and PUD, against which the converter is then evaluated. The test set consists of three subsets, one for each of the three phenomena we address. The subset for coordinate constructions contains 50 sentences containing at least one `conj` relation between two tokens, while the subsets for relative clauses and raising and control structures contain 80 sentences containing respectively an `acl` or subtype of it or an `xcomp` relation. The more specific relation subtype `acl:relcl` would have been more suitable for extracting relative clauses, however, the GSD treebank only uses the generic type `acl`.

These subsets were created by first collecting all sentences for each phenomenon from each treebank and then selecting every n/25th sentence for the coordination subset and every n/40th sentence for the relative clause and raising and control subsets, with n being the total number of sentences found for this phenomenon in this treebank. After removing duplicates and six sentences with severe errors in the basic layer, this results in 196 annotated sentences which are the basis for our evaluation.

**Annotation.** The test set was then converted to include Enhanced UD representations by our system and manually corrected by three annotators, two computational linguists with extensive linguistic training, and one undergraduate student with basic linguistic training. Each annotator corrected a subset of the test set, discussing sentences they were unsure about with the other annotators.

Minor errors occuring in the basic layer (e.g., a wrong dependency type assigned or an inverted head and child node of a relation) were corrected, while more severe errors affecting multiple dependency relations or tokens lead to an exclusion of the sentence from the evaluation set. In total, 18 sentences were found to contain errors in the basic layer, and six were removed from the evaluation set due to severe errors.

**Performance of Converter in Gold Setting.** Rather than starting out with the original treebanks, we create the input for the converter by removing the enhanced layer from the manually corrected gold set. In other words, our evaluation takes into account the minor fixes we applied to the basic layer. Arguably, this procedure does not result in a completely realistic estimate how well our system would perform on the still somewhat noisy treebanks; however, we consider improving the consistency and quality of the current German treebanks to be future work and decided to abstract away from this

problem in our study in order to make it less dependent on the idiosyncrasies of the current treebank versions.

We then run our converter on the basic trees for these sentences and compare the enhanced dependency relations in the converted set to the enhanced layer of the gold set. In order to make a fair comparison, we freeze the code of our converter before starting out with our annotation study. The results reported here are thus an estimate how far we can get with a rule-based approach (which was based on extensive data analysis, but not targeted to our evaluation set).

In Table 5.3, we report precision, recall, and F1 for all links in the enhanced layer, as well as for several relevant relations. In addition to the overall score, calculated by evaluating all sentences in our test set, we compute scores for each phenomenon by evaluating all sentences in which this phenomenon occurs. This results in the evaluation set for a phenomenon containing more sentences than the originally sampled test set for this phenomenon did, because sentences from the other phenomena's subsets can be included. For example, a sentence originally selected for the relative clause set is included in the evaluation set for conjunctions if it also contains a `conj` relation. This results in the evaluation for coordination constructions being based on 117 sentences, the evaluation for relative clauses being based on 105 sentences and the evaluation for raising and control being based on 98 sentences.

Considering all sentences and label types, we achieve a precision of 89.5%, recall of 89.1% and F1 of 89.3%. Comparing the scores for the three phenomena implies that the converter works best on relative clauses with F1 of 94.2%. Recall of 100% for the `ref` relation indicates that all relative clauses in the test set were detected as such. For sentences containing conjunctions, all scores were slightly lower with an F1 of 89.3%. Raising and control is the phenomenon for which we achieved the lowest scores, with an F1 of 82.1%. Here, `nsubj` is the only relation type that our converter automatically propagates, while we did not restrict propagation of relation types in our manual annotation. Precision and recall both around 77% indicate that for some cases, our converter selects the wrong token as subject of the embedded or controlled predicate. This decreases both precision and recall, because the actual subject is lacking an `nsubj` link to the controlled or embedded predicate and at the same time a spurious `nsubj` link to the wrong node is created. E.g., for the sentence "wir verwenden Zeit darauf, ein Konzept zu entwickeln" ("we spend time on developing a concept,") instead of "wir" ("we"), the converter would treat "Zeit" ("time"), the object of "verwenden" ("spend"),

| Phenomenon | Basic Layer | System | Score | all | advmod | amod | nmod | nsubj | obj | obl | ref |
|---|---|---|---|---|---|---|---|---|---|---|---|
| all | | | total | 440 | 10 | 14 | 22 | 198 | 40 | 40 | 97 |
| | gold | ours | P | 89.5 | 100.0 | 100.0 | 95.0 | 82.7 | 97.4 | 95.2 | 95.1 |
| | | | R | 89.1 | 90.0 | 100.0 | 86.4 | 84.3 | 92.5 | 100.0 | 100.0 |
| | | | F1 | 89.3 | 94.7 | 100.0 | 90.5 | 83.5 | 94.9 | 97.6 | 97.5 |
| | gold | SM16 | F1 | 79.5 | 57.1 | 100.0 | 90.5 | 82.4 | 66.7 | 80.8 | 89.3 |
| | automatic | ours | F1 | 80.1 | 94.7 | 88.9 | 76.0 | 77.2 | 83.8 | 79.0 | 90.5 |
| | end-to-end parser | | F1 | 46.7 | 20.3 | 70.0 | 31.3 | 65.7 | 72.3 | 51.3 | 88.2 |
| conjunctions | | | total | 315 | 8 | 14 | 21 | 135 | 31 | 36 | 51 |
| | gold | ours | P | 90.0 | 100.0 | 100.0 | 94.7 | 85.2 | 96.7 | 94.7 | 91.1 |
| | | | R | 88.6 | 87.5 | 100.0 | 85.7 | 85.2 | 93.5 | 100.0 | 100.0 |
| | | | F1 | 89.3 | 93.3 | 100.0 | 90.0 | 85.2 | 95.1 | 97.3 | 95.3 |
| | gold | SM16 | F1 | 80.8 | 66.7 | 100.0 | 90.0 | 84.9 | 78.4 | 86.7 | 87.3 |
| | automatic | ours | F1 | 79.0 | 93.3 | 88.9 | 75.0 | 79.7 | 84.2 | 76.7 | 86.0 |
| | end-to-end parser | | F1 | 47.1 | 23.5 | 80.0 | 38.3 | 63.2 | 76.2 | 56.3 | 85.0 |
| rel. clauses | | | total | 316 | 7 | 7 | 11 | 127 | 31 | 29 | 97 |
| | gold | ours | P | 93.5 | 100.0 | 100.0 | 90.0 | 89.9 | 96.7 | 96.7 | 95.1 |
| | | | R | 94.9 | 85.7 | 100.0 | 81.8 | 91.3 | 93.5 | 100.0 | 100.0 |
| | | | F1 | 94.2 | 92.3 | 100.0 | 85.7 | 90.6 | 95.1 | 98.3 | 97.5 |
| | gold | SM16 | F1 | 81.4 | 25.0 | 100.0 | 85.7 | 88.5 | 62.2 | 76.3 | 89.3 |
| | automatic | ours | F1 | 83.6 | 92.3 | 84.2 | 70.6 | 80.9 | 88.5 | 80.6 | 90.5 |
| | end-to-end parser | | F1 | 51.6 | 26.7 | 76.2 | 31.0 | 68.4 | 73.5 | 53.5 | 88.2 |
| raising + | | | total | 221 | 2 | 7 | 13 | 131 | 14 | 9 | 33 |
| control | gold | ours | P | 84.6 | 100.0 | 100.0 | 100.0 | 77.3 | 100.0 | 100.0 | 97.1 |
| | | | R | 79.6 | 50.0 | 100.0 | 76.9 | 77.9 | 78.6 | 100.0 | 100.0 |
| | | | F1 | 82.1 | 66.7 | 100.0 | 87.0 | 77.6 | 88.0 | 100.0 | 98.5 |
| | gold | SM16 | F1 | 76.2 | 66.7 | 100.0 | 87.0 | 77.2 | 44.4 | 72.0 | 97.1 |
| | automatic | ours | F1 | 76.1 | 66.7 | 92.3 | 76.9 | 72.5 | 76.5 | 82.4 | 90.4 |
| | end-to-end parser | | F1 | 41.2 | 5.3 | 63.6 | 32.4 | 61.6 | 48.8 | 34.3 | 85.3 |

Table 5.3.: **Annotation study: converter results**. Rows for phenomena report results for **all** links in enhanced layer for set of sentences that contain the phenomenon. **all** = all links in enhanced layer in the subset of sentences, **total** = total count of this label type in the evaluated gold sentences. Number of sentences in the evaluation subset: Coordination: 117, Relative Clause: 105, Raising and Control: 98. SM16 = Schuster and Manning (2016).

as a subject of the embedded verb "entwickeln" ("develop").[12]

In the following, we compare our converter to the one created by Schuster and Manning (2016).[13] Our aim is to check whether implementing a targeted version for German was worth the effort compared to simply running the existing system developed for English. We made one adaptation to the English converter, which originally expects only `acl:relcl` to indicate relative clauses (see Sec. 5.2.2). We relax this constraint to looking for `acl`, as several German treebanks do not use the `relcl` subtype.

Since the English converter by Schuster and Manning (2016) uses some relation subtypes that we did not use in the gold set, only the generic relation type was considered in the evaluation. Overall, we found that the English converter performed worse on our test set, with an overall F1 score of 79.5% (about ten percentage points lower than our targeted version for German). For `obj` and `obl` relations, the reported scores diverge from our scores by 28.2 and 16.8 percentage points, respectively. The main reason for this is the different treatment of relative clauses in which the relative pronoun is governed by an `obj` relation. While we add an `obj` relation between the antecedent and the token governed by the `acl:relcl` or `acl` relation in this case, the converter by Schuster and Manning (2016) inserts an `obl:relobj` relation in this case, which is treated as an `obl` relation in the evaluation due to the evaluation algorithm ignoring subtypes. There is no meaningful difference in the handling of coordination except for the usage of more specific subtypes. Therefore, the difference in scores in Table 5.3 stems from the other phenomena that also occur in this evaluation subset.

Regarding raising and control constructions, in contrast to our converter (as explained in Sec. 5.2.2), the English converter propagates links when the head of the `xcomp` relation is a noun. Furthermore, while we used the general `nsubj` relation, they used the subtype `nsubj:xsubj` for links added while treating a raising or control construction. This is in fact an extension of the UD guidelines, which do not include the `nsubj:xsubj` subtype to date. Following the official guidelines, our system outputs only `nsubj`, but our final release includes an option for adding the `xsubj` subtype.

[12] In our evaluation, we also noticed minor implementation errors, e.g., the evaluated version of the converter does not check whether the second conjunct already has an outgoing `nsubj` link before propagating an addional link when enhancing conjunctions. These errors were fixed in a subsequent version of our converter (not evaluated here).

[13] To ensure the strongest possible baseline, we here use the version of their converter that includes modifications by us (as described in Chapter 4), fixing dependency propagation in certain edge cases involving coordinate constructions.

**Performance of Converter in Automatic Parsing Setting vs. End-to-End Graph Parsing.** To evaluate our converter's performance on automatically parsed basic dependencies, we use the STEPS dependency parser (see Chapter 7), along with a pre-trained model for German, to parse our test set and then run our converter on the output. We find that the performance of our rule-based system drops by about 10% compared to the results achieved when using gold basic dependencies (see rows labeled "automatic" in Table 5.3). This is in line with our findings on coordinate constructions in English Enhanced UD (see Chapter 4), where we also saw that a pipeline-based approach leads to considerable performance decreases. However, precision and recall exceed 75% in most cases, meaning that the resulting output may still be useful in practice.

Finally, we also present the alternative path of directly training an enhanced UD parser on a treebank created using our converter. Using the STEPS framework, we train an end-to-end graph parser on the converted GSD treebank, use it to parse our test set, and again evaluate against our manual annotations. As shown in Table 5.3, F1 for the enhanced layer drops by over 40%. By contrast, the parser achieves an F1 score of 85.1% on the same evaluation set for the prediction of the *basic* layer. These results indicate that the enhanced data produced by converting the existing German treebanks is, as of yet, of limited use for training machine learning systems.

### 5.2.4. Summary and Outlook

In this section, we have described our approach of enriching German UD treebanks with enhanced dependencies by implementing a rule-based converter. Evaluating on a set of manually annotated sentences, our system achieves an F1 score of around 90% when operating on gold basic dependencies, outperforming the English-based system by Schuster and Manning (2016). In an automatic parsing setting, accuracy is lower with an F1 score of around 80 %, which may still be useful in practice. Training an enhanced parser on treebanks created using our converter yields poor results.

In our annotation study, we found that our system fails mainly due to errors in the basic layer, which have been derived semi-automatically from German treebanks annotated with different dependency frameworks. For example, an `xcomp` relation in the wrong direction leads to severe errors in the enhanced layer when following our rule-based approach. We further observed divergences in regards to the annotation schemes applied across German treebanks, for instance the usage of different relation types for the same sentence structures. We therefore conclude that measures to increase consistency across German treebanks are an important step to facilitate the conversion

to enhanced dependencies.

**Outlook.** There remains a number of ambiguous cases for which we have not found a solution thus far, such as the propagation of objects in case of two conjoined verbs (see Sec. 5.2.2). Regarding raising and control structures, examples like "Wir lassen den Jungen gehen" ("We let the boy leave") vs. "Wir lassen den Reifen montieren" ("We have the tyre mounted") are identical regarding their basic dependency structure and POS tags, yet should be treated differently by the converter. In control structures such as "Wir entschieden, hier nicht zu schwimmen" ("We decided not to swim here,") the subject of the embedded verb "schwimmen" ("swim") is the same as that of "entschieden" ("decided"), while in the sentence "Wir empfehlen, hier nicht zu schwimmen" ("We recommend not to swim here,") the subject does not co-refer with the speaker. A possibility to improve here might be the usage of word lists containing control/raising verbs for which the subject of the controlled/embedded predicate can not co-refer with the speaker (as for "recommend").

To conclude, our work demonstrates that rule-based conversion is a suitable method for creating enhanced dependencies for German. However, our analysis also shows that taking language-specific phenomena into account is important to ensure correctness of the extractions. Hence, creating targeted enhanced UD converters for more languages is a promising avenue, but will likely require further manual work for each instance.

## 5.3. Conclusion

In this chapter, we have presented two case studies on improving and extending German UD treebanks: First, we have unified the treatment of German preposition-determiner contractions across treebanks; and second, we have extended existing treebanks with enhanced dependencies. With this work, we simultaneously took steps to answer our research questions **RQ1**–**RQ3** in the context of German UD treebanks.

Regarding **RQ1,** we find that **the quality of existing German UD treebanks** is high, but that there are nonetheless specific areas where improvements are both desirable as well as possible. As a first example examined by us, annotation consistency across treebanks was poor for the linguistic phenomenon of preposition-determiner contractions. As a second example, the enhanced dependency layer was not present in German UD treebanks prior our work.

Regarding **RQ2,** we have investigated **rule-based approaches for improving**

**German UD treebanks.** Our results show that rules are performing virtually flawlessly in the context of the highly regular phenomenon of preposition-determiner contractions, but their performance is more ambiguous for the more complex task of extracting enhanced dependencies from the basic annotation layer. This holds particularly for cases where more involved semantic interpretation is required. For these instances, machine-learning based approaches may be better suited, as we also found for coordinate constructions in English Enhanced UD (see Chapter 4).

Similarly, regarding **RQ3,** we found that **our enhancements differ in terms of their effects on parsing performance.** In the case of preposition-determiner contractions, we see consistent increases in parsing accuracy, due to the fact that harmonization of annotations leads to less noise in the training signal. By contrast, when using rule-extracted enhanced dependencies to train a German enhanced UD parser, the resulting system displays poor performance, indicating that this problem requires either hand-annotation or considerably more sophisticated computational methods if it is supposed to yield a parser that is useful for practical applications.

**Open issues.** Our work in this chapter suggests a number of open issues and follow-up questions. Most strikingly, our results indicate that there are limits to what is achievable by purely rule-based corpus enhancing procedures, as evidenced by the remaining errors when creating enhanced UD from the basic layer via rules. The question therefore arises **how to better handle complex semantic phenomena where rules do not suffice.** Promising avenues for this may be hybrid or even purely end-to-end approaches; however, the fact that there is often little to no gold-standard data available remains a challenge for these systems.

A second aspect to consider is that of **transfer to other languages.** Though we have demonstrated via our experiments on German treebanks that rule-based approaches do not only work for English, these two languages are still closely related, raising the question of whether approaches similar to the ones outlined in this chapter also transfer to less closely related languages (e.g., Arabic, Hungarian, or Japanese). Transfer to more languages and corpora raises additional complications: Treebanks (even within the same language) often differ in how they handle the same linguistic phenomenon, making consistent processing more difficult. Furthermore, low resource languages further exacerbate the problem of training data availability.

# Part III.

# Computational Studies on Transformer-based Universal Dependency Parsing

# 6. Introduction to Graph-based Universal Dependency Parsing

In this part of the dissertation, we investigate dependency parsing architectures and techniques, addressing our research questions **RQ4 (Which are the main factors determining the performance of neural graph-based dependency parsers?)** and **RQ5 (How can neural dependency parsers be adapted for Enhanced UD?).** In particular, we determine factors which are most important for the performance of Basic and Enhanced UD parsers, respectively. Methodologically, we operate by implementing our own neural graph-based dependency parser called STEPS, which (a) is built on top of state-of-the-art transformer language models, and (b) has a modular structure that allows us to compare different architectural configurations in a principled fashion.

As a first main contribution, we use STEPS to identify the factors most crucial to dependency parsing performance. We find that the choice of underlying language model is by far the most important factor in this regard. By contrast, our experiments show that other setups and techniques sometimes considered helpful (e.g., the inclusion of an LSTM layer) actually have little to no impact and can in some cases even degrade parsing performance.

As a second main contribution, we extend the STEPS parser to the Enhanced UD formalism. With the goal of a simple yet robust architecture, we outline our methods for adapting our graph-based parser to the prediction of general graphs rather than trees, as well as other peculiarities of the Enhanced UD formalism (such as dependency labels with lexical material). We show that relatively simple adaptations suffice to reach very high parsing accuracy on a number of languages, as evidenced by STEPS's performance in two parsing shared tasks.

In sum, our findings indicate that modern language models display very high efficacy at learning syntactic patterns from data, to the extent that swapping out one such model with a better one (e.g., in terms of larger parameter count or more thorough pre-training) is usually the most reliable way to increase parser performance. By contrast,

the relative importance of sophisticated task-specific architectures is decreased. This is evidenced by the low impact of architectural variations on parsing accuracy as well as the small degree of modifications necessary to adapt a basic UD parser to the Enhanced UD formalism. Furthermore, the biaffine parser architecture can be transferred to other tasks framed as dependency parsing problems (in our case, Negation Resolution) with minimal architectural modifications and achieve robust results.

**Outline of this part of the dissertation.** The following chapters are structured as follows:

- This chapter briefly surveys techniques for UD parsing, with a special focus on graph-based parsing and recent developments in the neural era.
- Chapter 7 describes STEPS, our modular graph-based UD parser, the experiments on parser architectures we conducted with it, and our conclusions regarding important factors in parsing architectures (Grünewald et al., 2021a).
- Chapter 8 describes our work on extending STEPS to Enhanced UD and presents evaluations based on the participation in two parsing shared tasks (Grünewald and Friedrich, 2020a; Grünewald et al., 2021b).

## 6.1. Syntactic Dependency Parsing Approaches

The predominant approaches for syntactic dependency parsing are transition-based parsing, graph-based parsing (Jurafsky and Martin, 2021), and – more recently – parsing as sequence labeling (Strzyz et al., 2019). This section briefly outlines transition-based parsing and sequence labeling parsing and then describes graph-based parsing (and recent developments within it) in greater detail, as our models and experiments fall within this paradigm.

**Transition-based dependency parsing** uses two data structures – a stack and a buffer – to cast dependency parsing as a problem of predicting actions which correspond to assembling a dependency tree incrementally. Such actions may be shifting a token from the buffer onto the stack, or creating a dependency relation between the top two tokens on the stack. In the context of machine learning, transition-based parsers are trained to predict the correct next action based on the current state of buffer and stack (Jurafsky and Martin, 2021).

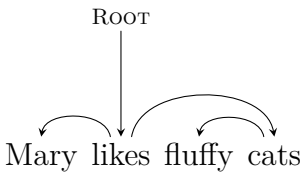



Figure 6.1.: Example dependency tree.

| | ROOT | Mary | likes | fluffy | cats |
|---|---|---|---|---|---|
| Mary | 0.01 | 0.02 | 0.88 | 0.07 | 0.02 |
| likes | 0.95 | 0.01 | 0.00 | 0.03 | 0.01 |
| fluffy | 0.09 | 0.13 | 0.05 | 0.02 | 0.71 |
| cats | 0.03 | 0.10 | 0.74 | 0.12 | 0.01 |

Figure 6.2.: Syntactic head probabilities.

Due to its incremental nature, transition-based parsing has traditionally struggled with error propagation (e.g., for long-range dependencies) and the construction of non-projective trees (McDonald and Nivre, 2007). However, more sophisticated transition schemes – e.g., including the possibility of swapping nodes from the stack back into the buffer (Nivre, 2009) – can allow for non-projective trees, while contextualized token representations (e.g., from pre-trained language models) have been shown to reduce error propagation due to their encoding of global sentence context (Kulmizev et al., 2019).

**Dependency parsing as sequence labeling** produces dependency trees for sentences by assigning to each token a label that encodes its attachment parent in the dependency tree. This location may be encoded in different ways, e.g., by utilizing the part-of-speech tag of the parent. In this case, a label (NOUN,+1,det) would indicate to attach the token to the nearest noun on the right of it via the *det* relation. Strzyz et al. (2019) find that this POS tag encoding scheme performs best, yielding a parser with near state-of-the-art accuracy and parsing speed superior to even recent transition-based parsers.

**Graph-based dependency parsing** involves searching the entire space of possible parse trees/graphs for a given sentence, with the goal of finding the one that maximizes some scoring function. *Arc-factored* parsing is a common formulation of this approach that assumes the score of a tree is the product of the scores of its individual dependency edges (Jurafsky and Martin, 2021). Using this assumption, a classifier (usually machine-learning-based) predicts, for each token within the input sentence, a probability distribution over its possible syntactic heads (i.e., all other tokens in the sentence, as well as a special ROOT token), as shown in the rows of Figure 6.2. In a second step, the logarithms of the resulting probabilities are then interpreted as edge weights between pairs of nodes corresponding to tokens and a maximum spanning tree is extracted, using, e.g., the Chu-Liu/Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967). This maximum spanning tree is then returned as the dependency tree for the input sentence, as it maximizes the product of all edge weights.

As it predicts the constituent parts of a parse tree (i.e., dependency edges) directly,

rather than merely outputting a sequence of actions to assemble such a structure (as in transition-based parsing), it can be argued that arc-factored graph-based parsing operates closer to the inherent structure of dependency trees/graphs, thus constituting an example of "inductive bias." As a consequence, graph-based parsers can handle non-projectivity without the need for special mechanisms and struggle less with long-range dependencies. Indeed, they have empirically been found to achieve better accuracies than transition-based parsers, though the gap has closed since the introduction of contextualized word representations (Kulmizev et al., 2019). Disadvantages of graph-based parsers are found in the realm of practical considerations: Because they must model relations between all pairs of tokens, they display lower parsing speed and have a larger memory footprint.

## 6.2. Recent Developments in Graph-based UD Parsing

Recent years have seen considerable improvements in the performance of graph-based UD parsers. These improvements can in large part be attributed to two developments: (a) the introduction of deep biaffine classifiers (Dozat and Manning, 2017), which at the point of publication constitute the de-facto standard approach for graph-based dependency parsing, and (b) the rise of pre-trained distributed word representations, particularly transformer-based contextualized embeddings such as BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019). Both characteristics are present in recent top-performing systems (Che et al., 2018; Straka et al., 2019; Kondratyuk and Straka, 2019; Kanerva et al., 2018, 2020; Che et al., 2018).

**Neural dependency parsing with biaffine classifiers.** Biaffine classifiers for graph-based dependency parsing can be traced back to a seminal paper by Dozat and Manning (2017). Assuming some vector representation of input tokens (*word embeddings*), graph-based parsing using biaffine classifiers essentially consists of the following steps:

First, a head representation $\mathbf{h}_i^{head}$ and a dependent representation $\mathbf{h}_i^{dep}$ are created for each input token $i$ represented as embedding vector $\mathbf{r}_i$ via two single-layer feedforward neural networks:

$$\mathbf{h}_i^{head} = \mathrm{FNN}^{head}(\mathbf{r}_i) \tag{6.1}$$

$$\mathbf{h}_i^{dep} = \mathrm{FNN}^{dep}(\mathbf{r}_i) \tag{6.2}$$

These representations are then fed into the biaffine function, which maps head–dependent

pairs $(i, j)$ onto vectors $\mathbf{s}_{i,j}$ of arbitrary size:

$$\mathbf{s}_{i,j} = \text{Biaff}\left(\mathbf{h}_i^{head}, \mathbf{h}_j^{dep}\right) \tag{6.3}$$

$$\text{Biaff}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top \mathbf{U}\mathbf{x}_2 + W(\mathbf{x}_1 \oplus \mathbf{x}_2) + \mathbf{b} \tag{6.4}$$

$\mathbf{U}$, $W$ and $\mathbf{b}$ are learned parameters; $\oplus$ denotes the concatenation operation.

The scores $\mathbf{s}_{i,j}$ can then be leveraged in different ways to construct an output tree or graph. In a **factorized** approach (Dozat and Manning, 2017), two instances of biaffine classifiers are used. The first classifier (the "arc scorer") is responsible for predicting which (unlabeled) edges exist in the output structure. It predicts, for each token, a probability distribution over potential syntactic heads (i.e., all other tokens in the sentence). Specifically, the individual $\mathbf{s}_{i,j}$ here are scalars that are transformed into log-probabilities via the log-softmax operation and then fed to a maximum spanning tree algorithm (e.g., the Chu-Liu/Edmonds algorithm; Chu and Liu, 1965; Edmonds, 1967). After this, a second classifier (the "label scorer") assigns dependency labels to the edges predicted by the spanning tree algorithm. In this classifier, the $\mathbf{s}_{i,j}$ are vectors whose dimensionality corresponds to the number of possible dependency labels.

