ACTA UNIVERSITATIS UPSALIENSIS
Studia Linguistica Upsaliensia
30

Artur Kulmizev

The Search for Syntax

Investigating the Syntactic Knowledge of Neural Language Models Through the Lens of Dependency Parsing

UPPSALA
UNIVERSITET

Dissertation presented at Uppsala University to be publicly examined in Humanistiska Teatern, Engelska parken, Thunbergsvägen 3C, Uppsala, Friday, 22 September 2023 at 14:00 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Professor Roger Levy (Massachusetts Institute of Technology).

Abstract
Kulmizev, A. 2023. The Search for Syntax. Investigating the Syntactic Knowledge of Neural Language Models Through the Lens of Dependency Parsing. *Studia Linguistica Upsaliensia* 30. 101 pp. Uppsala: Acta Universitatis Upsaliensis. ISBN 978-91-513-1850-9.

Syntax — the study of the hierarchical structure of language — has long featured as a prominent research topic in the field of natural language processing (NLP). Traditionally, its role in NLP was confined towards developing parsers: supervised algorithms tasked with predicting the structure of utterances (often for use in downstream applications). More recently, however, syntax (and syntactic theory) has factored much less into the development of NLP models, and much more into their analysis. This has been particularly true with the nascent relevance of language models: semi-supervised algorithms trained to predict (or infill) strings given a provided context. In this dissertation, I describe four separate studies that seek to explore the interplay between syntactic parsers and language models upon the backdrop of dependency syntax. In the first study, I investigate the error profiles of neural transition-based and graph-based dependency parsers, showing that they are effectively homogenized when leveraging representations from pre-trained language models. Following this, I report the results of two additional studies which show that dependency tree structure can be partially decoded from the internal components of neural language models — specifically, hidden state representations and self-attention distributions. I then expand on these findings by exploring a set of additional results, which serve to highlight the influence of experimental factors, such as the choice of annotation framework or learning objective, in decoding syntactic structure from model components. In the final study, I describe efforts to quantify the overall learnability of a large set of multilingual dependency treebanks — the data upon which the previous experiments were based — and how it may be affected by factors such as annotation quality or tokenization decisions. Finally, I conclude the thesis with a conceptual analysis that relates the aforementioned studies to a broader body of work concerning the syntactic knowledge of language models.

Keywords: syntax, language models, dependency parsing, universal dependencies

Artur Kulmizev, Department of Linguistics and Philology, Box 635, Uppsala University, SE-75126 Uppsala, Sweden.

ISSN 1652-1366
ISBN 978-91-513-1850-9
URN urn:nbn:se:uu:diva-508379 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-508379)

To Edwige ♡

List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.

I Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 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.

II Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do Neural Language Models Show Preferences for Syntactic Formalisms?. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4077–-4091 Online. Association for Computational Linguistics.

III Vinit Ravishankar, Artur Kulmizev, Mostafa Abdou, Anders Søgaard, and Joakim Nivre. 2021. Attention Can Reflect Syntactic Structure (If You Let It). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3031–-3045, Online. Association for Computational Linguistics.

IV Artur Kulmizev and Joakim Nivre. 2023. Investigating UD Treebanks via Dataset Difficulty Measures. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1076–1089, Dubrovnik, Croatia. Association for Computational Linguistics.

V Artur Kulmizev and Joakim Nivre. 2022. Schrödinger's tree – On syntax and neural language models. *Frontiers in Artificial Intelligence*, Volume 5.

Reprints were made with permission from the publishers.

Contents

Acknowledgements

I would like to thank the following people, without whom this work would have never seen the light of day.

Joakim — Thanks for everything, really. You've supported me throughout some of the toughest moments of the last five years, including a pandemic, a baby, and multiple paradigm shifts in the field. Your wisdom and guidance has been invaluable. You are one of the smartest people I've ever met and I'm lucky to have learned so much from you.

Anders — Thanks for always being available to chat and brainstorm ideas for projects. I'm sorry that several of them didn't work out, and that I couldn't arrange a visit to Copenhagen to collaborate more closely. I truly appreciated having you by my side.

Vinit and Mostafa — The triumvirate breaks with me, as we are blown asunder by the winds of fortune. Thanks for accompanying me on this odyssey through the NLP morass.

Anna and Maja — I am fortunate to have met you two in Sigtuna. Thanks for commiserating with me all these years.

Miryam — Thanks for welcoming me to Uppsala and helping me adjust to lingfil. I am glad that we can still meet up in Antwerp, of all places.

Fredrik — Thanks for all of your help with teaching. And procrastinating.

Daniel — Thanks for all of your wisdom and advice. I am glad we could be colleagues, if only for a short time.

CL group — Thanks for being wonderful colleagues and helping me with all sorts of matters.

Philipp, Erik, Harald, and all my lingfil colleagues — Thanks for the great conversations, about linguistics or otherwise. I learned a lot from you and will miss going to Williams on Fridays.

Walter and CLiPS — Thank you for welcoming me into your lab, even though I could only stay a short while. I am grateful to have met all of you.

Mom, Dad, Diana, Mark, Baba Toma and Nina — Although it sometimes hurts to be so far, thanks for supporting me at every stage of this strange European journey. I couldn't do this without you. Вы мое сердце.

Ed — You are the reason this dissertation could even begin to be written. You helped me through countless episodes of complacency and self-doubt, and now we've reached the finish line. We have grown so much together and been through more than what should be asked of any couple. It's now time to start a new journey — hopefully a smoother one than what we're used to. We'll do great things together.

Lev — I love you, little man. This dissertation will be your bedtime story for the foreseeable future. Da-da-da-da-da-da.

1. Introduction

Language models (LMs) — machine learning models that assign probabilities to strings in a language — have transformed the field of natural language processing in the last decade. Despite their simplistic learning objectives, LMs have been leveraged in various ways in order to push forward the state of the art across numerous tasks. Most of this success is owed to the deep learning paradigm, which allows for the end-to-end training of general-purpose algorithms without the need to incorporate strong theoretical assumptions (such as the Markov assumption in n-gram language modeling). To this end, sequence processing models such as the LSTM (Hochreiter and Schmidhuber, 1997) and Transformer (Vaswani et al., 2017) have showcased considerable utility, serving as automated feature extractors capable of encoding salient linguistic information without an explicit supervisory signal. Aided by rapid advancements in graphical processing hardware, this framework has been administered at increasing scale — both in terms of data and model parameter size — continually resulting in more expressive and accurate models.

Generally speaking, pre-trained language models can be leveraged in one of two ways. Primarily, they are employed for text generation, where the resulting output is often fluid, grammatical, and thematically coherent. In line with this, it is often possible to prompt models to generate answers to various NLP tasks, without requiring task-specific training data. However, this usually requires that the model is large enough, has been pre-trained on sufficient data, and that the prompt is correctly formed. Alternatively, language models can be fine-tuned so as to calibrate their parameters to the task in question. Doing so typically presupposes that the model has learned relevant linguistic features throughout pre-training, which could then be leveraged towards a specific task. In such cases, only the intermediate parameters are updated, while the generation layer is replaced with an appropriate mechanism befitting the downstream task, such as text classification, sequence tagging, or structured prediction.

Both of the aforementioned approaches (and numerous others) have yielded success across a wide array of NLP tasks, leading researchers to posit that LMs possess considerable linguistic knowledge. However, evaluating the extent to which this claim is true remains a central issue. Perhaps the most straightforward method for doing so is by measuring a model's performance on downstream tasks for which human baselines are available. This, however, presents potential caveats that can be obfuscated by traditional performance metrics, such as accuracy. For example, it is difficult to ascertain whether a fine-tuned question answering (QA) model demonstrably relies on the general linguistic

knowledge it acquired in the pre-training stage, or if it simply learns spurious correlations between input and output. Likewise to this end, one cannot draw parallels between the linguistic capabilities of humans and models without showing how the processing of the latter aligns with the former (or theories thereof).

In an attempt to address these concerns, researchers have turned to the topic of syntax — the area of linguistic study which concerns how words combine to form phrases or sentences. Syntax has long been a prominent subject in the field of linguistics, with decades of corresponding research, theory, and debate. Though many open questions remain, the study of syntax has nonetheless produced numerous theories and frameworks through which principled observations about natural language continue to be made. Applying these principles to computational models of language is not a straightforward endeavor, but one that has been earnestly pursued in an attempt to demystify the notoriously opaque language models that have become mainstream in NLP. Through the syntactic lens, researchers have been able to identify cases where models match expected human behavior, where they fail, and how well their internal syntactic processing aligns with theoretical expectations.

In this dissertation, I investigate the syntactic knowledge of language models through the perspective of another type of model: *dependency parsers*. Parsing carries a long tradition in NLP, with ample research devoted to developing efficient, state-of-the-art algorithms capable of analyzing text across many languages and domains. Unlike language models, which are self-supervised and trained on unlabeled data, parsers are structured prediction models, which are usually trained on labeled linguistic data in the form of treebanks. Such data typically adheres to a particular syntactic framework, where *dependency* has proven to be particularly popular due to its mathematical properties and practical usage. Given unlabeled text — for example, a single sentence — a parser is tasked with returning its corresponding labeled syntactic structure. As such, parsers can be imagined as idealized computational models of syntax in that they are designed for processing sentences according to the rules of a particular syntactic framework.

Studying language models through the perspective of treebanks and parsers can provide interesting insights about their syntactic capabilities. For example, given one's understanding of the inner workings of parsers, one might investigate the extent to which the language-internal components of language models, such as attention weights, behave similarly. Furthermore, one could evaluate the utility of language models' hidden state representations (learned end-to-end) when used as input in downstream supervised parsing (which traditionally requires hand-crafted features). In the case that such representations are indeed useful for parsing, one might choose to examine which aspects of syntactic structure are actually encoded therein. Ultimately, positive findings in these directions could provide support for the idea that language models pro-

cess their input hierarchically — much like a parser — despite being trained in a linear, left-to-right manner.

Unfortunately, not all positive findings in the above context can be interpreted as concrete proof that language models possess syntactic knowledge (or that their processing aligns with linguistic theories thereof). Indeed, when employing parsers as instruments for understanding language models, one needs to ascertain that the conclusions one draws apply directly to the latter — and that alternative explanations do not exist. This is especially salient when working with linguistic data like treebanks, which often vary in terms of text type, as well as the frequency at which various syntactic structures are attested. Failing to acknowledge such aspects runs the risk of misattributing models' performance to their general linguistic competence, rather than to more straightforward factors, such as the domain or genre of the underlying text. Linguistic typology is also a vital aspect to consider, as models that are considered to be linguistically competent should likewise be able to generalize equally across typologically diverse languages. In the case that they cannot, the claims that they possess syntactic knowledge become increasingly dubious. I attempt to integrate these concerns throughout the entirety of this dissertation, as they are vital to the pursuit of understanding language models.

1.1 Research Questions

This dissertation addresses several research questions that are relevant to understanding the syntactic capabilities of language models. I outline them as follows:

RQ1 How do representations from pre-trained language models affect the behavior of neural dependency parsers?

Addressing this question will aid us in understanding the benefit of contextualized representations for the task of dependency parsing, as well as their impact on the error profiles of different dependency parsing approaches.

RQ2 To what extent is syntax encoded in language model components, and how does this vary across different languages?

Addressing this question will ascertain whether or not methods proposed for extracting syntactic structure from English models can likewise generalize to models trained on other, typologically distant languages. Also, it will shed light on how syntactic structure is distributed across different model components, such as hidden state representations

or self-attention distributions.

RQ3 How is our understanding of language model components affected by experimental variables such as the choice of alternative syntactic frameworks or learning objectives?

Addressing this question will help us disentangle the various factors at play when investigating the syntactic knowledge of neural language models. In particular, it will highlight the role played by linguistic representations in probing studies. In addition, it will show the influence of the learning objective employed to train the model.

RQ4 How can we characterize treebank properties that affect parser learnability, such as annotation quality or tokenization decisions?

Addressing this question will provide insight into the various properties and idiosyncrasies inherent to treebanks, which can aid us in understanding how models interface with such data, and avoid building conclusions around non-linguistic confounds.

1.2 Structure of the Dissertation

This dissertation consists of seven chapters, which focus on answering the aforementioned research questions in the context of the included articles (see Section 1.3). The present chapter, Chapter 1, which introduces the research questions and the included articles, is followed by a background chapter, Chapter 2, which introduces all prerequisite concepts and terminology relevant to the rest of the dissertation. The remaining five chapters are organized as follows, detailing the work of me and my collaborators.

In Chapter 3, we address RQ1 by investigating the performance of neural dependency parsers which have been enhanced with contextualized word representations. We find that representations based on ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) yield noticeable accuracy improvements across 13 typologically diverse Universal Dependencies treebanks (Nivre et al., 2020) when compared to static embeddings. This performance boost applies to both transition-based and graph-based parsers, though the latter benefit slightly more. In investigating the parsers' error profiles, we find that they both resemble each other, especially when infused with contextualized representations. This finding is at odds with previous research, which demonstrates that non-neural transition-based and graph-based parsers possess complementary benefits and drawbacks. We take this to mean that contextualized representations extracted from ELMo and BERT are expressive enough to overcome the limitations of the respective parsing paradigms.

Having demonstrated the benefit of language model representations in supervised dependency parsing, I dedicate Chapter 4 to exploring RQ2 in order to understand how exactly syntactic information is encoded in such models. First, we employ a structural probe (Hewitt and Manning, 2019) in order to decode dependency structure from language models' hidden state representations. We find that, for both BERT and ELMo, such information is indeed extractable across the 13 treebanks mentioned above, with BERT's middle layers returning the best accuracy. Second, we investigate the extent to which dependency syntax is reflected in the distributions of multilingual BERT's 144 attention heads. We observe considerable variation across 19 languages, with some out-performing a naive, adjacent-branching baseline and others faring considerably worse. In investigating specific dependency relations, we find that various heads specialize to capture such relations across languages, implying that some linguistic knowledge is encoded in a multilingual fashion.

In Chapter 5, I address RQ3 in order to shed a nuanced light on the mostly positive findings related to RQ2. First, I revisit the probing results in the context of an alternative dependency framework (Surface-syntactic Universal Dependencies, or SUD (Gerdes et al., 2018)), which, unlike UD, emphasizes function word heads. In probing for SUD trees, we encounter significant variation in decoding accuracy compared to UD, with some treebanks strongly preferring the latter, others the former, and some returning no preference at all. We demonstrate that such variation in results is not primarily due to the syntax encoded by the surveyed models, but rather the graph properties and annotation of the treebanks on which our probes were trained. Second, we contrast the aforementioned attention distributions with those obtained after fine-tuning our models directly for dependency parsing. We find that we can decode dependency trees far more reliably in the latter case, indicating that attention can reflect syntactic structure if a model's training objective entails learning such structures.

In light of the findings related to RQ3, which emphasize the methodological confounds inherent to the search for syntax in language model components, Chapter 6 investigates whether or not the *learnability* of treebanks can be reliably measured (RQ4). To do so, we employ a popular neural graph-based parser, which we train on 88 Universal Dependencies treebanks. We experiment with three methods for estimating dataset difficulty, which are broadly based on model training dynamics and information theory. Ultimately, we find that each metric reveals various idiosyncratic properties of treebanks, such as lexical and structural variety, morphological complexity, domain/genre, and annotation choices. We observe that such factors are typically obfuscated by accuracy-based metrics like labeled attachment score (LAS), which should inspire caution when employing treebanks in probing model representations for syntactic information.

Chapter 7, the conclusion, attempts to synergize the findings of the aforementioned studies in the context of a broader research agenda devoted to study-

ing the syntactic knowledge of language models. We begin by outlining (what we perceive to be) the three dominant experimental paradigms typically employed as a means of this pursuit. We then identify considerations central to this body of work, which we argue can aid in the formulation of more precise research questions and add nuance to the study of language models. This is followed a future work section which contextualizes our work within recent developments in the field and offers directions for future research.