In an **unfactorized** approach, proposed by Dozat and Manning (2018) for semantic graph parsing, only a label scorer is used to predict the dependency label for each pair of tokens directly. In this formulation, non-existence of a dependency is encoded using simply another label ($\varnothing$). The unfactorized approach is suited for generating general graph structures, as it does not impose the constraint of each token having exactly one parent.

**Contextualized word embeddings in Dependency Parsing.** Like in other sub-fields of natural language processing, using contextualized word embeddings has become the de-facto standard when building syntactic parsers. Dyer et al. (2015) use LSTM-based contextual representations for the stack and buffer in transition-based parsing, while Kiperwasser and Goldberg (2016) use BiLSTM-based feature representations for individual tokens in both graph-based and transition-based parsing. In both of these cases, the underlying LSTM is trained simultaneously with the target task. In contrast, recently the predominant approach towards contextualized word representations has been to pre-train systems on large-scale language modeling objectives, then taking their representations as input for a target task, optionally while continuing to fine-tune them. This approach was initially proposed using an LSTM-based system (ELMo; Peters et al.,

| Parser | Pre-trained embed. | LSTM | MTL |
|---|---|---|---|
| StanfordNLP | word2vec, fastText | yes | no |
| UDPipe 2.0 | word2vec, fastText | yes | yes |
| HIT-SCIR | fastText, ELMo | yes | no |
| UDify | mBERT | no | yes |
| Trankit | XLM-R | no | yes |

Table 6.1.: Settings for a number of previously state-of-the-art graph-based dependency parsers. "LSTM" states whether the parser makes use of an LSTM for its token representations, and "MTL" states whether the parser is also trained to simultaneously predict other UD properties such as POS tags or morphological features.

2018) and has since been transferred to transformers (e.g., BERT; Devlin et al., 2019). Transformer-based pre-trained language models have proven wildly successful and have become a standard method for a wide range of NLP tasks, including syntactic dependency parsing.

**Recent graph-based parsers.** Table 6.1 shows the configurations of three parsers that were among the best-performing systems in the CoNLL 2017 and CoNLL 2018 Shared Tasks on UD parsing, as well as the more recent UDify and Trankit parsers.

**StanfordNLP** (Dozat et al., 2017) was one of the first systems to apply the biaffine graph-based parser architecture to Universal Dependency parsing. Its token representations make use of pre-trained word2vec (Mikolov et al., 2013) embeddings that are contextualized using a BiLSTM. **UDPipe 2.0** (Straka, 2018) uses a multi-task setup in which POS and feature tagging, lemmatization, and dependency parsing share layers. The system was later extended (Straka et al., 2019, henceforth **UDPipe+**) by incorporating multilingual BERT (mBERT; Devlin et al., 2019) in its token representations. **HIT-SCIR** (Che et al., 2018) was one of the first UD parsers to make use of contextualized pre-trained word embeddings (in the form of ELMo; Peters et al., 2018). The model does not make use of a multi-task training setup. **UDify** (Kondratyuk and Straka, 2019) differs from previous UD parsers in two ways. First, it does not use an LSTM layer for token representation, instead using a learned scalar mixture of mBERT layers and fine-tuning mBERT during training. This is in contrast to the three aforementioned parsers, which do not fine-tune their pre-trained token embeddings. Second, UDify learns a single model for all languages, concatenating all UD 2.5 training sets. **Trankit** (Nguyen et al., 2021) is a recently released end-to-end UD parsing system built on the XLM-R language model. In contrast to UDify and our own STEPS parser, it does not fine-tune

the entire language model, but instead inserts Adapter layers (Pfeiffer et al., 2020a,b) to efficiently create language-specific models for 56 languages.

**Overarching trends.** Recent trends in graph-based dependency parsing can be seen as a microcosm of general developments within NLP. In particular, we can summarize the following points:

- While there continues to exists a large variety of different architectures for graph-based dependency parsing, one can nonetheless observe a convergence towards the same "backbone," namely (fine-tuned) transformer language models with some classifier layers on top. This is in line with the general ubiquity of transformers in NLP.

- The importance of recurrent architectures (e.g., LSTMs) is diminishing, with such layers now often being omitted in the presence of transformers, again in line with general trends towards "general-purpose" transformer backbones.

- The majority of graph-based parsers are now first-order, arc-factored systems. Hand-crafted structural features (such as grandparent or sibling information) barely play a role anymore, and remaining explicit structural constraints are mostly restricted to ensuring tree/graph validity.

- Using transformers, performance differences between parser paradigms (graph-based, transition-based) have diminished, confirming the larger trend of model architecture having less impact in the presence of high-quality language models. (Note, however, that this presupposes the existence of such language models, which may not be the case particularly for low-resource languages.)

- Multi-task learning setups (e.g., performing POS tagging simultaneously with parsing) are becoming more popular.

## 6.3. Summary and Open Issues

In this chapter, we have provided a brief overview of techniques for syntactic dependency parsing, focusing in particular on graph-based UD parsing. In our short survey of recent parsing systems, we have found that overall trends within NLP are reflected within dependency parsing, particularly w.r.t. the usage and utility of pre-trained language model

backbones and a simultaneous relative loss in importance for task-specific, structurally informed, or feature-based neural architectures.

**Open issues.** Despite the overall trend towards relatively simple transformer-based architectures, there remain a considerable number of implementation and configuration choices whose impact on parser performance is less well understood. This is evidenced by the many different model configurations (see Table 6.1) present in parsers that have achieved top results in recent shared tasks addressing UD parsing (Zeman et al., 2017, 2018; Bouma et al., 2020, 2021). The choices include (a) the particular pre-trained word embeddings or language model to use, (b) whether to utilize an LSTM in addition to (fine-tuned) contextualized word embeddings; and (c) whether to use a multi-task training setup simultaneously predicting additional UD features (such as morphology or parts of speech) during parsing.

In Chapter 7, we investigate these configurations in detail in order to answer **RQ4 (Which are the main factors determining the performance of neural graph-based dependency parsers?)**. Following this, in Chapter 8 we present an extension of our parser model to the Enhanced UD framework in order to answer **RQ5 (How can neural dependency parsers be adapted for Enhanced UD?)**.

# 7. STEPS: A Modular Graph-Based Dependency Parser

## 7.1. Introduction

As shown in Chapter 6, current state-of-the-art dependency parsers differ across many architectural dimensions. Recalling **RQ4 (Which are the main factors determining the performance of neural graph-based dependency parsers?)**, the aim of this chapter is to disentangle the effects of such factors and determine their individual impact. In order to do so, we appeal to the concept of "Occam's razor" by way of avoiding architectural elements that do not bring about a testable advantage. With this idea in mind, we introduce STEPS (the **S**tuttgart **T**ransformer-based **E**xtensible **P**arsing **S**ystem), a modular graph-based dependency parser which implements commonly used modules such as biaffine scorers (Dozat and Manning, 2018) or LSTM layers (Straka, 2018) (see Figure 7.1). Using STEPS, we perform a series of experiments on the UD treebanks of a diverse set of languages. Our setup facilitates estimating the impact of the various architectures and configuration decisions in a comparable way.

Our most important insight is that a relatively simple architecture using biaffine heads on top of a fine-tuned XLM-R language model (Conneau et al., 2020) leads to the highest parsing accuracy for almost all languages in our study, outperforming prior systems on most languages. Our analysis further indicates that LSTM layers do not lead to benefits. Simplifying the architecture even further by using a single scorer for edge and label prediction (unfactorized classifier) results in similar performance but on average leads to longer training times. Taken together, these results give credence to the notion that fine-grained architectural choices are less important in the presence of state-of-the-art pre-trained word representations; in fact, increased model complexity may even hurt performance. At the same time, the less efficient training of unfactorized classifiers also indicates that structural considerations cannot be completely ignored even in modern pre-trained models.

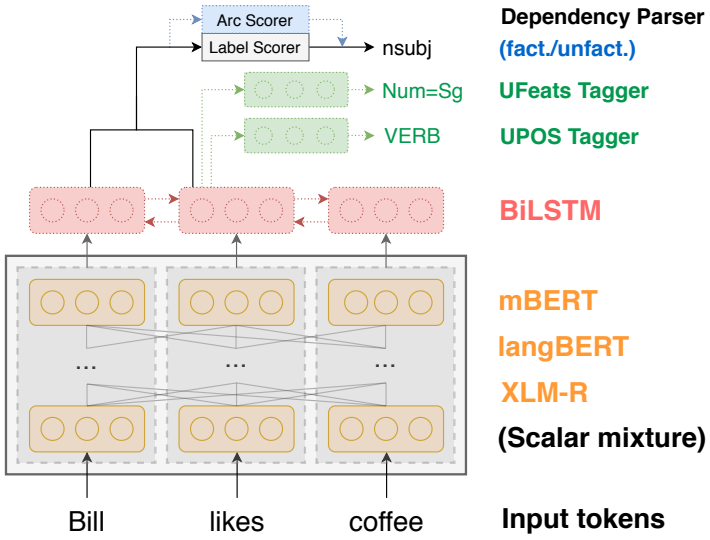



Figure 7.1.: Modular architecture of the STEPS parser. Dotted lines denote optional components, i.e., in its most basic setup, the parser consists only of a transformer language model with a biaffine classifier on top.

In summary, our contributions in this chapter are as follows:

(1) We introduce STEPS, a new graph-based dependency parser designed to be modular and easily extensible. At the time of its release, STEPS achieved new state-of-the-art UD parsing performance (in terms of LAS) for 10 out of the 12 typologically diverse languages in our study.

(2) We conduct a detailed experimental study, identifying components of parser architecture that are really necessary to obtain a strongly performing system that is applicable across a wide range of languages. The best configuration uses XLM-R, no LSTM layer, and a factorized edge and label scoring architecture.

(3) We show that multi-task setups predicting additional features as commonly employed in UD parsing may confound results for parsing for individual languages; we hence propose to compare parsing accuracy in unified evaluation settings in future work.

The rest of this chapter is structured as follows. Sec. 7.2 presents related work on analysing and comparing parsers. Sec. 7.3 describes the architecture and configuration options for our new STEPS parser, at the same time introducing the various factors studied in our experiments. The results of these experiments are reported in Sec. 7.4. Finally, we discuss our results w.r.t. our research questions and implications for future

parser design (Sec. 7.5).[1]

## 7.2. Related Work

This section presents related work on multi-purpose parsers and systematic parser analyses.

**Multi-Purpose Parsers.** Other parsers with modular or extensible architectures include **Alto** (Gontrum et al., 2017), a prototyping tool for new grammar formalisms based on Interpreted Regular Tree Grammars (IRTGs), and **PanParser** (Aufrant and Wisniewski, 2018), a modular framework for transition-based dependency parsing. In contrast to these two, STEPS is a graph-based dependency parser that focuses on easy configuration of different transformer-based language models and neural architecture variants.

**Parser Analyses and Comparisons.** Recent years have seen a wide range of studies comparing different language models for dependency parsing (e.g., Kanerva et al., 2018; Pyysalo et al., 2020; Smith et al., 2018). Additionally, several studies have investigated the amount of implicit syntactic information captured in pre-trained LMs such as ELMo and BERT (Tenney et al., 2019a,b; Hewitt and Manning, 2019). Conversely, several studies have investigated the utility of structural features for dependency parsing in the presence of LSTMs and/or contextualized word embeddings, generally finding that their impact is diminished in the presence of contextual information (Falenska and Kuhn, 2019; Fonseca and Martins, 2020).

Kulmizev et al. (2019) compare the effect of deep contextualized word embeddings on transition-based and graph-based dependency parsers, showing that their inclusion makes the two approaches virtually equivalent in terms of parsing accuracy. Our work is similar to theirs in the sense that we also evaluate several very different dimensions of parser architecture at the same time, utilizing the same underlying backbone and thus ensuring comparability across experiments.

## 7.3. Model Description

In this section, we describe our modular dependency parser STEPS (**S**tuttgart **T**ransformer-based **E**xtensible **P**arsing **S**ystem). Each subsection focuses on a particular aspect of

[1]The text of this chapter is based on Grünewald et al. (2021a).

| Language | Transformer LM | UD Treebank |
|---|---|---|
| Arabic | ArabicBERT-large (Safaya et al., 2020) | PADT (Smrž et al., 2008) |
| Chinese | Chinese BERT (Devlin et al., 2019) | GSD |
| Czech | Slavic-BERT (Arkhipov et al., 2019) | PDT (Bejček et al., 2012) |
| English | RoBERTa-large (Liu et al., 2019) | EWT (Silveira et al., 2014) |
| Finnish | FinBERT (Virtanen et al., 2019) | TDT |
| German | German BERT (`github.com/dbmdz/berts`) | GSD (McDonald et al., 2013) |
| Hindi | WikiBERT-Hindi (Pyysalo et al., 2020) | HDTB (Bhat et al., 2017; Palmer et al., 2009) |
| Italian | Italian BERT-XXL (`github.com/dbmdz/berts`) | ISDT |
| Japanese | WikiBERT-Japanese (Pyysalo et al., 2020) | GSD |
| Korean | KR-BERT (Lee et al., 2020) | Kaist (Chun et al., 2018) |
| Latvian | WikiBERT-Latvian (Pyysalo et al., 2020) | LVTB |
| Russian | RuBERT (Kuratov and Arkhipov, 2019) | SynTagRus (Droganova et al., 2018) |
| Multilingual | mBERT (Devlin et al., 2019) | – |
| Multilingual | XLM-R (Conneau et al., 2020) | – |

Table 7.1.: Language models and UD treebanks used in our experiments. Citations for treebanks are given where provided in treebank repository documentation.

the parser setup, providing background on its usage and its potential impact on parser performance.

### 7.3.1. Input Token Representation

STEPS provides a number of different options for input token representation. As described in Chapter 6, parsers have made use of a variety of pre-trained embeddings, with transformer-based language models having become the predominant current approach. We hence focus on the latter and compare multilingual BERT (**mBERT**; Devlin et al., 2019), language-specific BERTs (henceforth collectively referred to as **langBERT**), and the multilingual **XLM-R-large** model (Conneau et al., 2020). XLM-R utilizes the pre-training optimizations first proposed for RoBERTa (Liu et al., 2019), which includes training on a considerably larger amount of data than BERT. A detailed overview of all transformer models used in our experiments is provided in the second column of Table 7.1.

STEPS represents each token $i$ using a vector $\mathbf{r}_i$ corresponding to the embedding of its first word-piece token. Following Kondratyuk and Straka (2019), we compute token embeddings as weighted sums of the representations of the respective tokens given by the internal transformer encoder layers, resulting in either 768- oder 1024-dimensional embeddings depending on the transformer model used. Coefficients for this sum are learned during training, and layer dropout is applied in order to prevent the model from focusing on particular layers. Our model learns a different set of these coefficients for each output task (see Sec. 7.3.2 and Sec. 7.3.3 below). In addition to the above described

**transformer-only** setting, we also compute another version of token embeddings by feeding the embeddings computed by the sum operations into a multi-layer bidirectional **LSTM** (BiLSTM), whose per-token output then constitutes $\mathbf{r}_i$.

### 7.3.2. Biaffine Classifier Architecture

STEPS makes use of biaffine classifiers as proposed by Dozat and Manning (2017). We compare two variants of the method for extracting dependency trees: The **factorized** approach and the **unfactorized** approach.

As outlined in Sec. 6.2, the **factorized** approach (Dozat and Manning, 2017) uses two instances of biaffine classifiers: One to predict the (unlabeled) tree structure and one to assign dependency labels to the predicted edges. This is the standard approach for dependency tree parsing. By contrast, the **unfactorized** approach uses only a label scorer to directly predict dependency labels for all token pairs, representing non-existence of dependencies using simply another label ($\varnothing$). We adapt this approach to tree parsing by discarding the arc scorer and computing the edge weights for the Chu-Liu/Edmonds maximum spanning tree algorithm as $\log(1-P(\varnothing))$ in order to extract a labeled dependency tree directly. To the best of our knowledge, this is the first time that the unfactorized architecture has been applied to the parsing of dependency tree structures.

### 7.3.3. Multi-Task Training

We study the effects of a multi-task training setup by implementing two approaches to training our parser: (a) **dep-only**, in which the model is trained only on syntactic dependencies; and (b) multi-task learning (**MTL**), in which the model additionally predicts universal part-of-speech tags (UPOS) and morphological features (UFeats). We follow Kondratyuk and Straka (2019) by learning different coefficients for the transformer layers for these tagging tasks (see Sec. 7.3.1) and then using a single-layer feed-forward neural network to extract logit vectors over the respective label vocabularies. By default, the loss for the entire system is computed as the sum of losses for the individual output modules (UPOS tagger, UFeats tagger, and dependency parser). However, we also add the option of scaling the loss of the individual output modules in order to prevent individual tasks from overwhelming the system as a whole (see Sec. 7.4.6).

## 7.4. Experiments

This section describes our experimental setup and reports the results of our experiments on pre-trained embeddings (Sec. 7.4.3), factorized vs. unfactorized parser architecture (Sec. 7.4.5), LSTM usage (Sec. 7.4.4), and multi-task training (Sec. 7.4.6).

### 7.4.1. Experimental Setup

**Languages and treebanks.** We select 12 languages, covering a diverse range of language families and writing systems, by applying linguistic criteria similar to those outlined by de Lhoneux et al. (2017). For each language, we select the largest available treebank from UD 2.6 for which token data is freely available. These treebanks are listed in the third column of Table 7.1. In all of our experiments, we use gold tokens and train language-specific models, testing on the test set of the respective treebank.

**Evaluation metrics.** We compute UAS and LAS using the official evaluation script for the CoNLL 2018 Shared Task.[2] UAS (Unlabeled Attachment Score) computes the fraction of tokens that have been assigned the correct syntactic head. LAS (Labeled Attachment Score) records the fraction of tokens that have been assigned the correct syntactic head with the correct edge label. For more details on UD evaluation metrics, see Chapter 3.

### 7.4.2. Implementation

Our parser is implemented in Python, using PyTorch (Paszke et al., 2019) and the Huggingface Transformers library (Wolf et al., 2019). Training is performed on a single nVidia Tesla V100 GPU.

**Hyperparameters.** We aim to obtain a simple yet high-performing hyperparameter configuration. To do so, we start out with the configuration of UDify, which is architecturally quite similar to STEPS, and tune parameters using grid search in ca. 40 runs on a small development set (consisting of English, Arabic, and Korean data), aiming at a simplified setup that achieves good results across these diverse languages. The hyperparameters examined by us were

- Hidden size of the biaffine classifier (256 / 512 / 768 / 1024)

[2] `https://universaldependencies.org/conll18/conll18_ud_eval.py`

| | |
|---|---|
| **Transformer LM** | |
| Token mask probability | 0.15 |
| Layer dropout | 0.1 |
| Hidden dropout | 0.2 |
| Attention dropout | 0.2 |
| Output dropout | 0.5 |
| **Biaffine classifier** | |
| Arc scorer dimension | 768 or 1024[a] |
| Label scorer dimension | 256 or 768/1024[b] |
| Dropout | 0.33 |
| **LSTM** | |
| Hidden size | 330 |
| Number of layers | 3 |
| Dropout | 0.5 |
| LSTM learning rate | $5e^{-4}$ |
| **Optimization** | |
| Optimizer | AdamW |
| $\beta_1$, $\beta_2$ | 0.9, 0.999 |
| Weight decay | 0 |
| Batch size | 32 |
| Base learning rate | $4e^{-5}$ |
| LR schedule | Noam |
| LR warmup | 1 epoch |

Table 7.2.: Hyperparameter values. [a]Identical to hidden size of the transformer encoder. [b]256 in factorized models, hidden size of transformer encoder in unfactorized models.

- Batch size (16 / 32)
- Base learning rate ($7e^{-6}$ to $5e^{-5}$)
- Early stopping patience (10 / 15 / 20 epochs)
- Learning rate schedule (constant LR / warmup only / cosine annealing / Noam)

In large part, our final settings are identical to UDify's values with the following differences. We use the AdamW optimizer (Loshchilov and Hutter, 2019b) instead of Adam; we perform neither label smoothing nor gradient clipping; and we do not use differential learning rates. In addition, we do not train for a fixed number of epochs, but instead stop once performance on the validation set does not increase for 15 epochs, or after at most 24 hours.

For model variants involving LSTMs, we tuned the hyperparameters involved in these layers (number of layers; hidden size; dropout; learning rate) in a second round of optimization consisting of 15 trials of random search on the English data. We then picked the two best-performing models and ran them on the other languages, finding that one of them performed best on all languages.

All of our final hyperparameter settings can be found in Table 7.2.

| | ar PADT | cs PDT | de GSD | en EWT | fi TDT | hi HDTB | it ISDT | ja GSD | ko Kaist | lv LVTB | ru STR | zh GSD |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UDPipe+ | 84.62 | 92.56 | 84.06 | 90.40 | 89.49 | 92.50 | 93.38 | **94.27** | 87.54 | 84.50 | 93.68 | 86.74 |
| UDify | 82.88 | 92.88 | 83.59 | 88.50 | 82.03 | 91.46 | 93.69 | 92.08 | 84.52 | 85.09 | 93.13 | 83.75 |
| $\text{UDify}_{\text{mono}}$ | 83.34 | 91.58 | 84.28 | 89.52 | 86.74 | 91.44 | 93.14 | 92.14 | 86.45 | 85.45 | 92.32 | 82.95 |
| $\text{Trankit}_{\text{large}}$ | 86.51 | 93.11 | **86.27** | 91.64 | 94.31 | 93.17 | 94.63 | 78.14 | 40.76 | 91.76 | 95.16 | 87.38 |
| $\text{STEPS}_{\text{mBERT}}$ | 83.80 | 92.69 | 84.08 | 89.16 | 88.91 | 91.30 | 93.13 | 92.22 | 24.49 | 85.05 | 93.74 | 84.94 |
| $\text{STEPS}_{\text{langBERT}}$ | **86.60** | 92.99 | 85.87 | **91.98** | 93.57 | 91.16 | 94.25 | 92.98 | 84.56 | 82.61 | 94.44 | 86.20 |
| $\text{STEPS}_{\text{XLM-R}}$ | 86.55 | **94.58** | 86.07 | 91.91 | **94.36** | **93.34** | **94.86** | 94.10 | **89.93** | **91.93** | **95.30** | **87.75** |
| $\text{STEPS}_{\text{XLM-R-LSTM}}$ | 86.41 | 94.52 | 86.20 | 91.58 | 93.92 | 93.28 | 94.57 | 94.01 | 89.91 | 91.61 | 95.27 | 86.96 |
| $\text{STEPS}_{\text{XLM-R-unfact}}$ | 86.32 | 94.38 | 86.19 | 91.57 | 94.11 | 93.21 | 94.58 | 93.58 | 89.86 | 91.76 | 95.17 | 87.47 |

Table 7.3.: Labeled Attachment Score (LAS) for **basic dependency parsing** varying input embeddings and architecture. STEPS scores are averages of three runs.

### 7.4.3. Impact of Pre-Trained Word Embeddings

We first evaluate how parsing performance differs when varying the underlying pre-trained language model. We here do not include an LSTM layer and perform only dependency parsing. Table 7.3 shows results for all 12 treebanks used in this study. UDPipe+ refers to the version of UDPipe enhanced with BERT and Flair embeddings proposed by Straka et al. (2019) and described in Sec. 7.2. UDify refers to the original system trained on all UD languages without treebank-specific fine-tuning. As multilingual training usually results in improved performance for low-resource languages at the cost of lowering scores for high-resource languages (Üstün et al., 2020), for meaningful comparison, we train $\text{UDify}_{\text{mono}}$ on single treebanks. $\text{Trankit}_{\text{large}}$ refers to the version of Trankit which uses XLM-R-large as the underlying language model, same as $\text{STEPS}_{\text{XLM-R}}$.

$\text{STEPS}_{\text{mBERT}}$ roughly corresponds to $\text{UDify}_{\text{mono}}$, and indeed the models overall perform similarly. We attribute differences to slightly different training setups. While UDify is trained for 80 epochs, STEPS employs early stopping after 15 epochs without improvement. Moreover, we did not disable multi-task learning for parallel UD feature prediction in $\text{UDify}_{\text{mono}}$, and this may be an explanation for why $\text{STEPS}_{\text{mBERT}}$ does much better on Finnish, Czech and Russian, where morphological features may be harder to predict. (For a principled comparison of multi-task setups, see Sec. 7.4.6.) By contrast, UDPipe+ often outperforms UDify, $\text{UDify}_{\text{mono}}$, and $\text{STEPS}_{\text{mBERT}}$, which is likely due to the fact that it trains its own word embeddings in addition to mBERT and additionally makes use of character-level representations via GRUs.

Parsing accuracy of STEPS is very high across the board, with new state-of-the-art results being achieved on all languages except Japanese and German. For most languages, the best results are achieved using $\text{STEPS}_{\text{XLM-R}}$, with $\text{STEPS}_{\text{langBERT}}$ coming in second. In contrast, using mBERT is not the best option on any treebank. In fact, the only

languages for which mBERT achieves better results than langBERT in our experiments are Latvian and Hindi.[3] While using langBERT usually yields worse parsing accuracy than XLM-R, results are roughly on par for Arabic and English. We note that the language-specific models we chose for these treebanks (ArabicBERT-large and RoBERTa-large, respectively) are the only ones with a number of trainable parameters similar to XLM-R, while all others have a considerably smaller number of parameters. This highlights the importance of model size in pre-trained word embeddings.

$\text{STEPS}_{\text{XLM-R}}$ and $\text{Trankit}_{\text{large}}$ show rather similar performance overall, which is to be expected given the fact that both are built on the same underlying language model (XLM-R-large). The slight advantage for $\text{STEPS}_{\text{XLM-R}}$ observed on most languages may stem from the fact that it fine-tunes the entire transformer model instead of merely adding Adapter layers, and that it does not use a multi-task training setup (cf. Sec. 7.4.6). Interestingly, on Finnish and Latvian, both systems outperform other existing parsers by very large margins (around 4.9 and 6.5 LAS, respectively). We assume that there are two main reasons for this. First, XLM-R is pre-trained on CommonCrawl data (Conneau et al., 2020; Wenzek et al., 2020) as opposed to Wikipedia dumps, which results not only in several orders of magnitude more training data (over 1 billion tokens for both languages), but also presumably more heterogenous data, which may provide better generalizations for the domains in our test data.[4] Second, XLM-R has a much larger vocabulary size than mBERT (250k vs. 100k), which means it may account better for the rich morphology of these languages. On average, a Finnish (Latvian) token is split up into 2.4 (2.1) word pieces when using mBERT, but only 1.9 (1.8) word pieces when using XLM-R.

Finally, we note that $\text{STEPS}_{\text{mBERT}}$ and $\text{Trankit}_{\text{large}}$ perform extremely poorly on Korean (24.49/40.76 LAS on average), indicating that the models do not properly learn from the data. We assume that this may be a tokenization or character encoding issue unique to the Korean-Kaist treebank.[5] However, a similar pattern is not observed for any of the other parser models, and we were unfortunately unable to identify the exact

[3] In a similar study comparing mBERT- and langBERT-based parsers, Kanerva et al. (2020) also found Latvian to be one of the few languages for which mBERT outperformed the language-specific (WikiBERT) version. Both the Latvian and the Hindi Wikipedias are rather small, consisting of only 21M and 35M tokens, respectively (Pyysalo et al., 2020).

[4] Both fi-TDT and lv-LVTB contain, among others, "non-standard" data such as blog entries, legal texts, and spoken language (Haverinen et al., 2014; Pretkalniņa et al., 2018).

[5] As pointed out by an anonymous IWPT reviewer for our publication (Grünewald et al., 2021a), Korean-Kaist uses a rather different tokenization strategy than other UD treebanks, with tokens corresponding to larger chunks. Relying on just the first word pieces for token embeddings may be problematic in this context.

cause despite our best efforts.

### 7.4.4. Impact of LSTM Layer

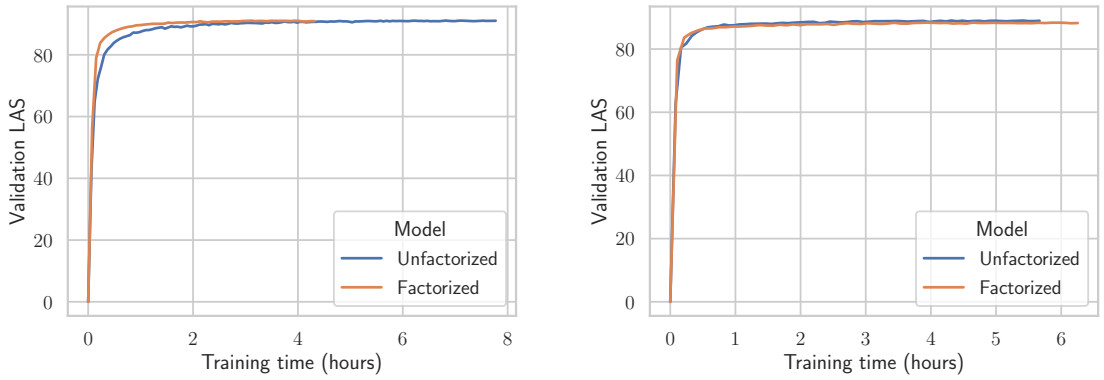
We evaluate the performance of a system identical to $\text{STEPS}_{\text{XLM-R}}$ described above, but with 3 additional BiLSTM layers added on top of the language model ($\text{STEPS}_{\text{XLM-R-LSTM}}$ in Table 7.3). Changes in performance are generally small. With the exception of German, including LSTM layers actually decreases parsing accuracy slightly. The LSTM model contains more trainable parameters and also makes use of differential learning rates, yet, we did not find any meaningful differences in convergence speed and training times. Hence, we conclude that when fine-tuning an underlying transformer-based language model, adding LSTM layers on top is not necessary. However, results may differ for systems that additionally train their own token embeddings or make use of character-based representations, both of which we do not address in our experiments.

### 7.4.5. Impact of Factorization

Dozat and Manning (2018) show that for semantic dependency graph parsing, a simplified parser architecture predicting edge presence and edge labels from the same scoring matrix achieves largely identical results compared to a model using two separate classifiers for arcs and labels. We here dive into the question whether such an unfactorized approach is also able to achieve competitive results in syntactic tree parsing. We do so by implementing a version of $\text{STEPS}_{\text{XLM-R}}$ that makes use of the unfactorized approach as descibed in Sec. 7.3.2.

Results of our experiments can be found in the row labeled $\text{STEPS}_{\text{XLM-R-unfact}}$ in Table 7.3. Overall, performance of the unfactorized approach is very close to the factorized version, but slightly lower. While this shows that the unfactorized approach is indeed viable for tree parsing, analysis of the training times reveals an increase by ca. 30 % on average when using the unfactorized model, indicating that the shared scorer takes a longer time to converge. However, as shown in Figure 7.2, there also exist some treebanks (such as German) for which the two approaches are equally fast.

In light of these results, we propose to stick with the factorized version for syntactic tree parsing. At least in a research setting, shorter training times allow for a larger set of experiments and thus ultimately in using fewer resources. When applying the parser, differences in model size and parsing time are negligible.




(a) Latvian. (b) German.

Figure 7.2.: Learning curves (validation LAS as a function of training time) for factorized vs. unfactorized systems in Latvian and German.

### 7.4.6. Impact of Multi-Task Approach

Finally, we analyze how performance changes when predicting UPOS and UFeats in addition to dependencies. For these experiments, we use XLM-R as input embeddings and a factorized architecture. For UFeats, we follow UDify's approach and consider each possible combination of morphological features a unique label. As shown in Table 7.4, $\text{STEPS}_{\text{MTL}}$ achieves very high accuracies for UPOS and UFeats, performing on par with or only slightly worse than the previous state of the art ($\text{Trankit}_{\text{large}}$) for most languages. However, we find that compared to the dependency-only system, parsing accuracy drops considerably in the multi-task setting (up to over 1 LAS for Finnish).

During training of $\text{STEPS}_{\text{MTL}}$, accuracy on the validation set increased very rapidly for the tagging tasks and reached levels close to the final values after only a few epochs, while accuracy for the parsing task increased much slower. This suggests that the loss for the tagging tasks might overwhelm the system as a whole, causing the parser modules to underfit. We therefore also test $\text{STEPS}_{\text{MTLscale}}$, in which the loss for UPOS and UFeats is scaled down to 5% during training. $\text{STEPS}_{\text{MTLscale}}$ performs close to $\text{STEPS}_{\text{MTL}}$, even outperforming it in the case of Hindi. In turn, however, accuracy for UPOS and particularly UFeats drops considerably.

To sum up, our experiments indicate that multi-task setups as commonly employed in UD parsing have a non-negligible effect on parsing performance. Hence, when comparing parser performance, it is crucial to take potential multi-task setups into account. If the respective setups differ, ignoring them may result in misleading interpretations of parsing performance of model architectures (unless the variable of interest is the multi-task setup

| Treebank | Model | UPOS | UFeats | UAS | LAS |
|---|---|---|---|---|---|
| Arabic (ar_padt) | $\text{Trankit}_{\text{large}}$ | 95.47 | **95.54** | 90.90 | 86.51 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **90.96** | **86.55** |
| | $\text{STEPS}_{\text{MTL}}$ | **97.24** | 94.89 | 90.34 | 86.01 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 96.47 | 87.74 | 90.80 | 86.41 |
| Czech PDT (cs_pdt) | $\text{Trankit}_{\text{large}}$ | 99.37 | **98.23** | 95.51 | 93.11 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **95.89** | **94.58** |
| | $\text{STEPS}_{\text{MTL}}$ | **99.41** | 98.06 | 95.59 | 94.19 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 98.97 | 94.20 | 95.85 | 94.52 |
| German (de_gsd) | $\text{Trankit}_{\text{large}}$ | **95.48** | **91.91** | **90.11** | **86.27** |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | 90.02 | 86.07 |
| | $\text{STEPS}_{\text{MTL}}$ | 95.40 | **91.91** | 89.53 | 85.46 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 94.65 | 83.33 | 89.74 | 85.80 |
| English (en_ewt) | $\text{Trankit}_{\text{large}}$ | **97.91** | **98.04** | 93.59 | 91.64 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **93.90** | **91.91** |
| | $\text{STEPS}_{\text{MTL}}$ | 97.84 | 98.02 | 93.47 | 91.50 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 96.58 | 96.49 | 93.80 | 91.78 |
| Finnish TDT (fi_tdt) | $\text{Trankit}_{\text{large}}$ | **98.72** | **97.07** | 95.55 | 94.31 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **95.69** | **94.36** |
| | $\text{STEPS}_{\text{MTL}}$ | 98.52 | 96.75 | 94.62 | 93.11 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 98.19 | 88.70 | 95.59 | 94.26 |
| Hindi (hi_hdtb) | $\text{Trankit}_{\text{large}}$ | **98.12** | 93.98 | 96.16 | 93.17 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | 96.11 | 93.34 |
| | $\text{STEPS}_{\text{MTL}}$ | 98.09 | **94.49** | 95.96 | 93.03 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 97.51 | 88.60 | **96.18** | **93.39** |

| Treebank | Model | UPOS | UFeats | UAS | LAS |
|---|---|---|---|---|---|
| Italian (it_isdt) | $\text{Trankit}_{\text{large}}$ | 98.80 | 98.43 | 95.93 | 94.63 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **96.25** | **94.86** |
| | $\text{STEPS}_{\text{MTL}}$ | **98.81** | **98.58** | 95.80 | 94.36 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 98.43 | 94.28 | 96.09 | 94.69 |
| Japanese (ja_gsd) | $\text{Trankit}_{\text{large}}$ | 92.57 | 97.58 | 86.62 | 78.14 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **95.62** | **94.10** |
| | $\text{STEPS}_{\text{MTL}}$ | **98.21** | **99.98** | 95.38 | 93.78 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 96.94 | 99.91 | 95.53 | 94.00 |
| Korean (ko_kaist) | $\text{Trankit}_{\text{large}}$ | 69.86 | 98.95 | 67.96 | 40.76 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **91.71** | **89.93** |
| | $\text{STEPS}_{\text{MTL}}$ | **96.41** | **100.00** | 91.48 | 89.73 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 93.90 | **100.00** | 91.53 | 89.77 |
| Latvian (lv_lvtb) | $\text{Trankit}_{\text{large}}$ | **97.83** | **95.38** | 94.19 | 91.76 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **94.32** | **91.93** |
| | $\text{STEPS}_{\text{MTL}}$ | 97.73 | 94.79 | 93.44 | 90.99 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 96.72 | 81.42 | 93.97 | 91.61 |
| Russian (ru_syntagrus) | $\text{Trankit}_{\text{large}}$ | **99.34** | **98.47** | 96.18 | 95.16 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **96.32** | **95.30** |
| | $\text{STEPS}_{\text{MTL}}$ | 99.27 | 98.34 | 95.94 | 94.88 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 98.99 | 95.91 | 96.19 | 95.20 |
| Chinese (zh_gsd) | $\text{Trankit}_{\text{large}}$ | 96.83 | 99.48 | 90.03 | 87.38 |
| | $\text{STEPS}_{\text{dep-only}}$ | – | – | **90.72** | **87.75** |
| | $\text{STEPS}_{\text{MTL}}$ | **97.20** | **99.50** | 89.70 | 86.86 |
| | $\text{STEPS}_{\text{MTLscale}}$ | 95.21 | 98.53 | 90.31 | 87.39 |

Table 7.4.: Results for **basic dependency parsing** vs. **parsing and feature prediction** (**multi-task**) for $\text{STEPS}_{\text{XLM-R}}$. Scores are averages of three runs. For UPOS and UFeats, we report accuracy.

itself).

### 7.4.7. Summary

Our experimental findings can be summarized as follows: (a) Choice of pre-trained embeddings has the greatest impact on parser performance, with XLM-R yielding the best results in most cases; (b) adding LSTM layers is not necessary when working with a large fine-tuned language model; (c) a factorized parser architecture is preferable due to faster training; (d) when using a multi-task approach incorporating UPOS and UFeats prediction, there is a tradeoff between tagging and parsing accuracy, and conclusions regarding architecture should be drawn by comparing experiments performed in the same setting. Crucially, one of the simplest parsers in our evaluation ($\text{STEPS}_{\text{XLM-R}}$) achieves the best results overall, often surpassing more complex previous work.

## 7.5. Conclusion and Future Work

In this chapter, we have addressed **RQ4 (Which are the main factors determining the performance of neural graph-based dependency parsers?)** and performed

a detailed and principled analysis on a variety of decisions arising during dependency parser design. In general, our findings support the notion that painstakingly optimized system architectures are not a necessity in the presence of high-quality pre-trained word representations, as evidenced by the lacklustre performance of models utilizing additional LSTM layers and multi-task training. By contrast, the choice of such word representations / language models is crucial for parsing accuracy, as shown by the large performance gaps between the different transformer models examined in our study.

In short, the answers to RQ4 arising from our study can be summed up as follows:

- **What works?** We have identified an architecture based on fine-tuned XLM-R embeddings and factorized scoring that lead to new state-of-the-art performance for 11 out of 12 diverse language in our study on basic UD parsing.

- **What doesn't?** Adding LSTM layers on top of the transformer leads to a decrease in accuracy in most cases, likely because they are superfluous in the presence of transformer language models. We have also shown that multi-task setups predicting UPOS and UFeats often degrade parsing performance.

- **What is really necessary?** For current state-of-the-art UD parsers, we recommend making sure that the pre-trained language model covers the intended domain well. In addition, keeping a factorized approach is a good idea for tree parsing.

A number of follow-up questions arises from our results. First, we have only addressed a high- to medium-resource scenario in our study, assuming that we know the application language of a parser and thus are training a single parser per language. Future work may address other approaches such as a massively multilingual training setup (as used by UDify) or the recently proposed UDapter (Üstün et al., 2020), which aims at boosting performance of low-resource languages while keeping performance of high-resource languages high.

Second, we have only examined parsing into the basic UD representation in our study, raising the question to what extent our results transfer to enhanced UD. This corresponds to **RQ5 (How can neural dependency parsers be adapted for Enhanced UD?)**, which is addressed by our adaptation of STEPS to enhanced UD as described in Chapter 8.

Third, relating to **RQ6 (Can we leverage the biaffine dependency parser architecture for other NLP applications?),** there is the question whether our results on architectures for syntactic dependency parsing also hold for non-syntactic tasks that have recently been framed as dependency parsing tasks, such as Named Entity

Recognition (Yu et al., 2020) or Semantic Role Labeling (Shi et al., 2020). This question is addressed in Chapter 9 and Chapter 10, which examine dependency parsing formulations for Negation Resolution and Relation Extraction.

In conclusion, in this chapter we have applied "Occam's razor" to graph-based dependency parsing. We believe that the insights from our study will foster further research on dependency parsing and on framing other tasks as dependency parsing, taking our simplified but robustly performing STEPS parser as a starting point.

# 8. Enhanced UD Parsing with STEPS

## 8.1. Introduction

In this chaper, we address **RQ5 (How can neural dependency parsers be adapted for Enhanced UD?).** To do so, we take the state-of-the-art STEPS parser (as described in Chapter 7) and investigate ways of adapting it to output Enhanced UD graphs. For empirical analysis, we describe our results in the in the IWPT 2020 and IWPT 2021 Shared Tasks on parsing from raw text into Enhanced UD (Bouma et al., 2020, 2021), in which we participated with the resulting adapted parser.[1]

As described in Sec. 3.3, Enhanced UD extends UD representations by including additional relations for several linguistic phenomena such as relative clauses, thereby turning trees into graphs. As a result, unique challenges that are not present in basic UD parsing arise when parsing text into Enhanced UD graphs. Existing work on inducing Enhanced UD graphs prior to the Shared Tasks (Nyblom et al., 2013; Simi and Montemagni, 2018; Nivre et al., 2018) inferred Enhanced UD representations by a pipeline approach of first parsing text into basic UD trees and then adding enhanced relations by applying rule-based or machine-learning modules. This approach has the disadvantage of propagating errors in the basic layer to the enhanced parse.

By contrast, with our adaptation of the STEPS parser, we propose to treat Enhanced UD parsing using an end-to-end paradigm and do not distinguish between the basic UD tree and the enhanced part of the graph. Instead, we treat all types of dependencies equally and extract them jointly, using STEPS's modular architecture based on the biaffine classifier architecture and fine-tuned contextualized word embeddings as a starting point and adapting it to produce general graphs rather than tree structures.

We overcome the problem of sparsity issues caused by Enhanced UD's large lexicalized label set by replacing lexical items with placeholders at prediction time and later retrieving them from the full parse using either a strictly rule-based algorithm (in the 2020 edition of

[1]The text of this chapter is based on Grünewald and Friedrich (2020a) and Grünewald et al. (2021b). Frederik Oertel implemented the label transducer for our IWPT 2021 system.

| Submission | Score |
|---|---|
| 1. RobertNLP (ours) | **88.94** |
| 2. TurkuNLP | 87.15 |
| 3. EmoryNLP | 85.30 |
| *Median* | 82.87 |

Table 8.1.: Overview of IWPT 2020 results (avg. ELAS F1 score on English). Full results can be found at `https://universaldependencies.org/iwpt20/Results.html`.

| Submission | Score |
|---|---|
| 1. TGIF | **89.24** |
| 2. ShanghaiTech | 87.07 |
| 3. RobertNLP (ours) | 86.97 |
| *Median* | 83.64 |

Table 8.2.: Overview of IWPT 2021 results (avg. ELAS F1 score on 17 languages). Full results can be found at `https://universaldependencies.org/iwpt21/results.html`.

the Shared Task) or a hybrid rule-based/machine-learning classifier (in the 2021 edition). Despite being conceptually straightforward and easy to implement, our system achieves a highly competitive performance, scoring 1$^{st}$ (English only) in the IWPT 2020 Shared Task and 3$^{rd}$ (overall) in the IWPT 2021 Shared Task on the blind test data according to the official ELAS evaluation metric (see Table 8.1 and Table 8.2).

In summary, this chapter's contributions are as follows:

1. We adapt STEPS, our state-of-the-art graph-based neural dependency parser, to Enhanced UD parsing.

2. We introduce and evaluate a number of novel rule-based and machine learning-based strategies for lexicalizing dependency labels, and demonstrate their effectiveness.

3. We further demonstrate that additional improvements can be achieved via model ensembling.

4. We demonstrate that our system achieves competitive performance by reporting its results in the IWPT 2020 and 2021 Shared Tasks on Enhanced UD parsing, where it scored 1$^{st}$ on English and 3$^{rd}$ overall, respectively.

## 8.2. IWPT20: System Description

This section describes the configuration of the version of the STEPS parser submitted to the IWPT 2020 Shared Task.

### 8.2.1. Pre-processing

For tokenization and sentence segmentation, we employ the StanfordNLP system (Qi et al., 2018), which achieved state-of-the-art results for these tasks on the English treebanks in the CoNLL 2018 Shared Task.

### 8.2.2. Input Token Representation

We retain the STEPS parser architecture described in Sec. 7.3 and generate contextualized word embeddings for the tokens of the input sentence as a weighted sum of the internal layers of a transformer language model. Since our model is English-only, we use RoBERTa (Liu et al., 2019) as the underlying pre-trained language model, fine-tuning it while training our parser. We also experimented with using BERT (Devlin et al., 2019) instead of RoBERTa, but found that this yielded lower parsing accuracy (see Sec. 8.3.3).

### 8.2.3. Dependency Classification

Figure 8.1 shows an overview of our neural-network based dependency classifier, which simultaneously predicts relation labels or absence of a relation between pairs of tokens.

**Classifier architecture.** We stick with STEPS's biaffine architecture (Dozat and Manning, 2018, see also Sec. 6.2), which is capable of producing general (bi-lexical) dependency graph structures. Because we are predicting enhanced graphs and not basic trees, we use an unfactorized classifier in our main configuration. We also tried using a factorized classifier, but found that this yielded slightly lower parsing accuracy (see Sec. 8.3.3).

**De-lexicalizing dependency labels.** Because Enhanced UD adds lexical information to certain dependencies (e.g. *obl:instead_of*), the number of dependency labels is huge; the EWT corpus contains 399 unique labels. To avoid sparsity issues, we strip lexical information from labels during training, instead replacing them with placeholders (e.g. *obl:[case]*) indicating where in the dependency graph the lexical information is expected to be found (see Sec. 8.2.4 for a detailed description of the reconstruction process). This

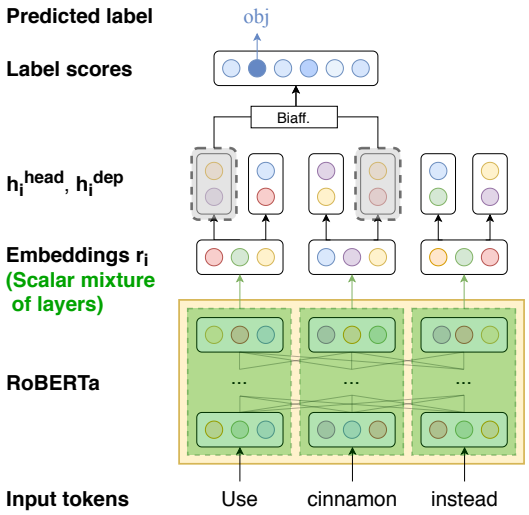



Figure 8.1.: Architecture of neural network predicting dependency relations between pairs of tokens.

way, we can remove all lexicalized relations from the label vocabulary, instead adding only five new placeholder labels: *nmod:[case]*, *obl:[case]*, *acl:[mark]*, *advcl:[mark]*, and *conj:[cc]*. We keep all other, non-lexicalized subtyped labels (such as *nmod:poss*). This brings the total label count down to 56 (including ∅).

### 8.2.4. Post-processing

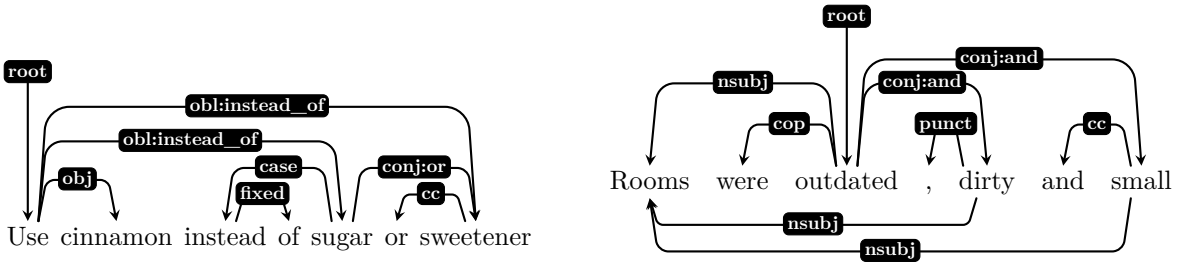
The outputs $P(y_{i,j})$ provided by the dependency classifier can be regarded as a 3-dimensional tensor, with one dimension corresponding to the tokens as heads, one dimension corresponding to the tokens as dependents, and the third dimension corresponding to the label set. Figure 8.2 gives a two-dimensional view of this tensor, with each cell containing the highest-scoring label for a head (row label) and dependent (column label) pair.

**Ensuring graph structure constraints.** Using the outputs provided by the dependency classifier, we can assemble a dependency graph by retrieving the highest-scoring dependency (or ∅, i.e., no relation) for each pair of tokens in the sentence (omitting the diagonal as enhanced UD does not allow links starting and ending at the same node) and forming their union.

Although enhanced UD eliminates the requirement that dependency graphs must be trees, it maintains a set of structural constraints. Specifically, each token needs to have at

| | [root] | Use | cinnamon | instead | of | sugar | or | sweetener |
|---|---|---|---|---|---|---|---|---|
| **[root]** | ∅ | *root* | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |
| **Use** | ∅ | ∅ | *obj* | ∅ | ∅ | *obl:[case]* | ∅ | *obl:[case]* |
| **cinnamon** | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |
| **instead** | ∅ | ∅ | ∅ | ∅ | *fixed* | ∅ | ∅ | ∅ |
| **of** | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |
| **sugar** | ∅ | ∅ | ∅ | *case* | ∅ | ∅ | ∅ | *conj:[cc]* |
| **or** | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |
| **sweetener** | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | *cc* | ∅ |

Figure 8.2.: Prediction matrix of the dependency classifier. Cell entries show the highest-scoring label for each ordered pair of tokens, with row/column labels indicating potential heads/dependents respectively.




(a) Dependency graph with lexicalization of labels.

(b) Dependency graph with "conjunction siblings."

Figure 8.3.: Re-lexicalization of dependency labels in the presence of conjunctions.

least one head and must be reachable from at least one of the root(s) of the graph.[2] These global constraints are not automatically adhered to by our simple graph construction method, which operates on pairs of tokens. Nonetheless, we observe that around 99 % of sentences are assigned structurally valid graphs as determined by the official validation script.[3]

To make the graphs of the remaining 1 % sentences structurally valid, we perform the following steps. In the case of tokens lacking a head, the ∅ label has received the highest score during classification for all possible heads. We now simply retrieve all second-ranked labels and their scores and pick the relation (and corresponding head) that received the highest score across all possible heads.