1.3 Papers and Contributions

This dissertation is based on the following papers, which are referred to in the text by their Roman numerals. Papers I through IV each address a specific research question, while Paper V consists of a conceptual analysis that is described in Chapter 7. I also provide a description of my contributions to each paper.

I Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 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.

The parsing experiments described in this article directly address RQ1. I was responsible for implementing and debugging the contextual embedding code, running all experiments, plotting, analysis, and writing.

II Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do Neural Language Models Show Preferences for Syntactic Formalisms? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4077–4091, Online. Association for Computational Linguistics.

This article partially addresses RQ2 and RQ3. Specifically, the UD probing experiments are discussed with respect to RQ2, whereas the SUD experiments relate to RQ3. I was responsible for refactoring John Hewitt's probing code base in order to comply with UD and multilingual BERT, running all experiments, plotting, analysis, and writing.

III Vinit Ravishankar, Artur Kulmizev, Mostafa Abdou, Anders Søgaard, and Joakim Nivre. 2021. Attention Can Reflect Syntactic Structure (If

You Let It). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3031–3045, Online. Association for Computational Linguistics.

Similarly to Paper II, this article also partially addresses RQ2 and RQ3. The attention decoding experiments concerning the pre-trained multilingual BERT model refer to RQ2, while the fine-tuning experiments relate to RQ3. I was responsible for implementing the attention decoding codebase, running all decoding experiments described in Section 4, plotting, and writing.

IV Artur Kulmizev, Joakim Nivre. 2023. Investigating UD Treebanks via Dataset Difficulty Measures. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1076–1089, Dubrovnik, Croatia. Association for Computational Linguistics.

The three metrics surveyed in this article — Dataset Cartography, $\mathcal{V}$-information, and Minimum Description Length — each together address RQ4. I was responsible for the implementation of each metric, running all experiments, plotting, analysis, and writing.

V Artur Kulmizev and Joakim Nivre. 2022. Schrödinger's tree – On syntax and neural language models. In *Frontiers in Artificial Intelligence, Volume 5*.

This article does not pertain to any individual research question. Rather, it serves as a synthesis of the various threads discussed throughout this dissertation, placed in the context of the broader research agenda concerned with investigating the syntactic knowledge of neural language models. I was responsible for defining the main points of analysis, performing the literature review, and writing.

2. Background

In this chapter, I review the technical background that underpins the studies collected in this dissertation. I begin by offering a high-level introduction to the concept of syntax — an area of linguistic study concerned with sentence structure. Next, I describe Universal Dependencies — a popular annotation framework based on dependency syntax — as well as an alternative framework proposed to complement it. Finally, I provide an overview of the NLP tasks of dependency parsing and language modeling, including classical and contemporary approaches thereof.

2.1 Syntax

Syntax is an area of linguistic study that is concerned with how words combine to form more complex expressions like phrases and sentences. In general, syntax can be distinguished from *morphology* (the internal structure of words), as well as *semantics* (the meaning of sentences). Put differently, it concerns the *form* sentences take when all words have been properly arranged and can be interpreted in relation to each other. Such an arrangement is privy to a set of rules that determine how words can combine to convey meaning — that is, a *grammar*. All of the world's languages are shaped by their own grammars, each of which contain unique rules, restrictions, and idiosyncrasies. As an example of the above, consider the following English sentence:

(1) those dogs from across the street chase our cat everyday

Per the English grammar, one can interpret the noun phrase (NP) *those dogs* as the subject of the sentence, since it appears before the main verb *chases*; in other words, the dogs are doing the chasing. Naturally, swapping the position of *those dogs* with *our cat* (and changing the form of the verb) elicits an entirely different interpretation: one where the cat is chasing the dogs. In a similar vein, subject-verb agreement dictates that the verb *chase* is inflected to reflect the plural number of its actual subject (*dogs*), rather than the singular noun *street* that intervenes between them. Such phenomena, among many others, are often called *coding properties*. Appropriately, their function lies in encoding the various types of grammatical relationships that can be observed between words and phrases.

A complex interplay between coding properties is a product of the grammatical structure of a given sentence. It is important to note two aspects inherent to this structure, however. First, it is implicitly hierarchical: words combine to form phrases, which themselves combine to form sentences. Indeed, the subject-verb agreement highlighted in the example above indicates the combination of two distinct phrases — the subject NP (*those dogs from across the street*), governed by the plural noun *dogs*, and the VP (*chase our cat everyday*), governed by the present-tense, 3rd-person plural verb *chase*.

Second, such structures are not directly observable: the exact nature of the aforementioned hierarchy is privy to interpretation, and many frameworks and theories have been proposed as a means of describing it. Nevertheless, a sentence's grammatical structure can, to an extent, be illustrated by means of various transformations. For example, *those dogs from across the street* can safely be substituted with the third-person plural pronoun *they*, indicating that these words constitute a well-formed grammatical unit. In contrast, the same cannot be said for *those dogs from across*, which, if substituted, would violate syntactic boundaries and render the sentence ungrammatical. In the following section, I will briefly outline two major means of representation that seek to formalize the notion of grammatical structure. Our focus, however, will be placed on *dependency grammar*.

2.1.1 Constituency Grammar

Constituency grammar is a popular syntactic framework among theoretical linguists, formalized by Chomsky (1956). At its most basic level, constituency grammar is motivated by the aforementioned observation: that groups of words (constituents) can behave as a single unit within a hierarchical structure. A constituent can therefore refer to a single lexical item, or, alternatively, a phrase comprising multiple words, yet headed by a lexical item. Typically, the part of speech of a phrase's head will indicate the grammatical category it represents, such as an NP in the case of *those dogs from across the street*. Such properties are typically specified by means of a Context Free Grammar (CFG) — a formal system designed for the description of languages' constituency structure. Figure 2.1 depicts a constituency tree for Sentence 1, following a simple English-language CFG.

2.1.2 Dependency Grammar

Like constituency grammar, dependency grammar holds a long tradition in the study of syntax (Tesnière, 1959; Mel'cuk et al., 1988). However, unlike constituency, dependency grammar's distinguishing feature is its emphasis on the functional relations (dependencies) between individual words, rather than phrase structure. Such relations are defined to be binary and asymmetrical,

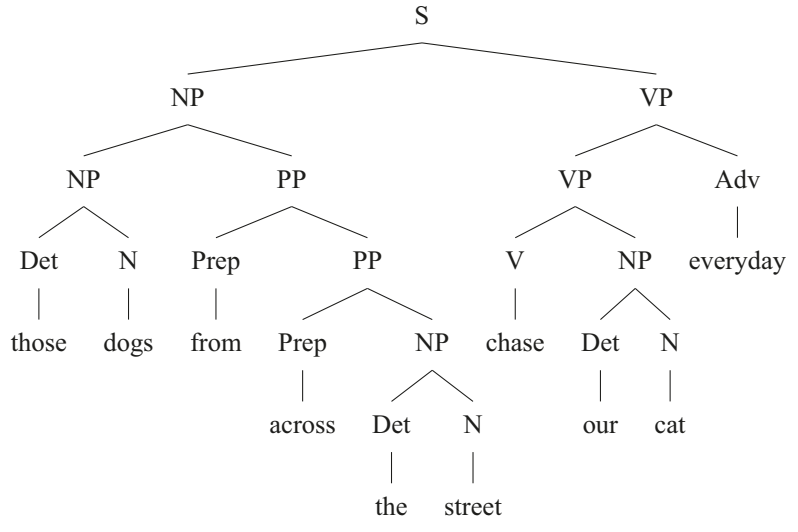

Figure 2.1. A basic constituency tree for the sentence *those dogs from across the street chase our cat everyday*.