[2]Graphs in enhanced UD may have more than one root.

[3]`https://github.com/UniversalDependencies/tools/blob/master/validate.py`

Further, in order to ensure reachability from the root, in the cases violating this constraint, we fall back to an external dependency tree parse, i.e., a representation of the UD basic layer, for generating candidate links to be added to our graph. We here use the UDify parser (Kondratyuk and Straka, 2019) to predict basic UD trees. We determine the set of nodes $V$ that are not reachable from any root, and for each node $v \in V$ we compute the number of nodes in $V$ that can be reached when starting at $v$. We then pick the node $v_i$ that can reach the largest number of nodes and check if the head of $v_i$ in the basic layer tree can be when starting at a root in *our* graph. If so, we add the relation between $v_i$ and its head as present in the basic layer tree to our graph, otherwise, we add a *dep* edge from the sentence's first root to $v_i$. We repeat this procedure until each node in the graph is reachable from at least one root node.

**Re-lexicalization of labels.** As outlined in Sec. 8.2.3, lexical information is stripped from dependency labels during training, using the format *base:[placeholder]*. At prediction time, we re-lexicalize predicted placeholder labels using the following set of rules.

First, if the token has a dependent that is attached via the placeholder of the de-lexicalized relation in question, we lexicalize the relation with the token of this dependent. For example, in Figure 8.3a, our parser predicts *obl:[case]* and we re-lexicalize this relation with the token(s) of the *case* dependents of "sugar." (Multiword expressions, such as "instead of", are handled by concatenating word forms linked by the *fixed* relation.)

If such a dependent does not exist, it may be due to the presence of a *conj* relation. For example, Figure 8.3a shows a case where for the de-lexicalized link *obl:[case]* ending at "sweetener," no *case* relation starts at this node. This is due to the presence of a *conj* relation, ending at "sweetener." We hence check if the head of the *conj* relation has an incoming lexicalized edge of the same base relation (here *obl*) and if so, re-lexicalize accordingly.

Similarly, *conj* links ending at siblings in coordinate constructions (here "dirty" and "small") are always lexicalized with the same item (in this case "and"). Unlike "small," the dependent "dirty" does not have its own *cc* dependent that could be used to execute the first step, i.e., to replace the placeholder of *conj:[cc]* with a dependent's token. For such nodes, we hence search the graph for siblings that are linked to the common governor via *conj* relations. If we find any, we use the lexicalized label of the corresponding *conj* relation for all siblings.

The above heuristics return a result for 98.9 % of the de-lexicalized relations predicted for the blind test data; in the remaining cases, we simply remove the placeholder without

substituting any lexical material. Provided that the underlying base relation was predicted correctly, we are able to retrieve the correct lexical material for 98.4 % of relations.

**Removal of relations.** In addition, UD contains several relations that empirically only appear on their own, i.e., whose dependent may have only one incoming edge of this type. These relations are *fixed*, *flat*, *goeswith*, *punct*, and *cc*. However, in around 0.4 % of cases our parser erroneously predicts several of these relations for a single token (e.g., punctuation being attached to several tokens at once). In these cases, we remove all but the most confidently predicted dependency.

## 8.3. IWPT20: Evaluation

This section describes the results of our main submission to the IWPT 2020 Shared Task, as well as a number of additional experiments we conducted to contextualize our results.

### 8.3.1. Experimental Settings and Hyperparameters

We use the training and development sections of the EWT corpus for training and validation, respectively. We use gold-tokenized and gold-segmented sentences as input for our system during training.

For hyperparameter settings, we mostly stick with the values used by Kondratyuk and Straka (2019). An exception to this is the training regime, where we found a low batch size, constant learning rate, no gradient clipping, and the AdamW optimizer (Loshchilov and Hutter, 2019b) to yield the best results. The final hyperparameters can be found in Table 8.3.

Our model was trained using a single nVidia Tesla V100 GPU, stopping early when ELAS F1 score on the development set did not improve for 10 epochs. The best model was found after 63 epochs, i. e., 73 training epochs were performed in total, taking ca. 9 hours. Parsing the English blind test set (3077 sentences) takes around 3 minutes in total, i.e., 0.06 seconds per sentence.

### 8.3.2. Results of Submission

Table 8.4 shows the results (in terms of ELAS F1 score) on the blind English test data for our system as well as the highest- and lowest-ranking competing submissions and the median submission. Our system (submitted as “RobertNLP”) achieves an ELAS F1 score

| | |
|---|---|
| **RoBERTa embeddings** | |
| Embeddings dimension | 1024 |
| Token mask probability | 0.15 |
| Layer dropout | 0.1 |
| Hidden dropout | 0.2 |
| Attention dropout | 0.2 |
| Output dropout | 0.5 |
| **Biaffine classifier** | |
| Hidden size | 1024 |
| Dropout | 0.33 |
| **AdamW Optimizer** | |
| Batch size | 5 |
| Learning rate | $5e^{-6}$ |
| $\beta_1$, $\beta_2$ | 0.9, 0.999 |
| Weight decay | 0.0 |

Table 8.3.: Hyperparameter values.

| Submission | IWPT-all | EWT | PUD |
|---|---|---|---|
| RobertNLP | **88.94** | **88.06** | **89.97** |
| TurkuNLP | 87.15 | 86.14 | 88.35 |
| *median* | 83.41 | 82.04 | 85.02 |
| Køpsala | 65.37 | 64.18 | 66.77 |
| UDify + converter | 85.67 | 84.55 | 87.00 |

Table 8.4.: Parsing results (ELAS F1) on English blind test data in the IWPT 2020 Shared Task.

of 88.94 %, ranking first with a margin of more than 1.5 points over the second-ranking submission.

As an additional baseline, we used the state-of-the-art UDify parser (Kondratyuk and Straka, 2019) to predict basic dependencies and then ran the rule-based converter by Schuster and Manning (2016) on the output to extract enhanced relations. This approach achieved an F-Score of 85.67 %, considerably lower than our system, confirming that our end-to-end graph parsing approach is superior to a pipeline model of basic parsing + rule-based conversion.

### 8.3.3. Analysis of Results

We here describe several experiments using variations of the setting used in our official submission. These experiments aim at determining the impact of different factors, including choice of pre-trained embeddings, training data, as well as segmentation and tokenization, on model performance. Some of the experiments described in this section

| Embeddings | Train | IWPT-test | EWT-dev |
|---|---|---|---|
| BERT-base | EWT | 87.49 | 87.64 |
| RoBERTa-base | EWT | 88.17 | 88.64 |
| BERT-large | EWT | 88.18 | 88.61 |
| RoBERTa-large | EWT | **88.94**[a] | **89.43** |
| RoBERTa-large | UD2.5[b] | 87.85 | 88.59 |

Table 8.5.: Effect of embeddings and training data on model performance (ELAS F1, English blind test data). [a]Official submission. [b]Concatenation of EWT, GUM, LinES, and ParTUT training data.

were conducted during the development of our system, others constitute post-evaluation analyses. For consistency, we present results for the blind test data in this section. Most experiments were initially conducted using the development data, showing the same tendencies.

**Choice of pre-trained embeddings.** We experiment with four different pre-trained embedding models, namely BERT and RoBERTa in their base and large variants respectively. As shown in Table 8.5, RoBERTa outperforms BERT, and the large variants outperform the base variants, with BERT-large and RoBERTa-base performing roughly equally. The best observed results are achieved by RoBERTa-large (our official submission). The superior performance of RoBERTa may stem from the fact that it was pre-trained on a considerably larger amount of data, and that it dropped the "next sentence prediction" objective, which may be irrelevant or even detrimental for a single-sentence task like syntactic parsing.

**Effect of additional training data.** While preparing our submission, we experimented with generating additional training data by using the rule-based UD enhancer by Schuster and Manning (2016), which was used to create the gold standard enhanced layers of the EWT and PUD corpora, to build enhanced versions of three other English UD treebanks (GUM, LinES, and ParTUT).

However, we found in preliminary experiments on the dev and test sections of the above mentioned corpora that including this additional training data actually slightly hurts performance if the test data is from a different corpus. This is correlated with the lexical distance between test and training data as computed using the Bhattacharyya distance

$$D_B(p, q) = -\ln \sum_{x \in X} \sqrt{p(x)q(x)} \tag{8.1}$$

| Corpus | Lex. dist. | ELAS F1 |
|---|---|---|
| EWT | 0.142 | 88.94 |
| GUM | 0.204 | 87.98 |
| LinES | 0.240 | 88.20 |
| ParTUT | 0.248 | 87.69 |

Table 8.6.: Lexical (Bhattacharyya) distance and parsing accuracy between the blind test data and the different training corpora. The rightmost column indicates parsing performance on the IWPT test set when adding the respective corpus to the EWT training data. (First line is EWT only.)

between the respective vocabulary probability distributions (Bhattacharya, 1943; Ruder and Plank, 2017).

As the lexical distance of the blind test set and EWT is much smaller than the ones between the test set and the other corpora (see Table 8.6), our official submission's model was trained only on EWT. Post-evaluation experiments (see rightmost column) confirm that when including corpora with higher lexical distance, parsing accuracy decreases. In addition, parsing results on the blind test set when including all additional data (results see last line in Table 8.5) confirm this approach. However, if a different test set showed greater similarity to other corpora, including them as training data would likely be beneficial. As one of the anonymous reviewers points out, in addition to lexical similarity, factors such as mean dependency distance or average sentence length may also play a role. In conclusion, our experiments once more highlight that selecting good training corpora for an application domain is a critical factor and an interesting direction for further research.

**Effect of segmentation and tokenization.** While our parser was trained on gold-tokenized and gold-segmented sentences, the Shared Task required parsing from raw text. In order to determine the extent to which automatic segementation and tokenization impacts results, we run our parser on the gold-tokenized and gold-segmented version of the test data.

We observe an ELAS F1 score of 90.80, which constitutes an increase of nearly 2 points over the results obtained using automatic segmentation. This indicates that our system is rather sensitive to these kinds of errors and would greatly benefit from improvements in segmentation accuracy. It might also be possible to increase the robustness of our system w. r. t. these errors by training it on system-predicted sentence segmentation.

**Performance on basic vs. enhanced relations.** We further evaluate how performance of our parser varies between (a) relations that result from enhancements, i. e., relations which are exclusive to the enhanced layer, and (b) relations that occur in the basic layer as well. Because our parser does not differentiate between basic and enhanced relations internally, we can only compute recall for the two classes, but not precision and F1.[4] We perform this evaluation for gold-segmented and gold-tokenized input.

Recall is considerably lower on relations exclusive to the enhanced layer (83.64 %) as opposed to relations that are also present in the basic layer (91.60 %), indicating that predicting the former is indeed a more difficult task compared to predicting the latter, as might be expected. The result further suggests that it might be promising to use our parser architecture in combination with a spanning tree algorithm to predict basic-layer style trees as well (e. g. in a multi-task setting). This would also eliminate the need to rely on external parser input to post-process dependency graphs for the rare cases of invalid graphs.

**Performance on individual label types.** Finally, while our system achieves a high parsing accuracy overall, we also compute F1-Scores for each individual label type in order to obtain a finer-grained picture of its strengths and weaknesses. Again, we perform this evaluation for gold-segmented and gold-tokenized input. A selection of the results is displayed in Table 8.7.

As might be expected, the label types on which our parser performs best are highly common functional relations such as *det* and *case*, as well as frequent content word dependencies such as *nsubj* and *amod*. More interestingly, it also performs close to the average on *nsubj:xsubj*, which is not only considerably rarer than the aforementioned relations but also exclusive to the enhanced representation, demonstrating that our joint approach is capable of capturing these dependencies as well.

Somewhat more challenging are the *flat* and *compound* labels (85.53 and 83.51 F-Score, respectively), which are used to annotate multiword expressions. The computational identification and treatment of such expressions is very challenging and constitutes a long-standing research area in itself (Gregoire et al., 2007; Savary et al., 2018, 2019).

The *punct* relation harbors perhaps the greatest potential for improvement, yielding an F-Score of only 84.28 despite being extremely common. This is likely due to the rather

[4]We compare to the gold standard which distinguishes between basic and enhanced relations. Our parser does not differentiate between basic and enhanced relations, i.e., the full graph is constructed without internally identifying the subgraph corresponding to the basic syntactic tree.

| Label | Freq. | ELAS F1 |
|---|---|---|
| *det* | 3879 | 99.04 |
| *case* | 4481 | 97.16 |
| *nsubj* | 3708 | 94.78 |
| *amod* | 2552 | 92.49 |
| (Total/avg.) | 48298 | 90.80 |
| *nsubj:xsubj* | 569 | 88.22 |
| *flat* | 482 | 85.53 |
| *punct* | 5519 | 84.28 |
| *compound* | 2005 | 83.51 |
| *appos* | 347 | 66.31 |
| *parataxis* | 301 | 59.35 |
| *list* | 251 | 47.72 |

Table 8.7.: Parsing accuracy (ELAS F1) for a selection of label types. Scores were computed on gold-segmented test data. *Freq.* denotes the number of occurrences of the label in the gold annotations.

complex set of rules that determines which token a piece of punctuation is attached to.[5] However, it might also be the label where improvements are most difficult to achieve, as the gold standard itself contains inconsistencies,[6] leading to a noisy training signal.

Finally, out of all label types which occur more than 200 times in the test data, the worst performance is observed on *appos*, *parataxis*, and *list*. While their low frequency is almost certainly part of the reason for this, it is also worth noting that these dependencies are unusual in that they represent "side-by-side" relations between words rather than more obviously hierarchical structures (as is the case for most other label types). Investigating parser performance on these kinds of constructions in greater detail may present a promising avenue for future work.

**Factorized vs. unfactorized classifier.** After the official deadline, we ran another set of experiments on English-EWT as well as six additional treebanks from other languages in order to determine to what extent our conclusions about factorized vs. unfactorized parsing (see Sec. 7.4.5) hold for Enhanced UD. We modify STEPS to generate dependency graphs using a factorized approach as proposed by Dozat and Manning (2018) for semantic dependency parsing, weighting the losses of the edge and label scorers:

$$\ell = \lambda_{edge}\ell_{edge} + \lambda_{label}\ell_{label}. \tag{8.2}$$

[5] See `https://universaldependencies.org/u/dep/punct.html`.

[6] As noted on the treebank's Github page at `https://github.com/UniversalDependencies/UD_English-EWT`.

| | Turku-NLP | Shanghai-Tech | STEPS$_{\text{XLM-R}}$ fact | STEPS$_{\text{XLM-R}}$ unfact |
|---|---|---|---|---|
| ar-PADT | **77.83** | 77.73 | 77.42 | 77.68 |
| cs-PDT | 88.17 | **90.63** | 89.49 | 89.43 |
| en-EWT | 86.14 | 86.30 | 87.11 | **87.28** |
| fi-TDT | 89.24 | 89.97 | **91.53** | 91.51 |
| it-ISDT | 91.54 | 91.49 | 92.35 | **92.39** |
| lv-LVTB | 84.94 | 87.64 | **89.04** | 88.95 |
| ru-STR | 90.69 | 92.31 | 93.68 | **93.75** |

Table 8.8.: Parsing accuracy (ELAS F1) for factorized vs. unfactorized parsing. Scores were computed on gold-segmented test data. Scores are averages of three runs.

After tuning on English in a set of preliminary experiments, we set the hyperparameters $\lambda_{edge}$ to 1.0 and $\lambda_{label}$ to 0.05.

For evaluation, we use gold tokenization and sentence segmentation and report averages of three model runs. We compare our factorized parser against its unfactorized counterpart, as well as the TurkuNLP and ShanghaiTech systems from the IWPT20 competition. Table 8.8 reports our results. Unlike in tree parsing, the unfactorized system actually slightly outperforms the factorized system on a number of languages, with the largest margins on Arabic and English, thus demonstrating that the unfactorized approach may be well-suited to graph parsing tasks. This is in line with the results of Dozat and Manning (2018).

## 8.4. IWPT21: System Description

For our submission to the 2021 edition of the IWPT Shared Task (Bouma et al., 2021), we adapt our English-only submission from the previous year (see Sec. 8.2) to all 17 languages that are part of the competition. The core principle of our system remains the same, in that we parse directly into enhanced graphs using a biaffine classifier on top of a fine-tuned transformer-based language model. However, we expand our strictly rule-based system for retrieving lexical material for dependency labels at prediction time into a hybrid rule-based/machine-learning system (see Sec. 8.4.5). In order to further increase our parser's accuracy as well as its robustness across treebanks, we also use model ensembling.

The following section describes the configuration of the version of the STEPS parser submitted to the Shared Task in detail.

### 8.4.1. Pre-processing

For tokenization and sentence segmentation, we employ Trankit$_{\text{large}}$ (Nguyen et al., 2021), which achieves state-of-the-art (or near state-of-the-art) results for these tasks on the languages present in the Shared Task. We use the default model for each language.

### 8.4.2. Input Token Representation

We use the transformer-based, multilingual XLM-R$_{\text{large}}$ language model (Conneau et al., 2020) to generate contextualized word embeddings for the tokens of the input sentence, fine-tuning the model while training our parser. We create the wordpiece-tokenized input for XLM-R by feeding each token into the XLM-R tokenizer. In addition, we prepend a special *[root]* token to each sentence, which serves as an artificial head of the *root* relation that must be present in every sentence. This token receives a fixed, learned embedding instead of a contextualized XLM-R embedding, but with the same number of dimensions.

The final embedding $\mathbf{r}_i$ for a token at position $i$ is extracted by forming a weighted sum of the internal XLM-R layers at the position corresponding to the first wordpiece of the original token. Following Kondratyuk and Straka (2019), coefficients for this weighted sum are learned during training, while randomly dropping layers to prevent the model from focusing on only a single layer.

### 8.4.3. Dependency Classification

**Classifier architecture.** We retain our biaffine classifier architecture from our submission to the 2020 edition of the shared task (see Sec. 8.2.3 and Figure 8.1), with the only difference that it takes XLM-R embeddings rather than RoBERTa embeddings as input.

**De-lexicalizing dependency labels.** As in the case of English, the number of possible dependency labels is also very large for most other languages due to Enhanced UD's lexicalized labels. For example, Arabic-PADT has over 1100 unique dependency labels. Among the languages being part of the Shared Task, French is an exception as its treebanks do not make use of lexicalized labels, resulting in a label count of only 48.

As in our 2020 system, we strip lexical information from labels during training, instead replacing them with placeholders to be filled at prediction time (Sec. 8.4.5 describes the reconstruction process). We keep subtype labels that are unrelated to lexical material: For example, in a label such as *nmod:poss*, the *poss* subtype is not a lexical item, but simply indicates a possessive modifier; this, we do not edit the label. Such non-lexical

subtype labels can also occur in conjunction with lexical material, e.g., in Estonian, the label *obl:järgi:gen* indicates a oblique nominal in the genitive case attached via the the preposition "järgi." In this case, we replace the lexical part with a placeholder but keep the rest of the label, i.e, *obl:järgi:gen* becomes *obl:[case]:gen*. This procedure reduces label counts substantially, e.g., to 59 for Arabic.

### 8.4.4. Assembling the Dependency Graph

As described in Sec. 8.2.4, the outputs $P(y_{i,j})$ provided by the dependency classifier can be regarded as a 3-dimensional tensor, with one dimension corresponding to the tokens as heads, one dimension corresponding to the tokens as dependents, and the third dimension corresponding to the dependency label set. In the following, we describe the steps to create an actual dependency graph from these scores.

**Ensembling.** Instead of continuing directly with the predicted matrices as described above, we train multiple models with different initializations for each language and then ensemble them by averaging their output probabilities during prediction. In other words, we compute the probabilities of labels $P(y_{i,j})$ between two ordered tokens $i$ and $j$ as

$$P(y_{i,j}) = \frac{1}{m} \sum_{k=1}^{m} \text{softmax}(\mathbf{s}_{i,j}^{(k)}) \tag{8.3}$$

where $m$ is the number of models and $\mathbf{s}_{i,j}^{(k)}$ is the unnormalized output vector of the $k$-th model for the token pair $(i, j)$.

For languages for which more than one training treebank is available, we ensemble models trained on different treebanks. For more details on this procedure, see Sec. 8.5.1.

**Ensuring graph structure constraints.** Using the output tensors created via ensembling, we build dependency graphs by applying the same methodology as described in Sec. 8.2.4. To ensure structural validity of the resulting graphs, we perform the following heuristic post-processing steps, which present a simplification of our IWPT 2020 system (i.e., no fallback to a 3rd-party parser):

1. If the graph has more than one root, we remove all but the most confidently predicted *root* dependency.

2. If there are one or more nodes in the graph that are not reachable from the root, we select the most confidently predicted non-∅ edge from a reachable to an unreachable

node and add it to the graph. We repeat this step until every node is reachable from the root.

**Removal of superfluous dependencies.** Like in our 2020 system, if our parser erroneously predicts several relations for a single token that empirically only appear on their own (*fixed*, *flat*, *goeswith*, *punct*, and *cc*), we remove all but the most confidently predicted dependency.

### 8.4.5. Label Lexicalization

As outlined in Sec. 8.4.3, lexical information is stripped from dependency labels during training, using the format *base:[placeholder]*. At prediction time, we re-lexicalize predicted placeholder labels using a novel two-step procedure that presents an extension of our IWPT 2020 system as described in Sec. 8.2.4. First, lexical material is retrieved from the dependency graph using a rule-based heuristic, and then a machine-learning classifier is run on the output to correct potential errors. We refer to this two-step procedure as a "hybrid" approach due to its reliance on both a rule-based and a machine learning-based component.

**Re-lexicalization heuristic.** For the rule-based heuristic, we stick with our approach from the IWPT 2020 Shared Task, which is described in Sec. 8.2.4. As can be seen in Table 8.9, the rule-based heuristic achieves good results in the case of English – the language that it was initially designed for – and for a number of other languages (e.g. Estonian and Latvian), with re-lexicalization accuracies greater than 98 % when evaluating on the gold development data. However, it performs markedly worse for some of the other languages in the Shared Task, such as Arabic, Czech and (especially) Tamil. The main reason for this is that the heuristic can only retrieve word forms directly from the raw sentence, whereas the lexical material in gold dependency labels is lemmatized.

**Rule-based label transducer.** To increase re-lexicalization accuracy, we perform a second step after running the heuristic on the initial parser output, automatically learning a "label transducer" for each language from treebank data. For each language, we train a RandomForest classifier (Breiman, 2001) that takes as input the lexicalized labels predicted by our heuristic as well as a representation of its sentential context. The label transducer then predicts a new label, which may differ from the initial prediction made by the heuristic. In other words, the label transducer functions as an ML-based

| Treebank | Heuristic | Hybrid |
|---|---|---|
| Arabic-PADT | 93.4 | 97.5 |
| Czech-PDT | 90.9 | 99.2 |
| English-EWT | 98.4 | 98.8 |
| Estonian-EDT | 98.8 | 99.8 |
| Latvian-LVTB | 99.4 | 99.7 |
| Polish-PDB | 91.8 | 98.9 |
| Slovak-SNK | 93.0 | 98.0 |
| Tamil-TTB | 16.1 | 66.1 |

Table 8.9.: Re-lexicalization accuracy (%) on gold graphs from a selection of development treebanks. "Hybrid" refers to our full two-step (heuristic+transducer) approach.

error correction mechanism that is able to go beyond merely copying lexical material from the input sentence by exploiting observed patterns in the data (e.g., the replacement of word forms by their lemmas in labels).

The input features for the classifier are (a) a one-hot encoding of the lexicalized label predicted by the heuristic, e.g., *conj:als*; and (b) a binary encoding of all tokens in the graph that are at most 1 dependency edge away from the endpoint of the relation in question. The output space is the set of lexicalized dependency labels as present in the gold training data.

In few cases, the label transducer predicts a different base relation type compared to the one given as input, i.e., it may transform an input of *conj:als* into an output of *nmod:in*. As we observed that such predictions are almost always incorrect, we keep the heuristic's output in these cases.

In preliminary experiments on the gold development data, we find that this hybrid approach leads to moderate to large accuracy increases (see Table 8.9). The improvements are particularly pronounced for languages where lexical material in dependencies often differs from the raw tokens in the graph (e.g., Arabic and Tamil).

## 8.5. IWPT21: Evaluation

This section describes our main submission, as well as a number of additional experiments.

### 8.5.1. Experimental Settings

We use the provided training and development data for training and validation, respectively. During training, we use gold-segmented sentences and gold tokenization.

| | |
|---|---|
| **XLM-R embeddings** | |
| Embeddings dimension | 1024 |
| Token mask probability | 0.15 |
| Layer dropout | 0.1 |
| Hidden dropout | 0.2 |
| Attention dropout | 0.2 |
| Output dropout | 0.5 |
| **Biaffine classifier** | |
| Hidden size | 1024 |
| Dropout | 0.33 |
| **AdamW Optimizer** | |
| Batch size | 32 |
| LR schedule | Noam |
| Warmup steps | 1 epoch |
| Peak learning rate | $4e^{-5}$ |
| $\beta_1$, $\beta_2$ | 0.9, 0.999 |
| Weight decay | 0.0 |

Table 8.10.: Basic hyperparameter values used in training.

For hyperparameter settings, we mostly stick with the values listed in Sec. 8.2. The exceptions are parameters related to the training process itself, where we use a batch size of 32 in conjunction with an inverse square root ("Noam") learning rate schedule (Vaswani et al., 2017) that reaches a peak LR of $4e^{-5}$ after one epoch of training. We found this configuration to yield results comparable to our previous setup, but at noticeably higher training efficiency. Table 8.10 shows the full set of hyperparameters.

The above setup works robustly across languages, with Tamil being the only exception, reaching only ca. 54 ELAS F1 on the development data. For the low resource setting of parsing Tamil, we hence use a batch size of 1, a lower learning rate ($1e^{-5}$), as well as a longer warmup time (5 epochs) and higher early stopping patience (40 epochs).

We train 5 models per language and ensemble these models for our final predictions (see Sec. 8.4.4). For languages with more than one training treebank, we train models on all treebanks provided, with more models trained on larger treebanks. The one exception to this is Polish, where we found ensembling of models trained on both the PDB and LFG treebanks to yield worse results than just training on PDB (likely due to systematic annotation differences). Table 8.11 shows the ensemble composition for all languages with multiple training treebanks.

Each model is trained using a single nVidia Tesla V100 GPU, stopping early when ELAS F1 score on the development set does not improve for 20 epochs, or after at most 24 hours. Training time varies substantially by treebank and correlates with treebank size, with training being fastest for Tamil-TTB (ca. 2 hours on average) and slowest for

| Language | Ensemble composition |
|---|---|
| Czech | 3xPDT, 1xCAC, 1xFictree |
| Dutch | 3xAlpino, 2xLassySmall |
| English | 3xEWT, 2xGUM |
| Estonian | 4xEDT, 1xEWT |
| Polish | 5xPDB |

Table 8.11.: Ensemble compositions for languages with more than one training treebank.

Russian-SynTagRus and Czech-PDT (both run into the 24-hour time limit).

### 8.5.2. Results of Submission

Table 8.12 shows the results (in terms of ELAS F1 score) on the blind test data for our main submission (rightmost column, $\text{ensemble}_{\text{hyb}}$) as well as the 1st- and 2nd-scoring submissions of the Shared Task (TGIF and ShanghaiTech), as well as the median submission for each language. Our system achieves an average ELAS F1 score of 86.97 %, ranking 3rd with a margin of more than 3 points over the median.

The best results achieved by our system are for Bulgarian and Italian, each with ELAS F1 scores of over 93. In contrast, Tamil is the language that we perform by far the worst on, with an ELAS F1 score of around 59. In an extreme low-resource scenario such as parsing Tamil (where the training data consists of only 400 sentences), adaptations to our framework will be necessary.

### 8.5.3. Analysis of Results

To tease out the effects of re-lexicalization and ensembling, we submitted two more experiments on the blind test data after the official deadline.

**Effect of re-lexicalization strategy.** In a first experiment, we did not use our machine learning-based label transducer for re-lexicalization of labels, instead relying only on the rule-based heuristic. The results of this experiment can be found in the column labelled "$\text{ensemble}_{\text{heur}}$" in Table 8.4.

Using only the heuristic for label lexicalization results in a modest, but noticeable accuracy hit across languages, reducing the average ELAS F1 score by roughly 0.2. The languages most affected are Czech (-0.70), Arabic (-0.46), Slovak (-0.35), and Tamil (-0.33), in which differences between lexical material in the sentence and their lemmas included in lexicalized labels are frequent. In contrast, many languages see only small or

| | Other teams | | | RobertNLP | | |
|---|---|---|---|---|---|---|
| **Language** | TGIF | Shanghai | *Median* | single | ensemble$_{\text{heur}}$ | ensemble$_{\text{hyb}}$ |
| Arabic | 81.23 | **82.26** | 76.39 | 81.37 | 81.12 | ***81.58*** |
| Bulgarian | **93.63** | 92.52 | 90.84 | 92.94 | 92.91 | ***93.16*** |
| Czech | **92.24** | 91.78 | 89.08 | 89.99 | 89.51 | ***90.21*** |
| Dutch | **91.78** | 88.64 | 84.14 | 88.02 | 88.21 | ***88.37*** |
| English | **88.19** | 87.27 | 85.70 | 87.29 | ***87.89*** | 87.88 |
| Estonian | **88.38** | 86.66 | 84.02 | 86.10 | 86.52 | ***86.55*** |
| Finnish | **91.75** | 90.81 | 89.02 | 90.77 | 90.97 | **91.01** |
| French | **91.63** | 88.40 | 87.32 | ***88.59*** | 88.51 | 88.51 |
| Italian | **93.31** | 92.88 | 91.81 | 93.00 | 93.16 | ***93.28*** |
| Latvian | **90.23** | 89.17 | 84.57 | 88.68 | 88.80 | ***88.82*** |
| Lithuanian | **86.06** | 80.87 | 78.04 | ***80.98*** | 80.76 | 80.76 |
| Polish | **91.46** | 90.66 | 88.31 | 89.49 | 89.54 | ***89.78*** |
| Russian | **94.01** | 93.59 | 90.90 | 92.55 | 92.33 | ***92.64*** |
| Slovak | **94.96** | 90.25 | 87.04 | 89.60 | 89.29 | ***89.66*** |
| Swedish | **89.90** | 86.62 | 84.91 | 87.72 | 88.02 | ***88.03*** |
| Tamil | **65.58** | 58.94 | 52.27 | 58.24 | 59.00 | ***59.33*** |
| Ukrainian | **92.78** | 88.94 | 86.92 | 88.56 | ***88.86*** | ***88.86*** |
| Average | **89.24** | 87.07 | 83.64 | 86.70 | 86.78 | ***86.97*** |

Table 8.12.: Parsing results (ELAS F1) on blind test data in the IWPT 2021 Shared Task. ensemble$_{\text{hyb}}$ is our main submission, using both the re-lexicalization heuristic and the label transducer.

no performance drops (e.g. Lithuanian, Swedish); for English, performance even increases very slightly when removing the label transducer.

These results indicate that while using our hybrid system is beneficial, good results for most languages can also be achieved when relying solely on our re-lexicalization heuristic. This makes it conceivable that in conjunction with a high-accuracy lemmatizer, a purely rule-based system may perform on par with a hybrid system, and we view this as an interesting avenue for future work.

**Effect of ensembling.** In a second experiment, we did not perform model ensembling for prediction, instead only using a single model for each language. The column labelled "single" in Table 8.4 reports the best results achieved for each language when using only the best single model.

As can be seen, utilizing only a single model per language results in a moderate average performance drop of 0.27 ELAS F1 points. With the exception of French and Lithuanian, all languages benefit from model ensembling, with Tamil (+1.09), English (+0.59), Estonian (+0.45), and Dutch (+0.35) showing the strongest improvements. As the latter three include data from different treebanks in their blind test sets, this indicates

that ensembling may also help parser robustness when mixing models trained on different datasets.

However, although the overall effect of ensembling is notable, our parser nonetheless retains a relatively strong performance even without it, and still would have scored 3rd in the Shared Task if it was only using single models.

## 8.6. Conclusion

In this chapter, we have addressed **RQ5 (How can neural dependency parsers be adapted for Enhanced UD?)** and proposed a conceptually straightforward architecture for parsing raw text into Enhanced Universal Dependency graphs. Our approach is based on STEPS, our graph-based parser introduced in Chapter 7, which we have extended to the Enhanced UD formalism using a simple set of architectural adjustments. With our submissions to the IWPT 2020 and IWPT 2021 Shared Tasks – which scored 1st (English only) and 3rd (all languages), respectively – we have furthermore demonstrated the high efficacy of our method in a controlled, competitive setting.

In summary, our contributions and conclusions are as follows:

- We have demonstrated that a simple architecture, based on state-of-the-art transformer-based language models and unfactorized biaffine classifiers, yields highly competitive performance for parsing raw text into Enhanced UD graphs. This mirrors our conclusions for basic UD parsing.
- Our end-to-end system outperforms pipeline approaches that were common for Enhanced UD parsing prior to the IWPT Shared Tasks.
- We present a novel rule-based label lexicalization heuristic for Enhanced UD dependency labels. While it can be improved slightly further with machine learning methods, it is already good on its own, showing that rule-based methods can still perform very well for certain NLP subproblems.
- Model ensembling can boost parsing accuracy further, but such gains are only modest and come at the expense of drastically increased compute requirements, exemplifying the problems with such a "computationally brute force" approach.

**Open issues.** Our work in this chapter raises two main follow-up questions. First, using STEPS for Enhanced UD parsing shows rather poor performance in a low-resource

setting (Tamil); addressing this weakness, ideally while maintaining the parser's relatively simple core architecture, may be a promising avenue for future work. Second, as our work demonstrates the feasibility of predicting graph structures at high accuracies by using a biaffine classifier on top of a state-of-the-art pre-trained language model, there is a clear relevance w.r.t. **RQ6 (Can we leverage the biaffine dependency parser architecture for other NLP applications?).** In particular, many NLP applications (e.g., Negation Resolution or Relation Extraction) may be framed as graph parsing tasks. It therefore stands to reason that the methods outlined in this chapter could also yield good results for these types of problems, a hypothesis which we evaluate in Chapter 9 and Chapter 10.

# Part IV.

# Biaffine Dependency Parsing in NLP Applications

# 9. Negation Resolution

## 9.1. Introduction

After having investigated dependency parser architectures, in this final part of the dissertation, we now turn our attention to **RQ6 (Can we leverage the biaffine dependency parser architecture for other NLP applications?).** To this end, we focus on two such applications in particular: Negation resolution (in the present chapter) and relation extraction (in Chapter 10).

**Introduction to Negation Resolution.** Negation is a complex semantic phenomenon in natural language that "transforms an expression into another expression whose meaning is in some way opposed to the original" (Morante and Blanco, 2021). It occurs frequently, with the proportion of sentences with negation in English corpora ranging between 9 and 32% (Jiménez-Zafra et al., 2020). NLP applications that may benefit from negation resolution include sentiment analysis (Wiegand et al., 2010; Moore and Barnes, 2021) and information extraction. Negation is also still a challenge in machine translation (Fancellu and Webber, 2015; Bentivogli et al., 2016; Hossain et al., 2020a) and natural language inference (Hossain et al., 2020c; Geiger et al., 2020).

Negation Resolution (Morante and Blanco, 2021) refers to the task of automatically retrieving the elements of a sentence that are affected by the negation introduced by a *cue*. The cue's *scope* is "the part of the meaning that is negated" (Huddleston and Pullum, 2002). The task is difficult due to the multitude of ways in which negation may be expressed. Despite having been a continuously active research area especially since two shared tasks (Farkas et al., 2010; Morante and Blanco, 2012), building robust computational models is far from being a solved task, in part due to a lack of annotation standards (Jiménez-Zafra et al., 2020).

Negation resolution has traditionally been addressed by heavily relying on syntactic parses (e.g., Sanchez Graillet and Poesio, 2007; Sohn et al., 2012; Mehrabi et al., 2015). Recently, end-to-end neural approaches to modeling negation resolution have claimed

superior performance (Fancellu et al., 2016, 2018; Khandelwal and Sawant, 2020; Kurtz et al., 2020). Addressing **RQ6 (Can we leverage the biaffine dependency parser architecture for other NLP applications?),** in this chapter we build on this line of work and evaluate the performance of a number of different neural approaches to negation resolution in a principled fashion, providing a competitive range of baselines for future work to compare with.[1]

Our contributions are as follows:

- Building on work by Kurtz et al. (2020), we propose and evaluate a new method of encoding negation resolution as dependency parsing task. We show that our new *nested* formalism for converting negation annotations to tree structures is easier to learn than the one proposed by Kurtz et al. (2020), and that transfer learning from a syntax-trained language model is effective.

- We conduct a reproducibility study on negation resolution, reporting performance scores of a variety of relevant baseline systems in a uniform experimental setup.

- We show that a modern transformer-based re-implementation of the tagging-based system by Fancellu et al. (2016) achieves the best negation resolution performance under most circumstances.

## 9.2. Related Work

We here introduce the linguistic terminology and concepts used in this chapter, and briefly survey related work in computational linguistics.

### 9.2.1. Linguistic Background

A negation **cue** signals to the listener or reader that the inverse of something is referred to. In the ConanDoyle-neg (CD-neg) annotation scheme (Morante et al., 2011), a negation cue may be a single word such as "not," a multi-word expression such as "no more," or a negation affix such as "im" (e.g., in "imprecise"). The **scope** is the part of the sentence that is "affected" by the negation signaled by the cue (Huddleston and Pullum, 2002).

[1]The text of this chapter is based on Sineva et al. (2021). Elizaveta Sineva implemented negation resolution evaluation scripts and biaffine parsing-based systems based on the STEPS parser. For more details on the evaluation aspect of negation resolution, the reader is referred to the aforementioned paper.

Logically, negation is an operator taking a proposition, which typically corresponds to a (sub-)clause (e.g., $like(m, p)$ for "Mary likes pizza"), converting the corresponding assertion to an assertion stating that something is not the case ($\neg like(m, p)$ for "Mary does **not** like pizza"). However, in natural language, depending on the embedding context as well as pragmatic presuppositions, it is not always the case that negation operators convert something to its logical complement (Horn, 2010; Blanco and Moldovan, 2011b). For example, "she is not unhappy" does not mean "she is happy."

In CD-neg, the aim of the annotation is to make explicit which event (process, or state) is affected by the change of polarity (Morante et al., 2011; Morante and Daelemans, 2012). The word referring to the event is marked as *Event*, but only if the event is factual. Thus, in cases such as "He may not know the answer," no event is annotated. To avoid terminological confusion, in this chapter, we ignore *Event* annotations and call "know" in the example above the *main predicate* of the negation.

The scope is annotated as the longest relevant part of the sentence, i.e., as the main predicate referring to the negated event and all its arguments and complements. In contrast to BioScope (Szarvas et al., 2008), CD-neg includes the subject, but not the cue in the scope. In constituent negation, the negation marker is attached to the object as in "Mary came to the lecture with **no** books." Still, the negation scopes over the entire sentence and is marked accordingly in CD-neg to achieve representational equality with the sentence "Mary did **not** come to the lecture with books." A constituent-negated subject receives the same treatment.

One element of the scope is singled out as the negation's **focus**, i.e., the part that is *intended* to be interpreted as false (Huddleston and Pullum, 2002). Detecting the focus usually requires leveraging phonetic cues as in "Your kids don't *hate* school" vs. "*Your* kids don't hate school" (Blanco and Moldovan, 2011b). Correct identification of a negation's focus is key to natural language understanding. However, to the best of our knowledge, no corpora annotating both negation scopes and focus exist to date.

### 9.2.2. Automatic Negation Resolution

Computational work on negation resolution is generally based on small- to medium-scale corpora, which in addition are often not very compatible due to differences in the employed annotation schemes and underlying tokenization. Jiménez-Zafra et al. (2020) provide a comprehensive survey of datasets annotated for negation.

As the problem of negation resolution is closely tied to syntax, there are many works leveraging syntactic information, using rules over syntactic structures to resolve negation

or speculation scopes (e.g., Velldal et al., 2012; Packard et al., 2014; McKenna and Steedman, 2020) or training systems with explicit or learned syntactic features (Read et al., 2012; Lapponi et al., 2012; Enger et al., 2017; Ren et al., 2018; Jiménez-Zafra et al., 2020).

Li et al. (2010) frame scope resolution as a shallow semantic parsing task backed up by syntactic parses. In the neural age, Kurtz et al. (2020) frame negation resolution as a dependency parsing task. Several works using neural networks (e.g., Fancellu et al., 2016, 2018; Lazib et al., 2020) train BiLSTMs, or syntactically structured BiLSTMs or GCNs. Qian et al. (2016) propose a CNN-based architecture combined with some path/position information. Recently, a range of papers has explored BERT-based models for negation resolution (Khandelwal and Sawant, 2020; Khandelwal and Britto, 2020; Britto and Khandelwal, 2020; Shaitarova and Rinaldi, 2021). Further related work includes datasets annotated for focus (Blanco and Moldovan, 2011a; Altuna et al., 2017), and the computational modeling thereof (e.g., Hossain et al., 2020b). In addition, there is a growing body of work addressing negation within the context of neural language models and commonsense reasoning using them (e.g., Hossain et al., 2020c; Geiger et al., 2020; Hosseini et al., 2021; Jiang et al., 2021).

## 9.3. Modeling

In this section, we describe the negation resolution models that we will compare in our experiments in Sec. 9.4. Due to the large number of negation resolution systems, often with no published code, it is infeasible to provide unified evaluation scores for all prior work. Instead, to provide competitive baselines for future work to compare with, we chose a wide range of neural architectures inspired by previous work and re-implement them. For comparability reasons, we base them all on the same robust transformer-based language model.

**Token representation.** Our token embedding backbone is the transformer-based XLM-R-large language model (Conneau et al., 2020), as well as the corresponding word-piece tokenizer. To obtain contextualized word embeddings, we take a weighted sum of the internal states corresponding to the first word piece for each token. The coefficients of this weighted sum are learned during training, employing layer dropout (see Kondratyuk and Straka, 2019, and Sec. 7.3.1). Transformer weights are fine-tuned during training. To determine the effect of injecting implicit syntactic knowledge into the system, in

addition to using the default pre-trained XLM-R model, we also run experiments on an XLM-R-**synt** model that was previously fine-tuned on the task of Universal Dependencies parsing on the EWT treebank (Silveira et al., 2014).

### 9.3.1. Sequence-Tagging Based Approaches

Tagging pipelines first identify negation cues, and then for each cue, identify its scope using a second tagger. The **NegBERT** system (Khandelwal and Sawant, 2020; Britto and Khandelwal, 2020), which we run using XLM-R, modifies the input for the second step, adding artificial tokens to indicate cues. In addition, we implement a **BiLSTM-Tagger** following the architecture proposed by Fancellu et al. (2016), but using XLM-R as the underlying language model.[2] Cues are predicted using a single linear layer with softmax on top of XLM-R. Scopes are predicted by feeding, for each negation instance, the XLM-R embeddings of the sentences concatenated with a *cue/notcue* embedding to a single-layer BiLSTM and once again using a token-wise linear+softmax layer for classification.

### 9.3.2. Dependency-Parsing Based Approaches

This class of models frame negation resolution as a dependency parsing (**DP**) task, as proposed by Kurtz et al. (2020), predicting cues and scopes in a single step by encoding negation instance annotations as dependency trees. The systems we present here differ w.r.t. this encoding (see Figure 9.1). In the **direct mapping** (Kurtz et al., 2020), cue tokens are modeled as dependents of the artificial root token, and scope and event tokens are attached via a dependency link to all cues they belong to. In addition, we propose a **nested mapping** in which in the cases of embedded scopes, there is only one link from the outer scope's cue to the inner scope's cue, and all other scope tokens are only linked to their corresponding nearest cue. We build these models using our graph-based dependency parser STEPS (Grünewald et al., 2021a, see also Chapter 7).

## 9.4. Experiments

We here report our results for end-to-end negation resolution including cue detection and scope resolution. In all of our evaluations, we ignore punctuation tokens. In addition to the models explained in Sec. 9.3, we report results for a punctuation baseline **(Punct-BL)**

[2]Fancellu et al. (2016) use task-specific learned or word2vec embeddings (Mikolov et al., 2013).

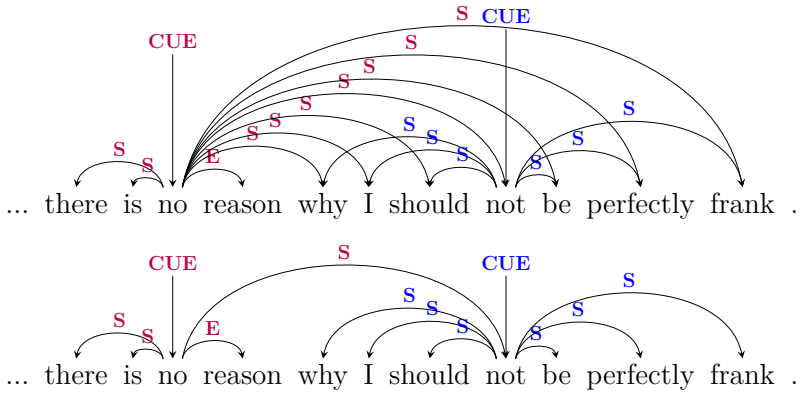



Figure 9.1.: **Direct** (upper) vs. **nested** (lower) encoding.

that uses gold cues and tags everything between the cue and the next punctuation marker as the scope.

### 9.4.1. Datasets

We conduct our experiments on three corpora from a variety of domains. In our experiments, we focus on the CD-neg (**CD**) dataset, which has been shown to be most challenging among the English negation corpora (Fancellu et al., 2017). The corpus comprises seven literary texts corresponding to 5,520 sentences with 1,432 negation instances. CD annotates "neither...nor" as a multiword cue with a single scope. However, from a semantic point of view, we interpret "Neither Mary nor Sam like pizza" as $\neg like(m, p) \wedge \neg like(s, p)$, which actually suggests annotating two separate instances with overlapping scopes.[3] We noticed that the majority of cases in which systems got multiword cue detection wrong were like this, detecting only part of the cue but resolving the scope correctly. Rather than punishing systems for this, we decide to re-annotate the dataset accordingly, fixing a total of 10 cases.

In addition, we conduct experiments on the **BioScope** corpus (Szarvas et al., 2008) using the abstracts subset (11,871 sentences), as well as the **SFU Review** corpus (Taboada et al., 2006; Konstantinova et al., 2012), which comprises 400 reviews in eight different domains. The BioScope and SFU datasets are also annotated for speculation; our experiments only use the negation annotations. More details on all corpora are given in Table 9.1.

[3]The original annotation may be seen as corresponding to an (equivalent) formalization of $\neg\left(like(m, p) \vee like(s, p)\right)$.

| Dataset | *ConanDoyle-neg* | *BioScope Abstracts* | *SFU Review* |
|---|---|---|---|
| **Source** | Morante and Daelemans (2012) | Szarvas et al. (2008) | Konstantinova et al. (2012) |
| **Domain** | fiction writing | biomedical | review |
| **Sentence #** | 5,520 | 11,871 | 17,263 |
| **Negation sentence #** | 1,227 | 1,597 | 3,117 |
| **Negation instance #** | 1,421 (original) / 1,432 (ours) | 1,719 | 3,518 |
| **Annotated for speculation** | no | yes | yes |
| **Cue is a part of the scope** | no | yes | no |
| **Includes discontinuous scopes** | yes | no | yes |
| **Includes events** | yes | no | no |
| **Annotates negation affixes** | yes | rarely, with the whole word as a cue | yes, but with the whole word as a cue |
| **Tokenized** | yes | no | yes |
| **File format** | CoNLL | XML | XML |

Table 9.1.: Overview of datasets annotated for negation used in our study.

### 9.4.2. Evaluation Metrics

A wide variety of evaluation metrics has been used for negation resolution, making it difficult meaningfully compare existing approaches and benchmark new models. This problem was addressed by Sineva et al. (2021), who provide a review of existing metrics and propose new, linguistically motivated metrics. For an in-depth discussion and precise mathematical formalization, the reader is referred to their paper; we here provide only a very brief description of the relevant metrics.

The metrics reported in this section are:

- **Cues-B:** *SEM 2012 metric for (exact) matching of **negation cues;**
- **SCM:** *SEM 2012 metric for (exact) matching of **negation scopes** (including exact cue match);
- **ST:** *SEM 2012 metric (precision, recall, F1) for **tokens** belonging to the scope of a negation instance in the gold data vs. system output;
- **$\text{NIS}_{\text{tok}}$:** precision, recall, and F1 scores that are the expectations of the **instance-level token-match scores when weighting each instance equally;**
- **$\text{NIS}_{\text{ex}}$:** same as $\text{NIS}_{\text{tok}}$ but requiring exact scope match.[4]

[4]Similar to SCM but with a slightly different method of calculation. For further details, see Sineva et al. (2021).

| Dataset | | Train | Dev | Test | Total |
|---|---|---|---|---|---|
| *ConanDoyle-neg* | sentence # | 3,644 | 787 | 1,089 | 5,520 |
| | sentence % | 66% | 14.3% | 19.7% | 100 % |
| | negation instance # | 984 | 173 | 264 | 1,421 |
| | negation sentence # | 848 | 144 | 235 | 1,227 |
| | negation sentence % | 23.3% | 18.3% | 21.6% | 22.2% |
| (reannotated) | negation instance # | 987 | 176 | 269 | 1,432 |
| *BioScope Abstracts* | sentence # | 9,500 | 1,185 | 1,186 | 11,871 |
| | sentence % | 80% | 10% | 10% | 100% |
| | negation instance # | 1,396 | 156 | 167 | 1,719 |
| | negation sentence # | 1,297 | 148 | 152 | 1,597 |
| | negation sentence % | 13.7% | 12.5% | 12.8% | 13.5% |
| *SFU Review* | sentence # | 13,614 | 1,817 | 1,800 | 17,231 |
| | sentence % | 79% | 11% | 10% | 100% |
| | negation instance # | 2,835 | 365 | 309 | 3,509 |
| | negation sentence # | 2,503 | 328 | 276 | 3,107 |
| | negation sentence % | 18.4% | 18.1% | 15.3% | 18% |

Table 9.2.: Dataset splits used in our experiments.

| Model | XLM-R-large |
|---|---|
| Token mask probability | 0.15 |
| Layer dropout | 0.1 |
| Hidden dropout | 0.2 |
| Attention dropout | 0.2 |
| Output dropout | 0.5 |

Table 9.3.: Hyperparameter values for the XLM-R language model.

### 9.4.3. Experimental Setup

We use the official training-dev-test split for CD-neg. For BioScope and SFU, we create our own 80-10-10 splits for these datasets. For more information on the splits, see Table 9.2. To tokenize BioScope, we use NLTK (Loper and Bird, 2002) with custom rules for punctuation and URLs.

Our models are implemented using PyTorch (Paszke et al., 2019) and the Huggingface Transformers library (Wolf et al., 2020). Training is performed on a single nVidia Tesla V100 GPU. We use a unified set of hyperparameters for the underlying XLM-R language model, which are given in Table 9.3.