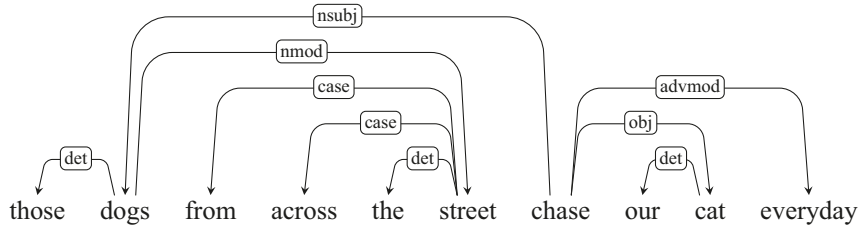

Figure 2.2. A basic dependency tree for the sentence *those dogs from across the street chase our cat everyday*, following the Universal Dependencies annotation scheme.

relating a *dependent* to its *head*. For example, the word *dogs* from Sentence 1 can be analyzed as a dependent of the verb *chase*, via a nominal subject (`nsubj`) relation. The set of dependencies occurring in a sentence can thus be represented as a tree, which is typically rooted at the main verb. Figure 2.2 depicts a dependency tree for Sentence 1.

Dependency grammar carries numerous advantages as a syntactic representation. Chief among these is its flexibility with regard to word order, since dependencies are drawn directly between words on a relation–type basis. This is particularly useful when analyzing morphologically rich languages with flexible word order, such as Russian. Figure 2.3 demonstrates the adaptability of the dependency representation to a sentence in Russian, where the same functional relations persist despite the words having been rearranged in various ways. This is non-trivial for a constituency framework, which would require

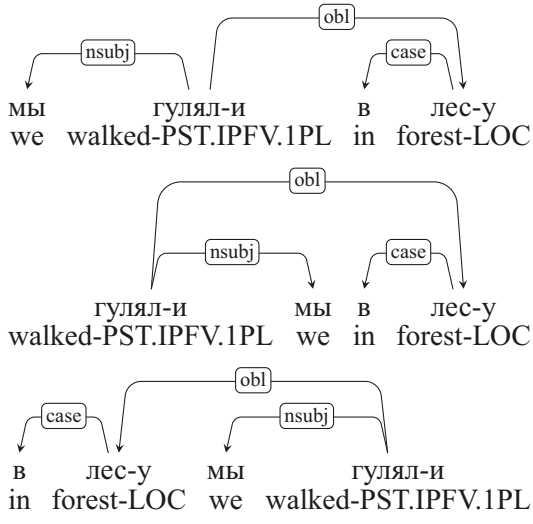

Figure 2.3. Three plausible orderings of the sentence meaning *we walked in the forest* in Russian.

the design of separate productions for every plausible word order or rely on mechanisms like traces.

Another advantage of dependency-based representations is that they can, in many cases, approximate the semantic relationship between predicates and their arguments. For example, given a dependency parse of Sentence 1, one can directly extract the root verb *chase* as the predicate, and *dogs* and *cat* as its core arguments. For this reason, dependency has proven to be especially popular in the fields of natural language processing, where many applications, such as question answering (Simmons et al., 1964; Brill et al., 2001; Lin et al., 2002; Cui et al., 2005; Wang et al., 2012), traditionally rely on such information being available. Though semantic structure can likewise be extracted from a constituency tree, it is not directly accessible like it is in dependencies and special algorithms are required in order to retrieve it.

A contentious aspect of dependency grammar is the notion of *headedness* (Zwicky, 1985; Hudson, 1987). Indeed, while some relations are straightforward to orient according to a variety of criteria (for example, distributional equivalence or morphological locus (Hudson, 1987)), others are more contentious and are therefore subject to theoretical preference. For example, the chief difference between the popular Universal Dependencies (UD) framework (Nivre et al., 2020) and its proposed complement, Surface-Syntactic Universal Dependencies (SUD) (Gerdes et al., 2018) (discussed in detail in Section 2.2), is that the former utilizes content word heads, while the latter prioritizes function words. In practice, this amounts to SUD treating lexical verbs as dependents of auxiliaries rather than the other way around. Another problematic construction is coordination, where two or more similar elements are connected

by a coordinating conjunction, for example: *the old men and women*. Here, the framework of choice must determine whether a conjunct or a coordinating conjunction serves as the head, as well as how all other element relate to it — justifying the interpretation of the ambiguous modifier *old*. This itself raises the issue of cases in which no clear head is present, such as multiword expressions or named entities, such as *New York*. In the following section, I will discuss these issues in detail as they relate to Universal Dependencies.

2.2 Universal Dependencies

Universal Dependencies (UD) (Nivre et al., 2016, 2020; de Marneffe et al., 2021) is a relevant NLP initiative focused on the development of dependency treebanks. It offers a unified, cross-linguistically consistent annotation framework aimed at facilitating the development of multilingual NLP systems and advancing typological research. Since its inception, UD has become widely adopted, growing from 10 treebanks across 10 languages in v1.0 to 243 treebanks across 138 languages in v2.11.

One of UD's core design principles is a dedication to *lexicalism*: a perspective on syntax which posits that (syntactic) relations are formed directly between words. As is common in dependency frameworks, this entails that each word in a sentence is assigned 1) another word as its syntactic head and 2) a label describing the functional relation between the two. Words are defined in the *syntactic* sense (as opposed to orthographic or phonological), meaning that contractions are undone (don't → do, n't) and clitics are separated from their hosts (Spanish: "give it to me": dámelo → da, me, lo). Although this approach does not take morphological structure explicitly into account, UD nonetheless provides such information in the form of lemmas, part-of-speech tags, and morphological features for each word. Additionally, UD proposes a set of 37 universal relations, which can be expanded as needed on a per-language basis. These relations are taxonomized vis-a-vis a dependent's *structural* category (nominal, clause, modifier word, function word) and the *functional* relation it bears to its head (core argument of clausal predicates, non-core dependent of clausal predicates, and dependent of nominals). UD also specifies additional relation classes that do not fit in this dichotomy, which pertain to coordination, multi-word expressions, and other not easily specified phenomena.

Another core principle is UD's prioritization of direct relations between content words. Such an approach is motivated by a need to highlight shared grammatical relations across languages, despite varying usages of functional elements therein, such as adpositions or case-markers. This is illustrated in Figure 2.4, which depicts the UD annotation for a parallel sentence in English and Russian. At first glance, one can observe the relations which the two languages have in common: namely, core arguments and oblique modifiers. However, the means by which these constructions are functionally expressed differ dras-

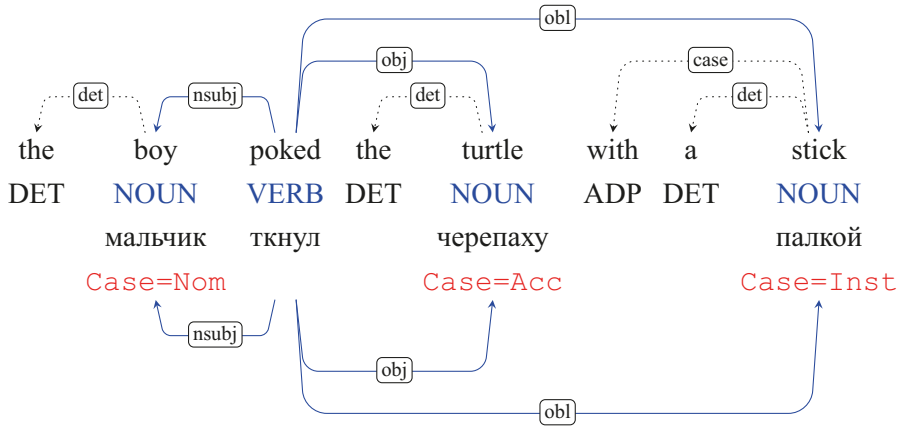

Figure 2.4. A parallel sentence in English (above) and Russian (below), as annotated in Universal Dependencies. Content relations are shown in blue, and functional relations are dotted.

tically from one another. For example, one can see that English makes use of determiners in order to indicate definiteness, while Russian does not mark such properties at all. Similarly, while English encodes the oblique modifier *with a stick* via the preposition *with*, Russian marks it directly on the noun by means of the instrumentative case suffix *-oй*. As such, UD's emphasis on content words is able to shed light on how functional relations are encoded across languages. This representational choice carries an additional advantage in that it can easily be employed in downstream natural language processing tasks, which are typically concerned with relations between content words.