**BiLSTM-Tagger.** Our BiLSTM tagger consists of two subsystems, one for cue tagging and one for scope tagging. For the cue tagging subsystem, we simply use a linear layer with softmax on top of the XLM-R model. The model is then trained using the

| | |
|---|---|
| Optimizer | AdamW |
| Weight decay | 0 |
| Batch size | 8 |
| Base learning rate | $2e^{-5}$ |
| LR schedule | Noam |
| LR warmup | 1 epoch |

(a)

| **BiLSTM** | |
|---|---|
| BiLSTM layers | 1 |
| BiLSTM hidden size | $2 \times 200$ |
| BiLSTM dropout | 0.0 |
| *cue*/*notcue* embedding dim. | 128 |
| **Optimization** | |
| Optimizer | AdamW |
| Weight decay | 0 |
| Batch size | 4 |
| Base learning rate (BiLSTM) | $2e^{-4}$ |
| Base learning rate (XLM-R) | $2e^{-5}$ |
| LR schedule | Noam |
| LR warmup | 1 epoch |

(b)

Table 9.4.: Hyperparameters for our BiLSTM tagger. (a) Cue tagging; (b) scope tagging.

hyperparameters shown in Table 9.4a.

For the scope tagging subsystem, we add a 1-layer BiLSTM on top of the XLM-R model and then use a linear layer with softmax to classify tokens as part of negation scopes. In this system, we use different learning rates for the XLM-R model vs. the BiLSTM and the classifier. We found that using low learning rates for the entire system causes it to underfit. Furthermore, because the scope tagger is only trained on a smaller number of instances (i.e., only those that actually contain negation instances), we found it beneficial to reduce the batch size compared to the cue tagger. Our final hyperparameters can be found in Table 9.4b.

**Dependency Parser.** Our parser implementation is based on STEPS (Grünewald et al., 2021a, see Chapter 7). Table 9.5 gives the hyperparameters used for this system. Like the cue tagger, we use only a single learning rate for the entire system (XLM-R model as well as classifier).

### 9.4.4. Results

Table 9.6 shows the results for our set of competitive neural systems on CD-neg. For cue detection, all systems perform similarly well, with the parsing-based approach using the nested representation and a syntactically fine-tuned XLM-R (DP-nested-synt) having a slight advantage. We ran the NegBERT system for end-to-end negation resolution

| **Biaffine classifier** | |
|---|---|
| Arc and label scorer dimension | 1024 |
| Dropout | 0.33 |
| **Optimization** | |
| Optimizer | AdamW |
| Weight decay | 0 |
| Batch size | 32 |
| Base learning rate | $4e^{-5}$ |
| LR schedule | Noam |
| LR warmup | 1 epoch |

Table 9.5.: Hyperparameter values for our dependency parsing-based system.

| **System** | **Cues-B** $F_1$ | **SCM** $F_1$ | **NIS$_{ex}$** $F_1$ | **ST** P | **ST** R | **ST** $F_1$ | **NIS$_{tok}$** P | **NIS$_{tok}$** R | **NIS$_{tok}$** $F_1$ |
|---|---|---|---|---|---|---|---|---|---|
| *Punct-BL* | *100.0* | *17.0* | *9.9* | *89.8* | *62.0* | *73.3* | *93.3* | *58.8* | *72.1* |
| NegBERT | 91.1 | 78.8 | **70.1** | 86.1 | **87.8** | 86.9 | 83.9 | **88.7** | 86.3 |
| BiLSTM-Tagger | 92.3 | **79.6** | 70.0 | **90.8** | 85.4 | **88.0** | **90.3** | 86.3 | **88.2** |
| DP-direct | 92.8 | 73.2 | 61.4 | 85.4 | 87.4 | 86.4 | 84.6 | 87.7 | 86.1 |
| DP-nested | 93.3 | 72.8 | 60.6 | 86.6 | 83.0 | 84.8 | 85.6 | 85.4 | 85.5 |
| DP-direct-synt | 92.8 | 79.1 | 69.2 | 87.4 | 86.5 | 86.9 | 86.2 | 88.3 | 87.2 |
| DP-nested-synt | **93.4** | 79.2 | 69.2 | 88.0 | 84.6 | 86.2 | 87.2 | 86.6 | 86.9 |

Table 9.6.: **Comparison of Evaluation Scores** when training and testing on **CD-neg**. Underlined: Comparison NegBERT vs. DP-direct-synt.

using the original code, but our evaluation scripts.[5] In this unified evaluation setup, in the SCM metrics, we can only see that the DP models based on standard XLM-R underperform; all other systems perform roughly similarly. The less strict ST and $NIS_{tok}$ metrics reveal different precision-recall trade-offs for the direct vs. the nested encoding. The BiLSTM-Tagger turns out to be the most accurate model for scope resolution, reflected similarly in the ST and the $NIS_{tok}$ scores. Our architecture similar to the one of Fancellu et al. (2018) seems to outperform NegBERT, which adds artificial tokens to indicate cues.[6]

As expected to some extent, the $NIS_{tok}$ and ST scores are similar in general. Both scores identify the BiLSTM-Tagger as the most precise system and NegBERT as having

[5]Khandelwal and Sawant (2020) report results for scope resolution that appear to be based on gold cues.

[6]We ran the scope tagger (trained on gold cues) on the output of the best cue tagger run as chosen by dev set performance. Note that while this reflects a real-life development setup, the DP models predict cues and scopes in one run. Likewise, NegBERT scores were produced by running the system off-the-shelf without optimizing the cue tagger separately.

| | System | Cues-B $F_1$ | $NIS_{ex}$ $F_1$ | ST $F_1$ | $NIS_{tok}$ P | $NIS_{tok}$ R | $NIS_{tok}$ $F_1$ |
|---|---|---|---|---|---|---|---|
| BioScope | *Punct-BL* | *100.0* | *43.2* | *65.7* | *77.0* | *69.7* | *73.5* |
| | NegBERT | 90.5 | 79.5 | 84.5 | 83.3 | 91.5 | 87.2 |
| | BiLSTM-Tagger | 94.3 | **83.0** | 88.6 | 91.7 | 91.7 | 91.7 |
| | DP-nested-synt | **95.4** | 82.7 | **90.4** | **92.7** | **92.1** | **92.4** |
| SFU | *Punct-BL* | *100.0* | *45.9* | *66.5* | *75.5* | *87.4* | *81.0* |
| | NegBERT | 81.9 | 69.5 | 71.0 | 67.9 | 86.3 | 75.9 |
| | BiLSTM-Tagger | **86.7** | **73.2** | **77.9** | **74.9** | **89.8** | **81.7** |
| | DP-nested-synt | 86.5 | 71.1 | 77.1 | 73.7 | 88.9 | 80.5 |

Table 9.7.: **In-Domain Comparison of Systems** on the BioScope and SFU datasets.

the highest recall, with the Tagger achieving the best $F_1$. However, while the bold-facing in Table 9.6 indicates similar patterns for the top systems, the ranking of the other systems differs when comparing ST and $NIS_{tok}$ scores. For example, ST assigns the same summary statistic ($F_1$) to NegBERT and DP-direct-synt, while in terms of $NIS_{tok}$, the $F_1$ of DP-direct-synt is more than 1 point higher. Comparing ST and $NIS_{tok}$ scores, we can see that $NIS_{tok}$ generally assigns higher recall, but slightly lower precision to systems such as NegBERT or DP; the BiLSTM-Tagger's precision drops less. Hence, the ST scores for models such as NegBERT or DP slightly under-estimate recall because the systems failed more often on longer instances; and in turn they slightly over-estimate precision, e.g., because wrongly predicted instances often have short scopes. We here argue that while monitoring the system on several metrics is generally a good idea, $NIS_{tok}$ constitutes a more realistic gradual end-to-end evaluation metric for negation resolution systems, and should be adopted as the main summary statistic by subsequent works or shared tasks.

Table 9.7 compares the various system architectures on the BioScope and SFU data when trained and tested in-domain. As the difference between the DP models was small, we report only DP-nested-synt. On both datasets, the DP-nested-synt models outperform NegBERT. On BioScope, the parsing-based approach clearly outperforms NegBERT and the BiLSTM-Tagger; on SFU, the BiLSTM-Tagger performs best.

Finally, in Table 9.8, we also give results for cross-domain performance. Overall, the BiLSTM-Tagger seems most robust. However, the NegBERT system performs close to or better than the BiLSTM-Tagger when moving from another to the ConanDoyle-neg dataset, and the DP-nested-synt model has an advantage when moving from SFU to BioScope. In sum, particularly cross-domain negation resolution is still far from being

| System | Train | Test | Cues-B $F_1$ | $NIS_{ex}$ $F_1$ | ST $F_1$ | $NIS_{tok}$ $F_1$ |
|---|---|---|---|---|---|---|
| NegBERT | Bio | CD | **65.2** | 8.9 | **55.2** | **51.1** |
| BiLSTM-Tagger | | | 61.6 | **10.1** | 46.0 | 46.8 |
| DP-nested-synt | | | 64.8 | 9.2 | 51.8 | 47.9 |
| NegBERT | SFU | CD | **69.3** | 9.7 | 56.7 | 53.6 |
| BiLSTM-Tagger | | | 68.8 | **10.1** | **58.0** | **53.8** |
| DP-nested-synt | | | 68.1 | 9.6 | 55.2 | 51.6 |
| NegBERT | CD | Bio | 60.6 | **19.0** | 49.3 | 46.2 |
| BiLSTM-Tagger | | | 64.3 | 17.7 | **56.7** | **54.3** |
| DP-nested-synt | | | **65.3** | 18.7 | 54.7 | 50.9 |
| NegBERT | SFU | Bio | 78.7 | 55.5 | 63.2 | 71.0 |
| BiLSTM-Tagger | | | 81.0 | 53.4 | 65.9 | 72.8 |
| DP-nested-synt | | | **82.0** | **57.2** | **67.2** | **74.0** |
| NegBERT | CD | SFU | 51.3 | 10.0 | 37.2 | 37.5 |
| BiLSTM-Tagger | | | **50.2** | **9.0** | **37.5** | **37.8** |
| DP-nested-synt | | | 50.3 | 9.0 | 36.2 | 36.5 |
| NegBERT | Bio | SFU | 63.4 | 48.8 | 53.7 | 58.6 |
| BiLSTM-Tagger | | | **64.5** | **53.1** | **57.0** | **61.4** |
| DP-nested-synt | | | 56.9 | 40.1 | 47.5 | 51.5 |

Table 9.8.: **Cross-Domain Comparison of Systems** between CD-neg, BioScope, and SFU.

solved. We hope that our linguistically motivated evaluation framework can aid the development of more robust negation resolution systems.

## 9.5. Conclusion and Outlook

In this chapter, we have addressed **RQ6 (Can we leverage the biaffine dependency parser architecture for other NLP applications?)** and in this context presented a comparison of different end-to-end neural approaches to negation resolution. We have also proposed and evaluated a novel "nested" method of encoding negation resolution as dependency parsing task, which we show to be easier to learn than the one previously introduced by Kurtz et al. (2020). Furthermore, we have shown that transfer learning from a syntax-trained language model is effective.

In sum, our findings confirm Kurtz et al.'s (2020) finding that biaffine dependency parsing is a valid approach for end-to-end negation resolution. Interestingly, however, we find that a BiLSTM-based tagging approach yields results that are on par or better under most circumstances. In this way, our results echo our conclusions from syntactic

dependency parsing: sophisticated task-specific architectures lose their importance in the presence of a powerful pre-trained language model.

**Future work.** Our experimental study may serve as a reference for future investigations on neural negation resolution. More specifically, one next step may be to evaluate the model suite presented in this chapter on further datasets in languages other than English (e.g., Zou et al., 2015; Liu et al., 2018; Jiménez-Zafra et al., 2018).

# 10. Relation Extraction from Materials Science Texts

## 10.1. Introduction

Designing meaningful experiments in empirical sciences requires maintaining a detailed overview of the huge amounts of literature published every year. Applying natural language processing (NLP) in this context has risen to be an active research area (Chandrasekaran et al., 2020; Beltagy et al., 2021; Cohan et al., 2022). Besides the biomedical field, which has been studied extensively in the past decades (e.g., Collier et al., 2004; Cohen et al., 2012; Demner-Fushman et al., 2022), the less-studied materials science domain has recently received more attention (Mysore et al., 2019; Friedrich et al., 2020; O'Gorman et al., 2021).

Materials science research aims to design and discover new materials. An important part of the papers is hence often dedicated to describing the *synthesis procedures*, in lay terms the "recipe" for creating a material. Extracting such information from papers has been covered by Mysore et al. (2019) and O'Gorman et al. (2021). Much materials science research develops materials in the context of creating a particular device, e.g., batteries or photovoltaic panels. The performance of the device is tested in various conditions and it is of importance to quickly search the literature in order to identify promising materials and set-ups. Friedrich et al. (2020) address such a problem for the sub-domain of solid oxide fuel cells.

In this chapter, we continue to address **RQ6 (Can we leverage the biaffine dependency parser architecture for other NLP applications?)** in the context of information extraction from materials science publications. Specifically, we propose to tackle relation extraction from materials science texts by using a biaffine dependency parser that predicts relations between a given set of named entities (NEs) in a sentence. We evaluate our method on **MuLMS** (the Multi-Layer Materials Science corpus), a recently introduced dataset of materials science publications annotated by a domain

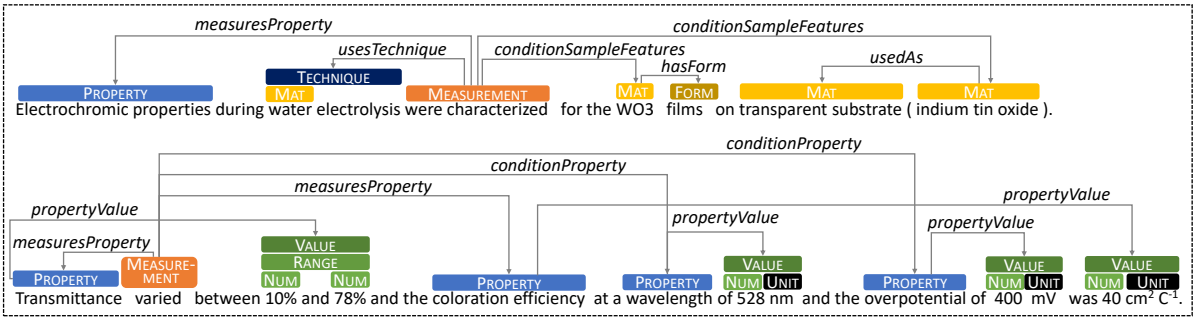



Figure 10.1.: Named entity and relation annotations in the **Multi-Layer Materials Science Corpus** corpus.

expert with NE mentions and relation structures corresponding to broad notion of materials science measurements as shown in Figure 10.1.[1]

Our contributions are as follows:

- We propose and describe using biaffine classifiers for relation extraction from materials science texts, extracting *all* relations between NEs in a sentence in a single processing step.
- We evaluate our method on the recently introduced MuLMS corpus, showing that it outperforms a majority baseline by a substantial margin.
- We perform an extensive set of multi-task learning experiments with related corpora, showing that MuLMS is a useful auxiliary task for two other materials science NLP datasets.

## 10.2. Related Work

Friedrich et al. (2020) treat their slot filling task in the SOFC sub-domain as a sequence tagging task, assuming that each sentence represents at most one experiment. To predict a possible relation between two entities, Swarup et al. (2020) retrieve a set of sentences similar to the input sentence, and learn to copy relations from these sentences. Mysore et al. (2017) experiment with unsupervised methods for extracting action graphs for synthesis procedures.

Sarrouti et al. (2022) compare various pre-trained transformer models for relation extraction in the biomedical domain. Recent works have used graph-neural networks

[1]The text of this chapter is based on sections from Schrader et al. (2023), a collaborative work to which the author of this dissertation contributed the experiments on biaffine parsing-based relation extraction.

| | |
|---|---|
| #Documents | 50 |
| #Documents train / dev / test | 36 / 7 / 7 |
| #Sentences | 10186 |
| #Sentences/Document | 203.7±73.2 |
| #Tokens/Sentence | 28.7±17.9 |

Table 10.1.: Basic **corpus statistics** for MuLMS.

to extract biomolecular relations (Huang et al., 2020). Ramponi et al. (2020) use a convolutional neural network with token, position, type, part-of-speech, and dependency embeddings. Semantic parsing of frame structures (Fillmore and Baker, 2001) has been addressed using graph-convolutional networks (Marcheggiani and Titov, 2020), BiLSTMs (He et al., 2018), and recently by generating structured output using encoder-decoder models (Hsu et al., 2022; Lu et al., 2021). Tackling semantic dependency parsing with a biaffine classifier architecture was first proposed by Dozat and Manning (2018).

## 10.3. MuLMS Corpus

In this section, we describe the MuLMS corpus forming the basis of our experiments.

### 10.3.1. Source of Texts

MuLMS comprises 50 scientific articles licensed under CC-BY from seven popular subareas of materials science: electrolysis, graphene, polymer electrolyte fuel cell (PEMFC), solid oxide fuel cell (SOFC), polymers, semiconductors, and steel. The four SOFC papers were selected from the SOFC-Exp corpus (Friedrich et al., 2020). 11 papers were selected from the OA-STM corpus[2] and classified into the above subject areas by a domain expert. The majority of the papers were retrieved from PubMed[3] and DOAJ[4] using queries prepared by a domain expert. As shown in Table 10.1, documents are rather long with a tendency towards long sentences (but with high variation due to, i.a., short headings).

### 10.3.2. Annotation Scheme

MuLMS's annotation scheme comprises multiple layers, including named entities and relation structures representing measurements.

[2]https://github.com/elsevierlabs/OA-STM-Corpus
[3]https://pubmed.ncbi.nlm.nih.gov
[4]https://doaj.org

### Named Entities

MuLMS contains annotations for the following materials-science specific named entity mentions and types.

**MAT:** mentions of materials described by their chemical formula ($WO_3$) or its chemical name (*indium tin oxide*).

**FORM:** mentions of the form or morphology of the material, e.g., *thin film*, *gas*, *liquid*, *cubic*.

**INSTRUMENT:** mentions of devices used to perform a materials-science-related measurement, e.g., *Olympus BX52 microscope*.

**DEVICE:** mentions of devices (target products) whose construction or improvement is the aim of the research (e.g., *photodetector*, *transistor*, *supercapacitor*). DEVICE is not used for instrumentation devices that are only used as a tool.

**NUM:** mentions of numbers such as *0.46*.

**UNIT:** mentions of units such as *nm* or *V*.

**RANGE:** mentions of numeric expressions indicating ranges, e.g., *0.46±.11*

**VALUE:** nested type capturing expressions of values usually composed of a NUM, RANGE and a UNIT, e.g., *~5 x 3mm2*

**CITE:** citations, e.g., *Setman et al.* or *[13]*.

**PROPERTY:** expressions referring to properties of materials or conditions in experiments, e.g., *stress rate* or *electron conductivity*.

**TECHNIQUE:** mentions of experimental techniques used in the characterization steps, e.g., *Scanning electron microscopy (SEM)*.

**SAMPLE:** mention of the material (or a component made of materials) that is studied in a measurement, either referred to by a particular name or its composition, its batch name (*Aq-825*) or by an expression referring either to the whole component (*MEA-Pt/C*) or to parts of the material's structure such as *ionomer patches*. In simulation papers, the SAMPLE may also be the computational model under study (*RBF-ANN*).

### Relations and Measurement Frame

We here describe MuLMS's set of relations. Measurement annotations are treated in a frame-like (Fillmore and Baker, 2001) fashion, using the span type MEASUREMENT to mark the triggers (e.g., *was measured*, *is plotted*) that introduce the *Measurement* frame to the discourse. About 88% of the triggers are verbs. The remaining 12% occur

in figure captions without verb phrases and are annotated either on nouns (*Comparison*) or, in absence of more suitable phrases, on figure labels such as *Figure 17*. The trigger annotations of these sentences serve as the root of the tree/graph annotations as illustrated in Figure 10.1.

There are also cases in which the *Measurement* frame is evoked, but there are no technical details or results that we can extract about the measurement. We mark the triggers of these sentences with the tag QUAL_MEAS (qualitative mention of a measurement). An example of such a sentence is "*We compare a critical volume to be detached from the different nanostructures.*"

**Measurement-related Relations.** There are several slots that can be filled in the context of measurements. They are annotated as relations that start at a MEASUREMENT tag and that end at the annotations of the corresponding slot fillers within the sentence. Consider the following sentence: "*To characterize the ORR activity of the catalyst, linear scan voltammetry (LSV) was tested from 0 to 1.2 V on an RDE with a scan rate of 50 mV/s in O2-saturated HClO4 solution.*"

***measuresProperty*:** indicates the PROPERTY (e.g., *ORR activity*) that is measured.

***conditionSampleFeatures*:** indicates the SAMPLE or MATERIAL whose property is measured. In the above example, the sample is the *catalyst*.

***usesTechnique*:** relates to the TECHNIQUE (e.g., *linear scan voltammetry*) used in a measurement.

***conditionInstrument*:** refers to the INSTRUMENT used to make a measurement, e.g., *RDE/rotating disk electrode*.

***conditionProperty*:** a property that is a condition in the experiment, e.g., *scan rate* (which in turn has the *propertyValue* of *50 mv/s*).

***propertyValue*:** connects the mention of a PROPERTY and that of its corresponding VALUE. This relation may also occur if a mention of a PROPERTY occurs independently of a measurement.

***conditionEnvironment*:** identifies the MATERIALS (e.g., *O2* and *HClO4*) and VALUES (e.g., an operating temperature of *30°C*) that provide the environment of the measurement.

***takenFrom*:** connects the MEASUREMENT with the bibliographic reference CITE from which the setup has been inspired or taken over.

In most cases, a *conditionProperty* or a *measuresProperty* connects the MEASUREMENT annotation to a PROPERTY node, at which a *propertyValue* relation starts that ends at

| Relation | Example |
|---|---|
| *hasForm* | silicon$_{\text{MAT}}$–*hasForm*→ hexagonal$_{\text{FORM}}$ |
| *usedIn* | Sic$_{\text{MAT}}$ –*usedIn*→ MOSFET$_{\text{DEVICE}}$ |
| *usedAs* | PtNi3M$_{\text{MAT}}$ –*usedAs*→ catalysts |
| *dopedBy* | chlorinated$_{\text{Material}}$–*dopedBy*→SiC$_{\text{MAT}}$ |

Table 10.2.: Measurement-independent **relations** annotated in MuLMS.

| Label | Count | Label | Count |
|---|---|---|---|
| MAT | 15596 | CITE | 1709 |
| NUM | 6081 | SAMPLE | 1461 |
| VALUE | 4852 | TECHNIQUE | 1036 |
| UNIT | 4330 | DEV | 808 |
| PROPERTY | 3925 | RANGE | 736 |
| FORM | 3568 | INSTRUMENT | 378 |
| MEASUREMENT | 2171 | | |

Table 10.3.: Corpus counts for **named entity** annotations.

the respective VALUE. However, in some cases, the condition or measured property is not mentioned explicitly. In this case, we link the VALUE directly to the MEASUREMENT node via a *conditionPropertyValue* or a *measuresPropertyValue* link. For consistency reasons, we also add these links in cases that mention the property explicitly, turning the trees into graph structures. Out of the added *conditionPropertyValue* links, 967 are for such explicit cases, while the other 206 describes implicit cases. In the case of *measuresPropertyValue*, 722 links are for explicit cases and 36 for implicit cases.

**Further Relations.** In the following, we explain relations that can appear independently of measurements. Examples are shown in Table 10.2.

***hasForm*:** connects mention of MATERIAL and the corresponding FORM annotation.

***usedIn*:** connects MATERIAL and the DEVICE it is used in. In the example in Table 10.2, *MOSFET* stands for *Metal Oxide Semiconductor Field-Effect Transistors*.

***usedAs*:** links a specific MATERIAL mention with a more generic one such as *catalyst*, a material class defined by its function.

***dopedBy*:** indicates dopants (e.g., *chlorine*), i.e., impurities added to a main material (e.g., *SiC*).

***usedTogether*:** connects two MATERIALS if they are used together in an experiment, i.e., if the materials are part of an assembly or a mixture.

| Label | Count | Label | Count |
|---|---|---|---|
| *hasForm* | 2910 | *meas.PropertyValue* | 751 |
| *measuresProperty* | 2080 | *usedTogether* | 672 |
| *usedAs* | 1839 | *conditionEnv.* | 549 |
| *propertyValue* | 1794 | *usedIn* | 434 |
| *conditionProperty* | 1648 | *conditionInstr.* | 357 |
| *conditionSample* | 1434 | *takenFrom* | 118 |
| *cond.PropertyValue* | 1158 | *dopedBy* | 65 |
| *usesTechnique* | 985 | | |

Table 10.4.: Corpus counts for **measurement relations.**

### 10.3.3. Corpus Statistics

Table 10.3 shows the total counts of **named entity** instances with their labels. There are roughly 1.5 MAT annotations per sentence as these are nested and occurrences of composite materials often result in many combined MAT tags. Table 10.4 reports the counts of **annotated relations**, with *hasForm* as the most frequent relation with 2910 instances and *dopedBy* the least frequent with only 65 instances.

**Measurement Sentences.** Out of all 10186 sentences, 2111 (20.7%) describe a measurement (i.e., they contain at least one MEASUREMENT annotation). On average, each document contains 43.4 MEASUREMENT annotations. In addition, there are 1476 sentences (14.5%) marked as containing a QUAL_MEAS, with 40 sentences of these also containing a MEASUREMENT annotation.

## 10.4. Task Definition and Modeling

In this section, we define the relation extraction task on MuLMS and describe our computational approach.