Despite is widespread adoption and utility, UD has been met with various strands of criticism relating to its faithfulness as a syntactic annotation scheme. One such criticism is put forth by Osborne and Gerdes (2019), who argue that UD's subordination of function words effectively make it a semantics-oriented framework. To illustrate their point, Osborne and Gerdes (2019) identify a number of constructions, which, in their view, UD does not annotate in a representative syntactic fashion. Gerdes et al. (2018) operationalize these concerns as part of an alternative annotation framework named Surface-Syntactic Universal Dependencies (SUD), which prioritizes "purely syntactic criteria". SUD is designed to be near-isomorphic to UD, whereby existing UD treebanks can be deterministically converted to an SUD representation. Figure 2.5 depicts a sentence which yields contrasting representations between UD and SUD. I make note of three phenomena therein (auxiliary verbs, adpositions, coordination), which are discussed below.

Auxiliary Verbs

In Figure 2.5, one observes that UD places the main verb *studying* (in the progressive aspect) at the root of the sentence, while SUD selects the auxil-

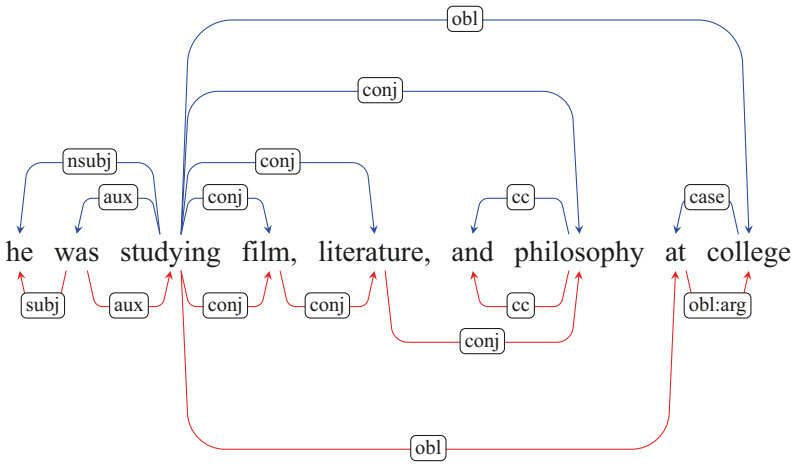

Figure 2.5. A sentence annotated in UD (top, blue) and SUD (bottom, red).

iary *was*. UD classifies auxiliary verbs as *non-core dependent function words*, which serve to specify various verbal features (such as tense, aspect, or modality) and cross-reference other arguments via agreement or other means. The main verb, on the other hand, is interpreted as dictating the underlying argument structure and is therefore prioritized as the head of all such dependencies. In contrast, SUD relies on distributional criteria to determine the relationship between main verbs and their auxiliaries. For example, since *was studying* and *studying* can be considered to have varying distributions, the latter cannot readily be slot into the existing structure: **he studying*. This analysis can also be interpreted through the lens of *subcategorization* (Osborne and Gerdes, 2019), which posits that, given words A and B, A subcategorizes for B if it determines the category B will take. In the case of Figure 2.5, the presence of *was* restricts *study* to the progressive *studying* (as opposed to the present *studies*, for example) and therefore subcategorizes for it.

Adpositions

Another key distinction between UD and SUD is the treatment of adpositions. As seen in Figure 2.5, UD draws a relation between *studying* and the nominal *college*, to which the locative preposition *at* bears a case relation. Such an analysis is adopted so as to promote cross-lingual parallelism, given the various means by which languages tend to encode grammatical functions (recall the discussion of Figure 2.4). Per UD's view, elements like *at college* constitute a referential core (*college*) and its corresponding functional markers (*at*) — reminiscent of Tesnière's notion of dissociated nuclei (Tesnière, 1959). SUD, on the other hand, again relies on distributional criteria to show that the oblique phrase *at college* is distinct from the nominal *college*.

Coordination

Coordination is a notoriously difficult phenomenon to represent via dependency formalisms. This is due to the fact that coordination composes elements belonging to a particular syntactic category (such as nominals or clauses) to create a larger element of the same type. Indeed, this is at odds with the asymmetric notion of dependency, which calls for certain elements to be prioritized as heads. With this restriction in mind, UD opts for a *bouquet*-style analysis, which assigns the first coordinated element (conjunct) as the head and each subsequent element as its direct dependent (see Figure 2.5, top). If present, the coordinating conjunction (*and*, *or*, *but*), is assigned as a dependent of the final coordinated element. SUD adopts an alternative, *chain*-style approach, where conjuncts appear as direct dependents of their closest neighbor conjuncts (see Figure 2.5, bottom). This results in deeper trees, which can serve to express the increased cognitive load associated with processing embedded structures like coordination (Liu, 2008; Futrell et al., 2015).

2.3 Dependency Parsing

Dependency parsing refers to the task of automatically analyzing sentences according to their underlying dependency structure. Formally, this amounts to mapping a sentence $S = (w_0, w_1, w_2, ..., w_n)$ of length n to its respective syntactic structure $\mathbf{y}$. An artificial root token $w_0 = \texttt{ROOT}$ is typically prepended to S and does not modify any other token in the sentence. In UD and SUD, $\mathbf{y}$ is a tree (a rooted, directed, acyclic graph) over S, where an individual arc $(w_i, r, w_j) \in \mathbf{y}$ represents a dependency relation from head w_i to dependent w_j, labeled with relation type r. Naturally, the root of the sentence (typically, the main verb) has no direct heads, and must be therefore be subordinated by the dummy root token w_0.

Unlike classification tasks, which entail predicting a single class y given a fixed label set $\mathcal{R}$, parsing is considered a structured prediction task, where the output space is constrained by the sentence length $n + 1$ (with the `root` symbol included). In data-driven parsing (the paradigm discussed in this thesis), a parser is typically a parameterized function f whose parameters θ are fit on some gold-annotated training set X_{train}, such as a UD treebank. A trained parser's predictions $\hat{\mathbf{y}} = f_\theta(\mathbf{x})$ can then be evaluated on a held out test set X_{test} by means of various metrics. Here (and throughout the remainder of this thesis), $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ refers to the numerical encoding of a sequence of tokens (either via one-hot or embedding lookup). A brief description of three popular dependency parsing evaluation metrics is provided below:

- **Unlabeled Attachment Score** (UAS): the percentage of all words in X_{test} that are assigned the correct head by f, irrespective of label.

- **Unlabeled, Undirected Attachment Score** (UUAS): the percentage of all relations in X_{test} that are correctly assigned as constituting a dependency, irrespective of the label or directionality of the relation.
- **Labeled Attachment Score** (LAS): the percentage of all words in X_{test} that are assigned the correct head by f, including the label.

2.3.1 Parsing Algorithms

Dependency parsing algorithms generally fall under one of two paradigms: transition-based and graph-based. Both approaches differ in their general properties, as well as the data structures they employ as abstractions of the parsing process.

Transition-based Parsing

Transition-based dependency parsing (Kudo and Matsumoto, 2002; Nivre, 2003; Yamada and Matsumoto, 2003; Nivre et al., 2004; Nivre and Nilsson, 2005; Attardi, 2006; Attardi and Ciaramita, 2007; Titov and Henderson, 2007a,b) makes use of data structures such as a stack σ and buffer β in order to incrementally parse an input sentence S into a set of arcs E. In arc-standard parsing (a basic transition-based parsing algorithm) (Yamada and Matsumoto, 2003; Nivre, 2004), the goal is to move from an initial configuration c_0 (where the buffer contains all words in a sentence and the stack contains the special `root` symbol) to a terminal configuration c_m (where the buffer is empty and the stack contains only the dummy root token). Such parsers move between configurations by means of *transitions* — partial functions from one configuration to the next. For an arc-standard parser, the set of defined transitions $\mathcal{T}$ is as follows:

- Left-Reduce: Add a left-directed arc (w_j, r, w_i) to E, where w_i and w_j are the two topmost words on the stack, respectively. Remove w_i from stack.
- Right-Reduce: Add a right-directed arc (w_i, r, w_j) to E, where w_i and w_j are the two topmost words on the stack, respectively. Remove w_j from stack.
- Shift: Push the first word w_i in the buffer onto the stack.

Given this formulation, the set of arcs E at c_m is guaranteed to yield a *projective* dependency tree, wherein the projection of each head word forms a continuous substring of the sentence[1]. This property contributes to a favorable parsing complexity, but limits the set of syntactic constructions that can be adequately represented (Kuhlmann and Nivre, 2006)

[1] In the case that the `root` token is positioned at the beginning of the sentence, projectivity simply indicates that the tree contains no crossing arcs.

An *oracle* o is required in order to process a given configuration c and deterministically output an optimal transition $\hat{t}$. The oracle is typically composed of two components: a feature function ϕ and a classifier f with parameters θ. In pre-neural parsing approaches, ϕ is most often implemented as a feature template, which takes selected elements of the configuration c as input (features) and returns a vector representation thereof. Such features may specify the form, lemma, or part of speech of the most recently processed word on top of the stack, or consider additional information extracted from words in the buffer (Nivre et al., 2006; McDonald and Nivre, 2007, 2011). A feature representation $\phi(c)$ is used to fit a classifier f_θ, which scores all $t \in \mathcal{T}$ and returns an optimal transition: $\hat{t} = f_\theta(\phi(c))$. The returned $\hat{t}$ is then applied to c, which returns a new c until the terminal configuration c_m is reached. In this setup, f often takes the form of a linear machine learning classifier, such as logistic regression or support vector machine (Yamada and Matsumoto, 2003; Nivre et al., 2006).

Graph-based Parsing

Graph-based methods (Eisner, 1996; McDonald et al., 2005a,b; McDonald, 2006; Koo et al., 2007; McDonald and Pereira, 2006; Smith and Smith, 2007) are another popular class of dependency parsing algorithms. The main assumption behind graph-based parsing is that dependency structures can be represented as *graphs* $G = (V, E)$, with individual tokens appearing as nodes V that are connected via a set of arcs E. The probability that any two words are connected by a dependency relation is formalized as a score attributed to the arc connecting their representative nodes, which is calculated by means of a parameterized function f_θ. Given a fully connected weighted graph of this nature, the goal of graph-based parsing is to find a *maximum spanning tree* (MST) that yields the highest total score across arcs. In this context, a spanning tree refers to any directed tree over G that spans all original nodes V. Such approaches are called *arc-factored* (Eisner, 1996; McDonald et al., 2005a), given that the score attributed to a tree $\mathbf{y}$ is factored directly through the arcs:

$$\text{score}(\mathbf{y}) = \sum_{(w_i, r, w_j) \in E} f_\theta(\phi(w_i, r, w_j)) \tag{2.1}$$

Given a fully connected weighted G, the Chu-Liu/Edmonds (CLE) algorithm (Chu, 1965; Edmonds et al., 1967) is applied over G in order to yield the highest scoring tree $\hat{\mathbf{y}}$ over V. This output $\hat{\mathbf{y}}$ satisfies all necessary formal tree constraints — in other words, that it spans all nodes in V, is rooted at the dummy node `root`, and is acyclic. Unlike the aforementioned transition-based algorithm, CLE imposes no constraint on projectivity, although projective graph-based parsing alternatives based on chart-parsing do exist (Eisner, 2000; McDonald and Pereira, 2006).

Similarly to their transition-based counterparts, graph-based parsers rely on a feature function ϕ and a training algorithm f_θ in order to learn arcs scores from data. As with transition-based parsers, ϕ is typically a feature template that highlights various properties of the arc (w_i, r, w_j), such as the identity or part of speech of w_i and w_j, the relation r, or properties of preceding, intervening or following words: for example, w_{i-1} or w_{j+1}. For training, standard graph-based parsers like MSTParser (McDonald et al., 2005b) employ online algorithms in order to tune the parameters θ with respect to each instance in the training data, incrementing and decrementing weights for correct and incorrect solutions, respectively.

Similarities and Differences

Transition-based and graph-based dependency parsers — the classical formulations of which are described above — have historically fared similarly in terms of overall accuracy (Buchholz and Marsi, 2006). However, McDonald and Nivre (2007, 2011) show that the contrast in the design of each algorithm carries considerable trade-offs relating to feature representation, training, inference, and error profile.

For example, transition-based parsing is typically characterized as *local* and *greedy*. In this context, *local* refers to the training objective, where a classifier $f_\theta(\phi(c))$ is tasked with scoring a single transition t, as opposed to calculating probabilities of entire transition sequences. This locality aspect enables the integration of rich feature spaces, wherein the entire parsing history up to timestep $i-1$ can be consolidated. Likewise, the inference process is referred to as *greedy*, given that the parser makes a series of locally optimal decisions in order to produce a globally optimal parse. Such a setup is advantageous in that the worst-case running time for an arc-standard parser is bounded by the maximum number of transitions in a sequence. This yields an efficient $O(n)$ time-complexity for inference[2]. A key disadvantage of this approach, however, is that it is sensitive to error propagation stemming from mistakes made early in the parsing process. Relatedly, transition-based systems like arc-standard are restricted to only be capable of producing *projective* dependency trees. Faced with non-projective structures (Kuhlmann and Nivre, 2006), which, though rare, represent a subset of linguistic phenomena, such systems must be supplemented with post-processing or additional transitions (Nivre, 2007, 2009; Kuhlmann and Nivre, 2010; Gómez-Rodríguez and Fernández-González, 2015).

In contrast to transition-based parsing, graph-based algorithms are generally described as being *global* and *exhaustive* in nature (McDonald and Nivre, 2007). The global aspect stems from the parser's training objective: optimizing the score of the entire (correct) dependency graph, as opposed to individual arcs. An advantage of this approach is that it is considerably less prone to er-

[2] Assuming that oracle and transition functions can be computed in constant time.

ror propagation, as arcs are typically misclassified in isolation — irrespective of the correctness of their encompassing subgraph(s). In a similar sense, the search for the highest scoring tree is *exhaustive* in that the spanning tree algorithm must consider the scores of all incoming arcs for each node in graph. The end result is a well-formed spanning tree that is not privy to the projectivity constraints of transition-based parsers. However, this carries a price of $O(n^2)$ complexity at inference time, making it slower than the greedy algorithm described above, and limiting the scope of features that can be leveraged for training.

The trade-off between local and global training as well as greedy and exhaustive inference has been shown to affect parser performance in various ways (McDonald and Nivre, 2007, 2011). For example, the tendency of transition-based parsers to propagate errors leads them to perform worse on longer sentences, longer dependencies, and arcs higher in the graphs. At the same time, their rich structural features enhance performance for shorter sentences and dependency lengths, as well as frequently occurring relation types, such as subjects and objects. The performance of graph-based parsers across various error categories is more evenly distributed, which is expected given their global nature.

2.3.2 Neural Dependency Parsing

Unlike the aforementioned approaches, which typically employ linear classifiers for prediction, contemporary dependency parsing largely relies on neural networks as the dominant modeling paradigm. This direction was first explored by Titov and Henderson (2007b) and Attardi et al. (2009), though neural networks only gained widespread prominence following the work of Chen and Manning (2014), Dyer et al. (2015), Kiperwasser and Goldberg (2016), Dozat and Manning (2017), and Kondratyuk and Straka (2019), *inter alia*. In this section, I provide an overview of three essential model components that have come to define neural dependency parsing: embeddings, feature extractors, and scoring modules. This is largely based on the work of Kiperwasser and Goldberg (2016), Dozat and Manning (2017), and Kondratyuk and Straka (2019), as variants of their models are employed in this thesis.