Given an input sentence along with all named entities within it, as well as their types (either gold or predicted depending on the experimental setting), we predict which (if any) relation is present between them. We treat all relations in a single model and predict all relations of a sentence simultaneously by modeling relation extraction as a graph parsing task. Following Toshniwal et al. (2020), we first create an embedding $\mathbf{e}_i$ for the $i^{\text{th}}$ NE in the sentence by concatenating the token embeddings of its first and last token ($\mathbf{e}_{i,\text{START}}$, $\mathbf{e}_{i,\text{END}}$). We also concatenate a learned embedding for the NE's label ($\mathbf{e}_{i,\text{LABEL}}$):

$$\mathbf{e}_i = \mathbf{e}_{i,\text{START}} \oplus \mathbf{e}_{i,\text{END}} \oplus \mathbf{e}_{i,\text{LABEL}} \tag{10.1}$$

Considering NEs as nodes in a graph, we use a biaffine classifier architecture (Dozat and Manning, 2017) as implemented in our STEPS parser to predict the relation between each pair. Non-existence of a relation is encoded as simply another label (∅). For further details on the parser architecture, see Chapter 7 and Chapter 8.

**Pre-trained models.** We use BERT-based models (Devlin et al., 2019) as the underlying text encoder for all of our models. In addition to the original BERT model, we also consider variants, namely SciBERT (Beltagy et al., 2019), which has been pretrained on articles in the scientific domain, and MatSciBERT (Gupta et al., 2022), a version of SciBERT further pre-trained on materials science articles. We use the uncased, 768-dimensional variant of each model, which we fine-tune during training.

## 10.5. Experimental Evaluation

We now detail our experimental results.

### 10.5.1. Experimental Settings

We split the MuLMS corpus into train, dev and test sets on a per-document basis. Within the train set, we provide five distinct tune splits (*train1* to *train5*). For all experiments and for hyperparameter tuning, we always train five models. Similar to cross validation, we train on four folds and use the fifth "training fold" for model selection (cf. van der Goot (2021) for details). Hyperparameters are chosen based on the best dev results, and we finally report results for the test set. The splits are the same across all tasks. Because the training data varies across the five runs for which we report results, standard deviations are usually larger than when using the same training data.

**Hyperparameters and training setup.** We use AdamW (Loshchilov and Hutter, 2019b) as the optimizer for all our models. We use an inverse square-root learning rate scheduler similar to the one used by (Vaswani et al., 2017) where $ws$ refers to the number of warmup steps:

$$lr = \sqrt{ws} \cdot \min(\frac{1}{\sqrt{step\_num}}, step\_num \cdot ws^{-1.5}) \tag{10.2}$$

We use a base learning rate of 4e-5 for all experiments, which we found to perform best in preliminary experiments. We employ early stopping with a patience of 15 epochs

| LM | dev | | test | |
|---|---|---|---|---|
| | Micro F1 | Macro F1 | Micro F1 | Macro F1 |
| *Maj. basel.* | 38.3 | 29.4 | 37.2 | 27.4 |
| BERT | $69.5_{\pm 0.5}$ | $63.4_{\pm 1.0}$ | $63.5_{\pm 0.6}$ | $57.7_{\pm 1.1}$ |
| SciBERT | $72.5_{\pm 0.8}$ | $65.7_{\pm 0.6}$ | $67.5_{\pm 0.9}$ | $\mathbf{62.0}_{\pm 2.2}$ |
| MatSciBERT | $\mathbf{73.2}_{\pm 1.0}$ | $\mathbf{66.5}_{\pm 1.1}$ | $\mathbf{67.6}_{\pm 1.0}$ | $\mathbf{62.0}_{\pm 1.0}$ |

Table 10.5.: **Relation extraction** results: gold entities.

for all experiments. Our models are trained with Nvidia A100 and V100 GPUs using the PyTorch framework (Paszke et al., 2019).

## 10.5.2. Basic Relation Extraction Results

Table 10.5 shows the results for relation extraction on gold entities. A predicted relation is counted as correct if and only if there is a relation with the same start span, end span, and relation label in the set of gold relations for the sentence. The majority baseline assigns to each pair of entities the relation that is most common in the training set for the respective entity types of the governing and dependent spans.

The results demonstrate that a biaffine dependency parsing approach achieves robust performance overall and outperforms the baseline by a substantial margin. The two models trained on scientific text outperform BERT. Their results are similar, with MatSciBERT having a slight edge.

Analysis of per-label scores (see Table 10.6 for MatSciBERT) shows that the highest scores are achieved for *conditionInstrument* (92.2 F1), *usesTechnique* (91.0 F1), and *takenFrom* (84.7). This is somewhat surprising especially for *conditionInstrument* and *takenFrom*, as these are among the rarest relation types in the corpus (see Table 10.4). However, our majority baseline achieves high accuracies on these relation types as well (>90 F1 for *conditionInstrument* and *usesTechnique*), i.e., they are easily inferable from entity types. The worst performance is observed on the relation types *usedTogether* (4.0 F1), *dopedBy* (22.7 F1), and *usedIn* (37.9 F1). These relations occur relatively rarely and also cannot be inferred from entity types.

**Relation extraction on predicted entities.** Finally, we also run our relation extraction module on predicted named entities using the respective best-performing models (both based on MatSciBERT). Models are evaluated as above, with the additional requirement that the boundaries of start/end spans of a predicted relation must also exactly match those of the respective gold spans. Prediction accuracy drops substantially: to micro-F1

| Label | dev | test |
|---|---|---|
| *hasForm* | $71.3_{\pm 0.8}$ | $76.1_{\pm 0.5}$ |
| *measuresProperty* | $88.0_{\pm 1.0}$ | $83.1_{\pm 0.8}$ |
| *measuresPropertyValue* | $84.1_{\pm 2.2}$ | $73.8_{\pm 1.0}$ |
| *usedAs* | $50.2_{\pm 2.3}$ | $41.8_{\pm 1.9}$ |
| *conditionProperty* | $83.0_{\pm 1.3}$ | $72.3_{\pm 1.0}$ |
| *conditionPropertyValue* | $74.7_{\pm 2.5}$ | $63.2_{\pm 2.5}$ |
| *conditionSampleFeatures* | $67.6_{\pm 0.9}$ | $66.0_{\pm 2.1}$ |
| *usesTechnique* | $94.6_{\pm 0.6}$ | $91.0_{\pm 0.9}$ |
| *conditionEnvironment* | $57.1_{\pm 6.2}$ | $39.0_{\pm 3.0}$ |
| *propertyValue* | $86.7_{\pm 1.3}$ | $82.5_{\pm 2.1}$ |
| *usedIn* | $49.3_{\pm 5.6}$ | $37.9_{\pm 3.7}$ |
| *conditionInstrument* | $98.2_{\pm 0.8}$ | $92.2_{\pm 0.9}$ |
| *dopedBy* | $0.0_{\pm 0.0}$ | $22.7_{\pm 18.7}$ |
| *takenFrom* | $85.5_{\pm 3.7}$ | $84.7_{\pm 3.6}$ |
| *usedTogether* | $7.5_{\pm 2.0}$ | $4.0_{\pm 1.7}$ |
| Macro-avg. | $66.5_{\pm 1.1}$ | $62.0_{\pm 1.0}$ |
| Micro-avg. | $73.2_{\pm 1.0}$ | $67.6_{\pm 1.0}$ |

Table 10.6.: Per-label F1 scores for relation extraction from MuLMS using MatSciBERT (gold named entities).

scores of 42.5 and 36.5 on dev and test, respectively, corresponding to macro-F1 scores of 37.9 and 32.8. The reason for this is error propagation as relations can only be retrieved if the entities are predicted correctly, and as incorrectly labeled entities can mislead the relation classifier.

### 10.5.3. Multi-Task Learning Across Datasets

To find out whether information extraction accuracy can be increased by employing multi-task learning (MTL), we perform a series of experiments in which we combine MuLMS training data with relation data from other materials science datasets, namely the MSPT (Mysore et al., 2019) and SOFC-Exp (Friedrich et al., 2020) corpora.[5] In all experiments, we use a shared MatSciBERT encoder and one classification head for each task (dataset). When reporting results on MuLMS, we use the same setup as before, but add the complete training sets of SOFC-Exp or MSPT during training. When reporting results on SOFC-Exp and MSPT, we train on all of their training data and the complete training data of MuLMS, and perform early stopping on dev. For these experiments, reported scores are averages over 5 runs with different random seeds.

[5]We removed one document from the test split of the SOFC-Exp corpus since it is also part of the train set of MuLMS.

| Dataset | dev | | test | |
|---|---|---|---|---|
| | **Micro F1** | **Macro F1** | **Micro F1** | **Macro F1** |
| MuLMS | $73.2_{\pm 1.0}$ | $66.5_{\pm 1.1}$ | $67.6_{\pm 1.0}$ | $62.0_{\pm 1.0}$ |
| + SOFC | $72.5_{\pm 0.9}$ | $65.8_{\pm 0.7}$ | $68.1_{\pm 0.7}$ | $61.1_{\pm 0.7}$ |
| + MSPT | $\mathbf{73.9}_{\pm 0.4}$ | $\mathbf{67.4}_{\pm 0.4}$ | $\mathbf{68.7}_{\pm 0.7}$ | $\mathbf{63.7}_{\pm 0.7}$ |
| SOFC-Exp | $71.3_{\pm 0.6}$ | $62.8_{\pm 2.1}$ | $66.9_{\pm 1.6}$ | $59.8_{\pm 1.5}$ |
| + MuLMS | $\mathbf{72.3}_{\pm 0.5}$ | $\mathbf{64.3}_{\pm 3.7}$ | $\mathbf{68.7}_{\pm 1.5}$ | $\mathbf{60.9}_{\pm 3.6}$ |
| + MSPT | $\mathbf{72.3}_{\pm 0.9}$ | $63.1_{\pm 2.6}$ | $67.6_{\pm 2.0}$ | $60.8_{\pm 4.3}$ |
| MSPT | $84.2_{\pm 0.6}$ | $82.5_{\pm 0.7}$ | $84.6_{\pm 0.8}$ | $83.0_{\pm 0.8}$ |
| + MuLMS | $\mathbf{85.3}_{\pm 0.2}$ | $\mathbf{83.4}_{\pm 0.8}$ | $\mathbf{85.6}_{\pm 0.4}$ | $\mathbf{84.1}_{\pm 0.6}$ |
| + SOFC | $83.7_{\pm 1.4}$ | $81.8_{\pm 1.4}$ | $84.7_{\pm 1.1}$ | $83.2_{\pm 1.4}$ |

Table 10.7.: Relation extraction multi-tasking results using MatSciBERT-based parser.

Results for MTL for relation extraction are shown in Table 10.7. We observe that adding MuLMS to the training data of both SOFC-Exp and MSPT results in improvements. Incorporating SOFC-Exp instances in the training does not meaningfully increase prediction accuracy on MuLMS, whereas incorporating instances from MSPT leads to modest improvements. Intuitively, this makes sense: relations in SOFC-Exp focus on a specific type of experiment, while MuLMS covers a broader range of measurements. Similarly, some MuLMS relations bear resemblance to MSPT relations (e.g., those dealing with instruments or apparatus), which explains why training jointly is beneficial.

## 10.6. Conclusion and Outlook

In this chapter, we have addressed **RQ6 (Can we leverage the biaffine dependency parser architecture for other NLP applications?),** applying our STEPS parser to the task of relation extraction from materials science papers. Our experiments with state-of-the-art neural models highlight that most distinctions can be learned with good accuracy, and that synergies can be achieved by training jointly with existing more specific materials-science NLP datasets.

Future work is needed to improve on end-to-end or joint models of NER and relation extraction as our experiments showed that a pipeline-based setting suffers from error propagation. A potential next step is to adapt sequence-to-sequence models to the structure induction tasks of MuLMS, following ideas of (Hsu et al., 2022; Lu et al., 2021). Finally, employing data augmentation techniques in particular to the less frequent relation types could be a viable path for future work.

# Part V.

# Conclusions

# 11. Summary

Our work in this dissertation was motivated by a simple question: What is the role of dependency structures in modern Natural Language Processing? To obtain an idea of an answer, we have investigated a range of research questions, namely in the areas of (a) datasets and annotation; (b) neural architectures of dependency parsers; and (c) usage of the biaffine parser architecture for other NLP tasks.

With regards to **datasets and annotation,** we have addressed the following research questions:

- **RQ1: What is the quality of existing UD datasets?**
- **RQ2: How can we improve and extend treebanks in an effective way with limited manual resources?**
- **RQ3: What is the effect of improvements to UD treebanks?**

Motivated by these questions, we have made several contributions to English and German UD resources, with a focus on extracting enhanced dependencies from the basic layer. Our main finding is that rule-based extensions and enhancements of existing resources are possible, and that their quality and ultimate usefulness is higher for more regular linguistic phenomena.

With regards to **neural architectures of dependency parsers,** we have addressed the following research questions:

- **RQ4: Which are the main factors determining the performance of neural graph-based dependency parsers?**
- **RQ5: How can neural dependency parsers be adapted for Enhanced UD?**

Using our modular STEPS parser and performing experiments on both basic and enhanced UD data, we have shown that in the presence of modern transformer-based language models, a simple parser architecture consisting of little more than a biaffine classifier head

is often as good as, or even superior to, more complicated setups. However, rule-based components can still be advantageous to ensure graph validity.

Finally, regarding **the usage of biaffine dependency parsing for downstream NLP applications,** we have addressed the following research question:

- **RQ6: Can we leverage the biaffine dependency parser architecture for other NLP applications?**

Here, we have explored the applications of Negation Resolution and Relation Extraction from scientific texts. For Negation Resolution, we have evaluated different ways of encoding the problem as a dependency parsing problem task and using our STEPS parser to predict the resulting dependency structures. We find that using parsing methods for Negation Resolution is a viable approach, though not superior to the arguably simpler method of sequence labeling. For Relation Extraction from scientific texts, we have worked on the MuLMS corpus of materials science documents and have found that a biaffine parsing approach can retrieve domain-specific relations at good accuracy. Additionally, we have found that performance can be increased further by employing a multi-task learning setup across related datasets.

**Discussion.** With this dissertation, we have examined the role of dependency structures in modern NLP from three angles: datasets, parser architectures, and downstream applications.

Our dataset contributions point to the **importance of high-quality annotation data** when working with Universal Dependencies. In a range of experiments, e.g., on coordinate structures or proposition–determiner contractions, we find that improving the consistency of treebanks also leads to accuracy improvements of parsers trained on them. Furthermore, when aiming to create a parser for a new framework and/or language (e.g., enhanced UD for German), it remains a necessary first step to collect or create appropriate training data. Our work shows that it is possible to create such data in a partially automated fashion from existing resources, especially if the linguistic phenomena involved display a high regularity (as in the case of German preposition-determiner contractions). Nonetheless, the process often remains dependent on human effort. Future work may therefore investigate more sophisticated methods of creating such data with the help of computer programs, e.g., using state-of-the-art generative language models.

With regards to **structure in neural (dependency parsing) models,** our findings are more complex. As a general trend, our observations are consistent with a diminishing

importance of explicit structure in NLP systems, as evidenced by the choice of language model being the most important factor impacting parser performance. By contrast, other architectural factors, such as a factorized vs. an unfactorized biaffine architecture, have only a rather small influence on parsing accuracy.

In addition to this main result, however, we argue that our observations also highlight more subtle potential influences of incorporating structural information into neural models. One way of thinking about this issue is to consider the ways in which variations of model architectures and/or task formulations imbue a system with different inductive biases, i.e., a-priori preferences (or even hard constraints) towards certain kinds of solutions. For example, a simple unfactorized biaffine parser architecture does not have an inductive bias towards the single-head property found in basic UD trees; as a result, we find that factorized variants of our STEPS parser achieve moderately higher accuracies and converge significantly faster.

At the same time, it is not trivial to predict which architectural variations actually result in tangible performance boosts. For example, one might intuitively expect multi-task training with POS tags and morphological features to improve parsing performance, as a model may be guided to better latent representations of linguistic information. However, in our experiments we actually found that such as setup hurts parsing accuracy, often quite noticeably. A preliminary conclusion may be that it is more promising to embed information about the task structure directly into the architecture of a model (as in the case of using a factorized biaffine system to parse dependency trees) as opposed to via additional predictions on top of the actual task in question (as in the case of multi-task training).

With our results, we have furthermore identified certain areas in which explicitly encoded structural information continues to be useful in the form of rule-based components of NLP systems. This includes not only problems in which structure is in some sense "inherent" – such as in ensuring the validity of predicted UD graphs via rule-based post-processing – but also other issues such as the prediction of lexicalized dependency labels in Enhanced UD. While lexicalized labels could in theory also be predicted by a purely neural system (by simply increasing the space of possible dependency labels to include all possible lexical material), retrieving the lexical material on the fly in a rule-based fashion serves to reduce the prediction space of the neural model to a manageable level. In general, rule-based subsystems appear useful for tasks with conceptually simple sub-problems that may nonetheless cause the problem space to needlessly inflate when tackled in a naive fashion. For example, outside of dependency parsing, similar approaches may be

used when predicting linearized structures with generative language models, e.g., by constraining their output via formal grammars (Lu et al., 2021; Scholak et al., 2021).

The last main area considered in this dissertation was that of **NLP applications.** In our experiments on Negation Resolution and Relation Extraction, we have found more complex ways of encoding a problem are not necessarily superior when it comes to performance in the relevant metrics. For example, a simple pipeline-based sequence-labeling model outperformed graph-parsing formulations of the Negation Resolution task. One reason for this may lie in the fact that the graph-parsing setup provides only rather sparse feedback to the underlying system, as there is no relationship between most tokens in a sentence (i.e., strong class imbalance towards the ∅ label). This problem is lessened in sequence-labeling schemes as they only consider only individual tokens, rather than all pairs (meaning that the imbalanced towards the ∅ label grows only linearly in sentence length, rather than quadratically). It seems possible that for larger training data sizes, the parsing approach may eventually outperform sequence tagging.

Finally, while this has not been the focus of the present dissertation, linguistic structure will continue to play an important role as a core foundation of theoretical linguistics and as an analysis tool for NLP systems (e.g., Muñoz-Ortiz et al., 2023). Depending on the development of the field, future neuro-symbolic AI systems may also rely on interpretable linguistic structures to offer improved explainability for their decisions.

**Final remarks.** Predictions about the future of scientific progress are challenging almost by definition, and this is especially true in the rapidly evolving field of NLP. Recent developments in neural architectures and large language models have shown that progress can occur at a remarkable pace, disrupting established research paradigms as a result. With this in mind, we hope that our dissertation may serve as a motivation for continued research on structured representations in NLP, for example as part of the growing research program of neuro-symbolic Artificial Intelligence (Sarker et al., 2021).

# Bibliography


Abend, O. and Rappoport, A. (2013). Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

Alt, C., Gabryszak, A., and Hennig, L. (2020). Probing linguistic features of sentence-level representations in neural relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1534–1545, Online. Association for Computational Linguistics.

Altuna, B., Minard, A.-L., and Speranza, M. (2017). The scope and focus of negation: A complete annotation framework for Italian. In *Proceedings of the Workshop Computational Semantics Beyond Events and Roles*, pages 34–42, Valencia, Spain. Association for Computational Linguistics.

Alzetta, C., Dell'Orletta, F., Montemagni, S., Simi, M., and Venturi, G. (2018). Assessing the impact of incremental error detection and correction. a case study on the Italian universal dependency treebank. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 1–7, Brussels, Belgium. Association for Computational Linguistics.

Arkhipov, M., Trofimova, M., Kuratov, Y., and Sorokin, A. (2019). Tuning multilingual transformers for language-specific named entity recognition. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 89–93, Florence, Italy. Association for Computational Linguistics.

Attardi, G., Sartiano, D., and Simi, M. (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.

Aufrant, L. and Wisniewski, G. (2018). Panparser: a modular implementation for efficient transition-based dependency parsing. *The Prague Bulletin of Mathematical Linguistics*,

111(1):57–86.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Bejček, E., Panevová, J., Popelka, J., Straňák, P., Ševčíková, M., Štěpánek, J., and Žabokrtský, Z. (2012). Prague Dependency Treebank 2.5 – a revisited version of PDT 2.0. In *Proceedings of COLING 2012*, pages 231–246, Mumbai, India. The COLING 2012 Organizing Committee.

Beltagy, I., Cohan, A., Feigenblat, G., Freitag, D., Ghosal, T., Hall, K., Herrmannova, D., Knoth, P., Lo, K., Mayr, P., Patton, R. M., Shmueli-Scheuer, M., de Waard, A., Wang, K., and Wang, L. L., editors (2021). *Proceedings of the Second Workshop on Scholarly Document Processing*, Online. Association for Computational Linguistics.

Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Bender, E. M. and Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016). Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas. Association for Computational Linguistics.

Bhat, R. A., Bhatt, R., Farudi, A., Klassen, P., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D. M., Vaidya, A., Vishnu, S. R., et al. (2017). The hindi/urdu treebank project. In *Handbook of Linguistic Annotation*. Springer Press.

Bhattacharya, A. (1943). On a measure of divergence between two statistical populations defined by their population distributions. In *Bulletin of the Calcutta Mathematical Society*, volume 35, pages 99–109.

Blanco, E. and Moldovan, D. (2011a). Semantic representation of negation using focus detection. In *Proceedings of the 49th Annual Meeting of the Association for Computa-*

*tional Linguistics: Human Language Technologies*, pages 581–589, Portland, Oregon, USA. Association for Computational Linguistics.

Blanco, E. and Moldovan, D. (2011b). Some issues on detecting negation from text. In *Twenty-Fourth International FLAIRS Conference*.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A. S., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N. D., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M. S., Krishna, R., Kuditipudi, R., and et al. (2021). On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258.

Bonfante, G., Guillaume, B., and Perrier, G. (2018). *Application of Graph Rewriting to Natural Language Processing*. John Wiley & Sons.

Borges Völker, E., Wendt, M., Hennig, F., and Köhn, A. (2019). HDT-UD: A very large Universal Dependencies treebank for German. In *Proceedings of the Third Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*, pages 46–57, Paris, France. Association for Computational Linguistics.

Bouma, G., Seddah, D., and Zeman, D. (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.

Bouma, G., Seddah, D., and Zeman, D. (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 and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 146–157, Online. Association for Computational Linguistics.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Britto, B. K. and Khandelwal, A. (2020). Resolving the scope of speculation and negation using transformer-based architectures. *arXiv preprint arXiv:2001.02885*.

Bürkle, T., Grünewald, S., and Friedrich, A. (2021). A corpus study of creating rule-based enhanced Universal Dependencies for German. In *Proceedings of The Joint 15th*

*Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, pages 85–95, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Candito, M., Guillaume, B., Perrier, G., and Seddah, D. (2017). Enhanced UD dependencies with neutralized diathesis alternation. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pages 42–53, Pisa,Italy. Linköping University Electronic Press.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D., Úlfar Erlingsson, Oprea, A., and Raffel, C. (2020). Extracting training data from large language models. *CoRR*, abs/2012.07805.

Chandrasekaran, M. K., de Waard, A., Feigenblat, G., Freitag, D., Ghosal, T., Hovy, E., Knoth, P., Konopnicki, D., Mayr, P., Patton, R. M., and Shmueli-Scheuer, M., editors (2020). *Proceedings of the First Workshop on Scholarly Document Processing*, Online. Association for Computational Linguistics.

Che, W., Liu, Y., Wang, Y., Zheng, B., and Liu, T. (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.

Chu, Y.-J. and Liu, T.-H. (1965). On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Chun, J., Han, N.-R., Hwang, J. D., and Choi, J. D. (2018). Building Universal Dependency treebanks in Korean. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Cohan, A., Feigenblat, G., Freitag, D., Ghosal, T., Herrmannova, D., Knoth, P., Lo, K., Mayr, P., Shmueli-Scheuer, M., de Waard, A., and Wang, L. L., editors (2022). *Proceedings of the Third Workshop on Scholarly Document Processing*, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Cohen, K. B., Demner-Fushman, D., Ananiadou, S., Webber, B., Tsujii, J., and Pestian, J., editors (2012). *BioNLP: Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, Montréal, Canada. Association for Computational Linguistics.

Collier, N., Ruch, P., and Nazarenko, A., editors (2004). *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications, NLPBA/BioNLP 2004, Geneva, Switzerland, August 28-29, 2004*.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. pages 8440–8451.

de Lhoneux, M., Stymne, S., and Nivre, J. (2017). Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement. In *The 15th Treebanks and Linguistic Theories Workshop (TLT)*, pages 99–110.

de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592, Reykjavik, Iceland. European Language Resources Association (ELRA).

De Marneffe, M.-C. and Manning, C. D. (2008). Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

de Marneffe, M.-C. and Manning, C. D. (2008). The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Manchester, UK. Coling 2008 Organizing Committee.

Dehouck, M., Anderson, M., and Gómez-Rodríguez, C. (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.

Demner-Fushman, D., Cohen, K. B., Ananiadou, S., and Tsujii, J., editors (2022). *Proceedings of the 21st Workshop on Biomedical Language Processing*, Dublin, Ireland. Association for Computational Linguistics.

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

Dozat, T. and Manning, C. D. (2017). Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Dozat, T. and Manning, C. D. (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.

Dozat, T., Qi, P., and Manning, C. D. (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.

Droganova, K., Lyashevskaya, O., and Zeman, D. (2018). Data conversion and consistency of monolingual corpora: Russian ud treebanks. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018), December 13–14, 2018, Oslo University, Norway*, number 155, pages 52–65. Linköping University Electronic Press.

Droganova, K. and Zeman, D. (2019). Towards deep Universal Dependencies. In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 144–152, Paris, France. Association for Computational Linguistics.

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

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

Enger, M., Velldal, E., and Øvrelid, L. (2017). An open-source tool for negation detection: a maximum-margin approach. In *Proceedings of the Workshop Computational Semantics Beyond Events and Roles*, pages 64–69, Valencia, Spain. Association for Computational Linguistics.

Falenska, A. and Kuhn, J. (2019). The (non-)utility of structural features in BiLSTM-based dependency parsers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 117–128, Florence, Italy. Association for Computational Linguistics.

Fancellu, F., Lopez, A., and Webber, B. (2016). Neural networks for negation scope detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 495–504, Berlin, Germany. Association for Computational Linguistics.