Embeddings

Embeddings are typically defined as dense, distributed vector representations of model input. In the context of neural dependency parsing, "input" usually refers to words, but can also encompass orthographic characters and part-of-speech tags. An important facet of embeddings is that they are *fixed*: a single, static vector is reserved per word type. In practice, this entails that words are represented identically regardless of the context in which they appear: for example, *light blue* and *turn on the light* assign the same vector for *light*.

In a sense, embeddings serve a similar role to the aforementioned feature vectors, in that they function as numerical representations of discrete input — for example, tokenized text. However, unlike the former, which requires manual selection of informative features, embeddings automatically encode distributional information in coordination with the training regime. Accordingly, they can be randomly initialized at the start of training, or imported as pre-trained vectors extracted from another model. Language modeling is a popular pre-training task in the latter case, as it captures a rich lexico-distributional signal from large-scale, un-annotated corpora, which can be leveraged downstream for a variety of tasks (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017).

With regard to parsing, Kiperwasser and Goldberg (2016) demonstrate that a concatenation of word and part-of-speech embeddings yields strong parsing performance. Put formally, an embedded representation for a given word $w_i \in S$ is computed as follows in their approach:

$$x_i = e(w_i) \oplus p(w_i) \tag{2.2}$$

where $e(w_i) \in \mathbb{R}^{100}$ and $p(w_i) \in \mathbb{R}^{25}$ are embedding lookup functions for words and part-of-speech tags, respectively, and $\oplus$ is the concatenation operator. In their case, word embeddings are initialized with pre-trained vectors and the part-of-speech embeddings are extracted from predicted tags. This input representation method is widely considered to be a strong baseline for neural dependency parsing, although it is occasionally supplemented with character embeddings, which have been shown to improve performance for morphologically rich languages (Dozat et al., 2017).

Feature Extraction

As discussed in Section 2.3.1, classical parsing architectures typically include a feature function ϕ that encodes selected properties of the input deemed useful for parsing. This is typically implemented in the form of a feature template, wherein the developer specifies all relevant features manually. In neural dependency parsing, feature extraction is partially accomplished by the embedding mechanism e, which works in conjunction with a separate encoding module ϕ (sometimes called a *featurizer*) to capture information automatically. In the case of Kiperwasser and Goldberg (2016) and Dozat and Manning (2017), this module is a long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) — a type of recurrent neural network (Elman, 1990) designed to mitigate the vanishing gradient problem (Pascanu et al., 2013).

LSTMs, like RNNs in general, are designed to capture information about a sequence of elements $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ (for example, word embeddings), where n is the number of elements in the sequence. An LSTM cell takes an element x_i as input and passes it through a series of gates, which serve to control flow of information across timesteps. The LSTM output vector h_n

can be interpreted as a *contextualized summary* of the entire input sequence $x_{1:n}$. Importantly, an incremental state h_i can likewise be extracted to serve as a contextual representation of input token x_i, conditioned on the preceding input $x_{1:i-1}$. This property also allows LSTMs to be *stacked* into k layers, where the output of LSTM at layer l can be input to another LSTM $l+1$, as in: $h^{l+1}_{1:n} = \texttt{LSTM}(h^{l}_{1:n})$.

Both Kiperwasser and Goldberg (2016) and Dozat and Manning (2017) employ LSTMs as the primary feature extraction mechanism for their parsers. However, the LSTM variant described there is *bidirectional*, which entails the use of two separate LSTM networks:

$$\texttt{BiLSTM}(x_{1:n}, i) = \texttt{LSTM}_F(x_{1:i}) \oplus \texttt{LSTM}_R(x_{n:i}) \tag{2.3}$$

Here, the term $\texttt{LSTM}_F(x_{1:i})$ refers to the output of a standard forward LSTM, which processes the input left-to-right and conditions on the history $x_{1:i-1}$. Additionally, $\texttt{LSTM}_R(x_{n:i})$ is the output of a *reverse* LSTM, which is conditioned on the *right-to-left* context $x_{n:i}$, starting with the final token. The hidden state representation h_i is thus the concatenation of each LSTM's outputs. Both Kiperwasser and Goldberg (2016) and Dozat and Manning (2017) report accuracy gains when utilizing BiLSTMs over their standard counterparts, due to the fact that incremental representations h_i encompass a full view of the sequence $\mathbf{x}$.

Unlike embeddings, which are typically initialized via pre-trained vectors, LSTMs are often randomly initialized and trained from scratch (along with the rest of the model's parameters). This has the effect of specializing their representations for the parsing task at hand, but can be insufficient for learning adequate features if the training treebank is too small. More recent approaches have sought to mitigate this issue by leveraging large, pre-trained language models as the feature extraction components of the parsing architecture (Kondratyuk and Straka, 2019; Üstün et al., 2020). In such systems, the embeddings and LSTM (or equivalent module) are fully replaced by the language model network (excluding the output layer), the parameters of which are then fine-tuned with respect to the parsing objective (Glavaš and Vulić, 2021).

Scoring Modules

Similarly to classical parsers, neural models feature a scoring module f. This module accepts contextualized features $\phi(h_{1:n})$ as input, which is typically the output of the last layer of a stacked BiLSTM or Transformer (Vaswani et al., 2017). In turn, the output of f is either a series of predicted transitions (in the case of transition-based parsing) or a score matrix over the entire sentence graph (graph-based parsing), from which a predicted parse $\hat{\mathbf{y}}$ can be obtained. This scoring module is typically trained to minimize a loss function $\mathcal{L}$ (such as cross-entropy), which is calculated with respect to the correct tree $\mathbf{y}$ — or, in the case of transition-based parsing, a sequence of valid transitions leading

to $\mathbf{y}$. The loss is then propagated back through the network and all parameters — including embeddings and feature extractor — are updated jointly.

Kiperwasser and Goldberg (2016) propose two different scoring modules. One of these is a transition-based algorithm, which employs a multi-layer perceptron (MLP) as a scoring function f. As mentioned in Section 2.3.1, f accepts a configuration c as input and returns the highest scoring transition via:

$$\hat{t} = \arg\max_{t \in T} f(\phi(c))$$
$$\phi(c) = h_{\sigma_2} \oplus h_{\sigma_1} \oplus h_{\sigma_0} \oplus h_{\beta_0}$$

Here, c is represented as the concatenation of the feature vectors corresponding to the top three items on the stack and the first item on the buffer. Though Kiperwasser and Goldberg (2016) experiment with using an extended feature set — encompassing the modifiers of $\sigma_2, \sigma_1, \sigma_0, \beta_0$ — they report only modest accuracy gains with its inclusion. It is also important to note that while the authors experiment with an arc-hybrid system (Kuhlmann et al., 2011), this framework is applicable to any algorithm for which a set of transitions T is defined.

Kiperwasser and Goldberg (2016) also propose an arc-factored graph-based scoring module, which likewise features an MLP as the scoring function:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in G} \sum_{(i,j) \in E} f(h_i \oplus h_j)$$

Here, h_i, h_j refer to the indices of possible head and dependent representations, which, as before, are extracted from the BiLSTM-encoded input features. Since this approach only accounts for unlabeled trees, Kiperwasser and Goldberg (2016) also propose using a separate MLP in order to score labels for the relation between w_i and w_j:

$$\hat{r}_{i,j} = \arg\max_{r \in \mathcal{R}} \texttt{MLP}(h_i \oplus h_j)$$

The arc and label MLPs are each trained via separate loss functions, the sum of which is minimized throughout training. As such, this system can be viewed as an instance of multi-task learning, where the BiLSTM and embedding parameters are updated jointly in accordance with both arc scoring and labeling objectives.