Fancellu, F., Lopez, A., Webber, B., and He, H. (2017). Detecting negation scope is easy, except when it isn't. In *Proceedings of the 15th Conference of the European Chapter of*

*the Association for Computational Linguistics: Volume 2, Short Papers*, pages 58–63, Valencia, Spain. Association for Computational Linguistics.

Fancellu, F., Lopez, A., and Webber, B. L. (2018). Neural networks for cross-lingual negation scope detection. *CoRR*, abs/1810.02156.

Fancellu, F. and Webber, B. (2015). Translating negation: Induction, search and model errors. In *Proceedings of the Ninth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 21–29, Denver, Colorado, USA. Association for Computational Linguistics.

Farkas, R., Vincze, V., Móra, G., Csirik, J., and Szarvas, G. (2010). The CoNLL-2010 shared task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning – Shared Task*, pages 1–12, Uppsala, Sweden. Association for Computational Linguistics.

Fillmore, C. J. and Baker, C. F. (2001). Frame semantics for text understanding. In *Proceedings of WordNet and Other Lexical Resources Workshop, NAACL*, volume 6.

Fonseca, E. and Martins, A. F. T. (2020). Revisiting higher-order dependency parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8795–8800, Online. Association for Computational Linguistics.

Foth, K. A. (2006). Eine umfassende constraint-dependenz-grammatik des deutschen.

Foth, K. A., Köhn, A., Beuck, N., and Menzel, W. (2014). Because size does matter: The hamburg dependency treebank. In Chair), N. C. C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Friedrich, A., Adel, H., Tomazic, F., Hingerl, J., Benteau, R., Marusczyk, A., and Lange, L. (2020). The SOFC-exp corpus and neural approaches to information extraction in the materials science domain. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1255–1268, Online. Association for Computational Linguistics.

Gashteovski, K., Gemulla, R., and del Corro, L. (2017). MinIE: Minimizing facts in open information extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2630–2640, Copenhagen, Denmark. Association for Computational Linguistics.

Geiger, A., Richardson, K., and Potts, C. (2020). Neural natural language inference models partially embed theories of lexical entailment and negation. In *Proceedings of*

*the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 163–173, Online. Association for Computational Linguistics.

Gerdes, K., Guillaume, B., Kahane, S., and Perrier, G. (2018). SUD or surface-syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 66–74, Brussels, Belgium. Association for Computational Linguistics.

Gerdes, K. and Kahane, S. (2016). Dependency annotation choices: Assessing theoretical and practical issues of Universal Dependencies. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 131–140, Berlin, Germany. Association for Computational Linguistics.

Gontrum, J., Groschwitz, J., Koller, A., and Teichmann, C. (2017). Alto: Rapid prototyping for parsing and translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 29–32, Valencia, Spain. Association for Computational Linguistics.

Gregoire, N., Evert, S., and Kim, S. N., editors (2007). *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, Prague, Czech Republic. Association for Computational Linguistics.

Grünewald, S. and Friedrich, A. (2020a). 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.

Grünewald, S. and Friedrich, A. (2020b). Unifying the treatment of preposition-determiner contractions in German Universal Dependencies treebanks. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 94–98, Barcelona, Spain (Online). Association for Computational Linguistics.

Grünewald, S., Friedrich, A., and Kuhn, J. (2021a). Applying Occam's razor to transformer-based dependency parsing: What works, what doesn't, and what is really necessary. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 131–144, Online. Association for Computational Linguistics.

Grünewald, S., Oertel, F. T., and Friedrich, A. (2021b). RobertNLP at the IWPT 2021 shared task: Simple enhanced UD parsing for 17 languages. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 196–203, Online.

Association for Computational Linguistics.

Grünewald, S., Piccirilli, P., and Friedrich, A. (2021c). 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.

Guillaume, B., Bonfante, G., Masson, P., Morey, M., and Perrier, G. (2012). Grew : un outil de réécriture de graphes pour le TAL (Grew: a graph rewriting tool for NLP) [in French]. In Besacier, L., Blanchon, H., and Sérasset, G., editors, *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 5: Software Demonstrations*, pages 1–2, Grenoble, France. ATALA/AFCP.

Gupta, T., Zaki, M., Krishnan, N. M. A., and Mausam (2022). MatSciBERT: A materials domain language model for text mining and information extraction. *npj Computational Materials*, 8(1):102.

Haverinen, K., Nyblom, J., Viljanen, T., Laippala, V., Kohonen, S., Missilä, A., Ojala, S., Salakoski, T., and Ginter, F. (2014). Building the essential resources for finnish: the turku dependency treebank. *Language Resources and Evaluation*, 48(3):493–531.

He, H. and Choi, J. D. (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.

He, L., Lee, K., Levy, O., and Zettlemoyer, L. (2018). Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia. Association for Computational Linguistics.

Heinecke, J. (2019). ConlluEditor: a fully graphical editor for universal dependencies treebank files. In *Proceedings of the Third Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*, pages 87–93, Paris, France. Association for Computational Linguistics.

Heinecke, J. (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.

Hewitt, J. and Manning, C. D. (2019). A structural probe for finding syntax in word rep-

resentations. 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. Association for Computational Linguistics.

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

Horn, L. R., editor (2010). *The Expression of Negation*. De Gruyter Mouton.

Hossain, M. M., Anastasopoulos, A., Blanco, E., and Palmer, A. (2020a). It's not a non-issue: Negation as a source of error in machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3869–3885, Online. Association for Computational Linguistics.

Hossain, M. M., Hamilton, K., Palmer, A., and Blanco, E. (2020b). Predicting the focus of negation: Model and error analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8389–8401, Online. Association for Computational Linguistics.

Hossain, M. M., Kovatchev, V., Dutta, P., Kao, T., Wei, E., and Blanco, E. (2020c). An analysis of natural language inference benchmarks through the lens of negation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9106–9118, Online. Association for Computational Linguistics.

Hosseini, A., Reddy, S., Bahdanau, D., Hjelm, R. D., Sordoni, A., and Courville, A. (2021). Understanding by understanding not: Modeling negation in language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1301–1312, Online. Association for Computational Linguistics.

Hsu, I.-H., Huang, K.-H., Boschee, E., Miller, S., Natarajan, P., Chang, K.-W., and Peng, N. (2022). DEGREE: A data-efficient generation-based event extraction model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1890–1908, Seattle, United States. Association for Computational Linguistics.

Huang, K.-H., Yang, M., and Peng, N. (2020). Biomedical event extraction with hierarchical knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1277–1285, Online. Association for Computational Linguistics.

Huddleston, R. and Pullum, G. K. (2002). *The Cambridge Grammar of the English Language*. Cambridge University Press.

Jiang, L., Bosselut, A., Bhagavatula, C., and Choi, Y. (2021). "I'm not mad": Commonsense implications of negation and contradiction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4380–4397, Online. Association for Computational Linguistics.

Jie, Z. and Lu, W. (2019). Dependency-guided LSTM-CRF for named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3862–3872, Hong Kong, China. Association for Computational Linguistics.

Jiménez-Zafra, S. M., Morante, R., Blanco, E., Martín Valdivia, M. T., and Ureña López, L. A. (2020). Detecting negation cues and scopes in Spanish. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6902–6911, Marseille, France. European Language Resources Association.

Jiménez-Zafra, S. M., Morante, R., Martin, M., and Ureña-López, L. A. (2018). A review of Spanish corpora annotated with negation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 915–924, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jiménez-Zafra, S. M., Morante, R., Teresa Martín-Valdivia, M., and Ureña-López, L. A. (2020). Corpora Annotated with Negation: An Overview. *Computational Linguistics*, 46(1):1–52.

Jurafsky, D. and Martin, J. H. (2021). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* 3rd ed. draft.

Kanerva, J., Ginter, F., Miekka, N., Leino, A., and Salakoski, T. (2018). Turku neural parser pipeline: An end-to-end system for the CoNLL 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 133–142, Brussels, Belgium. Association for Computational Linguistics.

Kanerva, J., Ginter, F., and Pyysalo, S. (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.

Khandelwal, A. and Britto, B. K. (2020). Multitask learning of negation and speculation

using transformers. In *Proceedings of the 11th International Workshop on Health Text Mining and Information Analysis*, pages 79–87, Online. Association for Computational Linguistics.

Khandelwal, A. and Sawant, S. (2020). NegBERT: A transfer learning approach for negation detection and scope resolution. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5739–5748, Marseille, France. European Language Resources Association.

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

Kondratyuk, D. and Straka, M. (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 (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Konstantinova, N., de Sousa, S. C., Cruz, N. P., Maña, M. J., Taboada, M., and Mitkov, R. (2012). A review corpus annotated for negation, speculation and their scope. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3190–3195, Istanbul, Turkey. European Language Resources Association (ELRA).

Kulmizev, A., de Lhoneux, M., Gontrum, J., Fano, E., and Nivre, J. (2019). Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2755–2768, Hong Kong, China. Association for Computational Linguistics.

Kuratov, Y. and Arkhipov, M. (2019). Adaptation of deep bidirectional multilingual transformers for russian language. *arXiv preprint arXiv:1905.07213*.

Kurtz, R., Oepen, S., and Kuhlmann, M. (2020). End-to-end negation resolution as graph parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 14–24, Online. Association for Computational Linguistics.

Lapponi, E., Velldal, E., Øvrelid, L., and Read, J. (2012). UiO 2: Sequence-labeling negation using dependency features. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference*

*and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 319–327, Montréal, Canada. Association for Computational Linguistics.

Lazib, L., Qin, B., Zhao, Y., Zhang, W., and Liu, T. (2020). A syntactic path-based hybrid neural network for negation scope detection. *Frontiers of computer science*, 14(1):84–94.

Lee, S., Jang, H., Baik, Y., Park, S., and Shin, H. (2020). Kr-bert: A small-scale korean-specific language model. *arXiv preprint arXiv:2008.03979*.

Lehmann, C. (2002). New reflections on grammaticalization and lexicalization. In *New Reflections on Grammaticalization*, pages 1–18. John Benjamins.

Li, J., Zhou, G., Wang, H., and Zhu, Q. (2010). Learning the scope of negation via shallow semantic parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 671–679, Beijing, China. Coling 2010 Organizing Committee.

Liu, Q., Fancellu, F., and Webber, B. (2018). NegPar: A parallel corpus annotated for negation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

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

Loper, E. and Bird, S. (2002). NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Loshchilov, I. and Hutter, F. (2019a). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Loshchilov, I. and Hutter, F. (2019b). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Lu, Y., Lin, H., Xu, J., Han, X., Tang, J., Li, A., Sun, L., Liao, M., and Chen, S. (2021). Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.

Marcheggiani, D. and Titov, I. (2020). Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3915–3928, Online. Association for Computational Linguistics.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

May, J. and Priyadarshi, J. (2017). SemEval-2017 task 9: Abstract Meaning Representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545, Vancouver, Canada. Association for Computational Linguistics.

McDonald, R. and Nivre, J. (2007). Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131, Prague, Czech Republic. Association for Computational Linguistics.

McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., Hall, K., Petrov, S., Zhang, H., Täckström, O., Bedini, C., Bertomeu Castelló, N., and Lee, J. (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.

McKenna, N. and Steedman, M. (2020). Learning negation scope from syntactic structure. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 137–142, Barcelona, Spain (Online). Association for Computational Linguistics.

Mehrabi, S., Krishnan, A., Sohn, S., Roch, A. M., Schmidt, H., Kesterson, J., Beesley, C., Dexter, P., Max Schmidt, C., Liu, H., and Palakal, M. (2015). Deepen: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of biomedical informatics*, 54:213—219.

Michael, J., Holtzman, A., Parrish, A., Mueller, A., Wang, A., Chen, A., Madaan, D., Nangia, N., Pang, R. Y., Phang, J., et al. (2022). What do nlp researchers believe? results of the nlp community metasurvey. *arXiv preprint arXiv:2208.12852*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Moore, A. and Barnes, J. (2021). Multi-task learning of negation and speculation for targeted sentiment classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2838–2869, Online. Association for Computational Linguistics.

Morante, R. and Blanco, E. (2012). *SEM 2012 shared task: Resolving the scope and focus of negation. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 265–274, Montréal, Canada. Association for Computational Linguistics.

Morante, R. and Blanco, E. (2021). Recent advances in processing negation. *Natural Language Engineering*, 27(2):121–130.

Morante, R. and Daelemans, W. (2012). ConanDoyle-neg: Annotation of negation cues and their scope in conan doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1563–1568, Istanbul, Turkey. European Language Resources Association (ELRA).

Morante, R., Schrauwen, S., and Daelemans, W. (2011). Annotation of negation cues and their scope: Guidelines v1. *Computational linguistics and psycholinguistics technical report series, CTRS-003*, pages 1–42.

Muñoz-Ortiz, A., Gómez-Rodríguez, C., and Vilares, D. (2023). Contrasting linguistic patterns in human and LLM-generated text. *arXiv preprint arXiv:2308.09067*.

Myers, S. and Palmer, M. (2019). ClearTAC: Verb tense, aspect, and form classification using neural nets. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 136–140, Florence, Italy. Association for Computational Linguistics.

Mysore, S., Jensen, Z., Kim, E., Huang, K., Chang, H.-S., Strubell, E., Flanigan, J., McCallum, A., and Olivetti, E. (2019). The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 56–64, Florence, Italy. Association for Computational Linguistics.

Mysore, S., Kim, E., Strubell, E., Liu, A., Chang, H., Kompella, S., Huang, K., McCallum, A., and Olivetti, E. (2017). Automatically extracting action graphs from materials science synthesis procedures. *NIPS Workshop on Machine Learning for Molecules and Materials*.

Nguyen, M. V., Lai, V. D., Pouran Ben Veyseh, A., and Nguyen, T. H. (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.

Nivre, J. (2009). Non-projective dependency parsing in expected linear time. 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 351–359, Suntec, Singapore. Association for Computational Linguistics.

Nivre, J. and Fang, C.-T. (2017). Universal Dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 86–95, Gothenburg, Sweden. Association for Computational Linguistics.

Nivre, J., Marongiu, P., Ginter, F., Kanerva, J., Montemagni, S., Schuster, S., and Simi, M. (2018). Enhancing universal dependency treebanks: A case study. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 102–107, Brussels, Belgium. Association for Computational Linguistics.

Nyblom, J., Kohonen, S., Haverinen, K., Salakoski, T., and Ginter, F. (2013). Predicting conjunct propagation and other extended Stanford dependencies. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 252–261, Prague, Czech Republic. Charles University in Prague, Matfyzpress, Prague, Czech Republic.

Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Cinková, S., Flickinger, D., Hajič, J., Ivanova, A., and Urešová, Z. (2016). Towards comparability of linguistic graph Banks for semantic parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3991–3995, Portorož, Slovenia. European Language Resources Association (ELRA).

Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Cinková, S., Flickinger, D., Hajič, J., and Urešová, Z. (2015). SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Flickinger, D., Hajič, J., Ivanova, A., and Zhang, Y. (2014). SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland. Association for Computational Linguistics.

O'Gorman, T., Jensen, Z., Mysore, S., Huang, K., Mahbub, R., Olivetti, E., and

McCallum, A. (2021). MS-mentions: Consistently annotating entity mentions in materials science procedural text. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1337–1352, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Packard, W., Bender, E. M., Read, J., Oepen, S., and Dridan, R. (2014). Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 69–78, Baltimore, Maryland. Association for Computational Linguistics.

Palmer, M., Bhatt, R., Narasimhan, B., Rambow, O., Sharma, D. M., and Xia, F. (2009). Hindi syntax: Annotating dependency, lexical predicate-argument structure, and phrase structure. In *The 7th International Conference on Natural Language Processing*, pages 14–17.

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

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (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. Association for Computational Linguistics.

Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K., and Gurevych, I. (2020a). AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Pfeiffer, J., Vulić, I., Gurevych, I., and Ruder, S. (2020b). MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Polinsky, M. (2013). *Raising and control*. Cambridge University Press, Cambridge.

Pretkalniņa, L., Rituma, L., and Saulīte, B. (2018). Deriving enhanced universal

dependencies from a hybrid dependency-constituency treebank. In *International Conference on Text, Speech, and Dialogue*, pages 95–105. Springer.

Pyysalo, S., Kanerva, J., Virtanen, A., and Ginter, F. (2020). Wikibert models: deep transfer learning for many languages. *arXiv preprint arXiv:2006.01538*.

Qi, P., Dozat, T., Zhang, Y., and Manning, C. D. (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.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (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*.

Qian, Z., Li, P., Zhu, Q., Zhou, G., Luo, Z., and Luo, W. (2016). Speculation and negation scope detection via convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 815–825, Austin, Texas. Association for Computational Linguistics.

Ramm, A., Loáiciga, S., Friedrich, A., and Fraser, A. (2017). Annotating tense, mood and voice for English, French and German. In *Proceedings of ACL 2017, System Demonstrations*, pages 1–6, Vancouver, Canada. Association for Computational Linguistics.

Ramponi, A., Plank, B., and Lombardo, R. (2020). Cross-domain evaluation of edge detection for biomedical event extraction. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1982–1989, Marseille, France. European Language Resources Association.

Read, J., Velldal, E., Øvrelid, L., and Oepen, S. (2012). UiO1: Constituent-based discriminative ranking for negation resolution. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 310–318, Montréal, Canada. Association for Computational Linguistics.

Ren, Y., Fei, H., and Peng, Q. (2018). Detecting the scope of negation and speculation in biomedical texts by using recursive neural network. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 739–742, Los Alamitos, CA, USA. IEEE Computer Society.

Ribeyre, C., Seddah, D., and Villemonte de la Clergerie, E. (2012). A linguistically-

motivated 2-stage tree to graph transformation. In Satta, G. and Han, C.-H., editors, *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 214–222, Paris, France.

Ruder, S. and Plank, B. (2017). Learning to select data for transfer learning with Bayesian optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 372–382, Copenhagen, Denmark. Association for Computational Linguistics.

Safaya, A., Abdullatif, M., and Yuret, D. (2020). Kuisail at semeval-2020 task 12: Bert-cnn for offensive speech identification in social media.

Saha, S., Nie, Y., and Bansal, M. (2020). ConjNLI: Natural language inference over conjunctive sentences. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8240–8252, Online. Association for Computational Linguistics.

Salomoni, A. (2017). Toward a treebank collecting German aesthetic writings of the late 18th century. In *Proceedings of the Fourth Italian Conference on Computational Linguistics (CLiC-it)*, pages 292–297.

Sanchez Graillet, O. and Poesio, M. (2007). Negation of protein protein interactions: Analysis and extraction. *Bioinformatics (Oxford, England)*, 23:i424–32.

Sarker, M. K., Zhou, L., Eberhart, A., and Hitzler, P. (2021). Neuro-symbolic artificial intelligence. *AI Commun.*, 34(3):197–209.

Sarrouti, M., Tao, C., and Mamy Randriamihaja, Y. (2022). Comparing encoder-only and encoder-decoder transformers for relation extraction from biomedical texts: An empirical study on ten benchmark datasets. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, pages 376–382, Dublin, Ireland. Association for Computational Linguistics.

Savary, A., Escartín, C. P., Bond, F., Mitrović, J., and Mititelu, V. B., editors (2019). *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, Florence, Italy. Association for Computational Linguistics.

Savary, A., Ramisch, C., Hwang, J. D., Schneider, N., Andresen, M., Pradhan, S., and Petruck, M. R. L., editors (2018). *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Schiller, A., Teufel, S., and Thielen, C. (1995). Guidelines für das Tagging deutscher Textcorpora mit STTS. *Universität Stuttgart, Universität Tübingen, Germany.*

Scholak, T., Schucher, N., and Bahdanau, D. (2021). PICARD: Parsing incrementally

for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Schrader, T. P., Finco, M., Grünewald, S., Hildebrand, F., and Friedrich, A. (2023). MuLMS: A multi-layer annotated text corpus for information extraction in the materials science domain. In *Proceedings of the Second Workshop on Information Extraction from Scientific Publications*, pages 84–100, Bali, Indonesia. Association for Computational Linguistics.

Schuster, S., de La Clergerie, É. V., Candito, M., Sagot, B., Manning, C., and Seddah, D. (2017). Paris and Stanford at EPE 2017: Downstream evaluation of graph-based dependency representations. In *Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017)*, pages 47–59, Pisa, Italy.

Schuster, S. and Manning, C. D. (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 (LREC'16)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association (ELRA).

Schuster, S., Nivre, J., and Manning, C. D. (2018). Sentences with gapping: Parsing and reconstructing elided predicates. 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 1156–1168, New Orleans, Louisiana. Association for Computational Linguistics.

Shaitarova, A. and Rinaldi, F. (2021). Negation typology and general representation models for cross-lingual zero-shot negation scope resolution in Russian, French, and Spanish. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 15–23, Online. Association for Computational Linguistics.

Shi, T., Malioutov, I., and Irsoy, O. (2020). Semantic role labeling as syntactic dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7551–7571, Online. Association for Computational Linguistics.

Silveira, N., Dozat, T., de Marneffe, M.-C., Bowman, S., Connor, M., Bauer, J., and Manning, C. D. (2014). A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-*

*2014)*.

Simi, M. and Montemagni, S. (2018). Bootstrapping enhanced universal dependencies for Italian. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253. CEUR-WS.

Sineva, E., Grünewald, S., Friedrich, A., and Kuhn, J. (2021). Negation-instance based evaluation of end-to-end negation resolution. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 528–543, Online. Association for Computational Linguistics.

Smith, A., de Lhoneux, M., Stymne, S., and Nivre, J. (2018). An investigation of the interactions between pre-trained word embeddings, character models and POS tags in dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720, Brussels, Belgium. Association for Computational Linguistics.

Smrž, O., Bielický, V., Kouřilová, I., Kráčmar, J., Hajič, J., and Zemánek, P. (2008). Prague arabic dependency treebank: A word on the million words. In *Proceedings of the Workshop on Arabic and Local Languages (LREC 2008)*, pages 16–23, Marrakech, Morocco.

Sohn, S., Wu, S., and Chute, C. (2012). Dependency parser-based negation detection in clinical narratives. *AMIA Summits on Translational Science proceedings AMIA Summit on Translational Science*, 2012:1–8.

Straka, M. (2018). UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Straka, M., Straková, J., and Hajič, J. (2019). Evaluating contextualized embeddings on 54 languages in pos tagging, lemmatization and dependency parsing. *arXiv preprint arXiv:1908.07448*.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Strzyz, M., Vilares, D., and Gómez-Rodríguez, C. (2019). Viable dependency parsing as sequence labeling. 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 717–723, Minneapolis, Minnesota. Association

for Computational Linguistics.

Swarup, D., Bajaj, A., Mysore, S., O'Gorman, T., Das, R., and McCallum, A. (2020). An Instance Level Approach for Shallow Semantic Parsing in Scientific Procedural Text. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3010–3017, Online. Association for Computational Linguistics.

Szarvas, G., Vincze, V., Farkas, R., and Csirik, J. (2008). The BioScope corpus: annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 38–45, Columbus, Ohio. Association for Computational Linguistics.

Taboada, M., Anthony, C., and Voll, K. (2006). Methods for creating semantic orientation dictionaries. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Tenney, I., Das, D., and Pavlick, E. (2019a). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S., Das, D., and Pavlick, E. (2019b). What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Toshniwal, S., Shi, H., Shi, B., Gao, L., Livescu, K., and Gimpel, K. (2020). A cross-task analysis of text span representations. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 166–176, Online. Association for Computational Linguistics.

Üstün, A., Bisazza, A., Bouma, G., and van Noord, G. (2020). UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

van der Goot, R. (2021). We need to talk about train-dev-test splits. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4485–4494, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg,

U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Velldal, E., Øvrelid, L., Read, J., and Oepen, S. (2012). Speculation and Negation: Rules, Rankers, and the Role of Syntax. *Computational Linguistics*, 38(2):369–410.

Virtanen, A., Kanerva, J., Ilo, R., Luoma, J., Luotolahti, J., Salakoski, T., Ginter, F., and Pyysalo, S. (2019). Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*.

Wang, X., Jiang, Y., and Tu, K. (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.

Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P.-S., Cheng, M., Glaese, M., Balle, B., Kasirzadeh, A., Kenton, Z., Brown, S., Hawkins, W., Stepleton, T., Biles, C., Birhane, A., Haas, J., Rimell, L., Hendricks, L. A., Isaac, W. S., Legassick, S., Irving, G., and Gabriel, I. (2021). Ethical and social risks of harm from language models. *CoRR*, abs/2112.04359.

Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., and Grave, E. (2020). CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

White, A. S., Reisinger, D., Sakaguchi, K., Vieira, T., Zhang, S., Rudinger, R., Rawlins, K., and Van Durme, B. (2016). Universal decompositional semantics on Universal Dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Wiegand, M., Balahur, A., Roth, B., Klakow, D., and Montoyo, A. (2010). A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 60–68, Uppsala, Sweden. University of Antwerp.

Wisniewski, G. (2018). Errator: a tool to help detect annotation errors in the universal dependencies project. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yu, J., Bohnet, B., and Poesio, M. (2020). Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J., and Petrov, S. (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.

Zeman, D., Popel, M., Straka, M., Hajič, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gokirmak, M., Nedoluzhko, A., Cinková, S., Hajič jr., J., Hlaváčová, J., Kettnerová, V., Urešová, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C. D., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., de Paiva, V., Droganova, K., Martínez Alonso, H., Çöltekin, Ç., Sulubacak, U., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Alcalde, H. F., Strnadová, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonça, G., Lando, T., Nitisaroj, R., and Li, J. (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.

Zhang, Y., Xia, Q., Zhou, S., Jiang, Y., Li, Z., Fu, G., and Zhang, M. (2021). Semantic role labeling as dependency parsing: Exploring latent tree structures inside arguments.

*arXiv preprint arXiv:2110.06865*.

Zou, B., Zhu, Q., and Zhou, G. (2015). Negation and speculation identification in Chinese language. 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 656–665, Beijing, China. Association for Computational Linguistics.