Dozat and Manning (2017) expand upon this algorithm by leveraging two separate MLPs in order to learn representations for a token w_i appearing in distinct head and dependent roles:

$$h_i^{\text{head}} = \texttt{MLP}_{\text{head}}(h_i)$$
$$h_i^{\text{dep}} = \texttt{MLP}_{\text{dep}}(h_i)$$

where h_i is the BiLSTM hidden state corresponding to w_i. Following this, the head of w_i can be found by applying a biaffine transformation over its dependent representation h_i^{dep} and all possible head representations H^{head}:

$$s_i = H^{\text{head}} W h_i^{\text{dep}} + H^{\text{head}} b^\top$$
$$y_i' = \arg\max_j = s_{ij}$$

Effectively, the goal of the biaffine operation is to output an optimal score matrix from which a maximum spanning tree can be decoded. To this end, Dozat and Manning (2017) also apply two separate MLPs and another biaffine function for dependency *labeling*, which is operationalized as its own task with a separate loss — similarly to Kiperwasser and Goldberg (2016).

Dozat and Manning (2017) demonstrate that their biaffine transformation technique outperforms the graph-based approach of Kiperwasser and Goldberg (2016), leading to significant gains across the Chinese and English Penn Treebanks. Furthermore, their model proves to be highly accurate across the entirety of UD, placing as the top overall system in the CoNLL 2017 shared task (Dozat et al., 2017; Zeman et al., 2017). Beyond this, the biaffine scoring module has served as the base parsing component for models built on language model representations, such as ELMo (Peters et al., 2018) or BERT (Devlin et al., 2019). For example, Che et al. (2018) show that replacing the embedding function e with pretrained, language-specific ELMo representations can lead to further improvements in UD parsing. Similarly, Kondratyuk and Straka (2019) demonstrate that fine-tuning multilingual BERT as the feature function ϕ can lead to even higher performance gains.

2.4 Language Modeling

The language modeling task involves predicting the probability of a sequence of words in natural language. Formally, given a sequence of words $S = (w_1, w_2, ..., w_n)$ of length n, a language model aims to estimate the conditional probability distribution over a word $w_i \in S$ given some available context $C \subset S$. Traditionally, this amounts to predicting the next word w_{i+1} given the *past* context $C = (w_1, w_2, ..., w_i)$: $P(w_{i+1}|C)$. This formulation follows from the sequential nature of language production and comprehension, wherein word are uttered and processed incrementally, one after another. Alternatively, language modeling can entail predicting a word w_i provided its past and *future* contexts $C = (w_i|w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n)$: $P(w_i|C)$. In both cases, the conditional probabilities are calculated by means of fitting a statistical model on a corpus of natural language text. Unlike dependency parsing, the data employed for training language models is typically un-annotated, naturally occurring text, which can comprise newswire, novels, blogs, and many

other genres. As such, language modeling is considered a *self-supervised* task, in that it does not require labeled data and instead relies on supervision signals from the data itself.

Language models are typically evaluated via the perplexity metric, which measures how well a given model can predict a test set of sequences based on the probability distribution learned from a training set. Formally, given a language model f_θ and a test set of sequences X_{test}, the perplexity of f_θ on X_{test} is defined as:

$$\text{PPL}(X_{\text{test}}) = 2^{-\frac{1}{n}\log_2 P_\theta(X_{\text{test}})}$$

where $P_\theta(X_{\text{test}})$ is the probability of the test set according to the language model and n is the number of tokens occurring in X_{test}[3]. Intuitively, perplexity can be interpreted as the average number of bits needed to represent each word $w_i \in X_{\text{test}}$, according to the probability distribution learned by f_θ. A lower perplexity indicates that f_θ is better at predicting X_{test}, as it can assign higher probabilities to the sequences of words that were, in fact, observed[4].

2.4.1 N-gram Language Models

N-gram models are a basic class of language models that seek to estimate the probability of word sequences based on the frequency of n-grams occurring in a text corpus. In this sense, an n-gram is a contiguous sequence of words, the frequency of which is employed towards fitting a statistical model. Specifically, n-gram models are built upon the Markov assumption, which states that the probability of a word in a sequence depends only on a fixed number of preceding words, rather than on the entire history. For a bigram model, this amounts to the sole preceding token:

$$P_\theta(w_{n+1}|w_1, w_2, ..., w_n) \approx P_\theta(w_{n+1}|x_n)$$

The Markov assumption leads to the simplification of the model in question, as it reduces the number of parameters required to estimate conditional probabilities. Furthermore, it allows the joint probability of a sequence to be estimated via the chain rule:

[3]The log base 2 is used to calculate the perplexity in bits (though the natural logarithm is also occasionally employed).

[4]It should be noted that Perplexity is typically measured with regard to past context, where $C = (w_1, w_2, ..., w_n)$. In cases where the context incorporates past and future tokens $C = (w_i|w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n)$, variations of perplexity must be employed, in accordance with how $P(w_i|C)$ is modeled. For example, see Salazar et al. (2020), who propose *pseudo* perplexity for BERT (Devlin et al., 2019) (discussed in Section 2.4.2).

$$P_\theta(w_1, w_2, \ldots, w_t) = P_\theta(w_1)P_\theta(w_2|w_1)P_\theta(w_3|w_2)\cdots P_\theta(w_n|w_{n-1}) \quad (2.4)$$

where $P_\theta(w_j|w_i)$ is the conditional probability of the bigram (w_i, w_j). This can be straightforwardly estimated by counting the frequency of tokens occurring in a text corpus as follows:

$$P_\theta(w_j|w_i) = \frac{\texttt{freq}(w_i, w_j)}{\texttt{freq}(w_i)}$$

where $\texttt{freq}(x_i, x_j)$ is the absolute frequency of a given bigram.

This method is referred to as Maximum Likelihood Estimation (MLE), and it is widely considered to be the most simple and effective method for calibrating n-gram language models. However, it likewise carries numerous limitations in its simplicity. Chief among these is the data sparsity problem, where many n-grams may never appear in the training corpus. This leads the model — no matter how large — to automatically assign a probability of 0 such n-grams, which can be catastrophic when calculating joint probabilities via chain rule, as in Eq. 2.4. To overcome this problem, techniques such as smoothing and interpolation are often employed as a means of adjusting the probability estimates of unseen tokens or n-grams (Chen and Goodman, 1999).

2.4.2 Neural Language Models

In recent years, neural networks have emerged as the language modeling framework of choice, vastly outperforming classical n-gram models across many established benchmarks. Given a sequence of words S as input, a neural language model f_θ is tasked with outputting a probability distribution for the next possible token w_{i+1}. This is accomplished via an encoder function ϕ (such as the LSTM described in Section 2.3.2), which allows the model to capture dependencies between words separated by arbitrary distances in S. Specifically, the encoder computes a hidden state vector h_i at each time step i, which serves to summarize the information pertaining to the previous words in the sequence:

$$h_i = \phi(h_{i-1}, x_i)$$

where h_{i-1} is the hidden state vector at the previous time step, and x_i is the embedding of the current input word w_i. As in neural parsing, the embedding x_i is typically obtained by means of a lookup function e:

$$x_i = e(w_i)$$

The probability distribution for w_{i+1} is then obtained by passing h_i through a softmax-activated affine transformation[5]:

$$P_\theta(x_{i+1}|x_1, x_2, ..., x_i) = \text{softmax}(Wh_i + b)$$

where W is a weight matrix and b is a bias vector. This distribution is typically computed over the entire vocabulary V, comprising all tokens observed during training. Given this formulation, the parameters θ are updated throughout training in order to minimize the negative log-likelihood of the training data:

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log P_\theta(w_i|w_1, w_2, \ldots, w_{i-1})$$

where n is the number of training examples.

Operationalized in this manner, neural language models carry numerous key advantages over their n-gram-based counterparts. Most importantly, they are able to capture long distance dependencies by means of the encoder mechanism, whereas n-gram models are restricted to windows of several words. This applies not only to sentences, but also to longer to contexts such as paragraphs and documents. Relatedly, contextualized representations allow neural models to infer information about unseen words, provided they have been trained on appropriate data and were properly calibrated. N-gram models, on the other hand, must rely on mechanisms such as smoothing and back-off in order to avoid assigning probabilities of 0 for unseen tokens. Taken together, these qualities have contributed to an explosion of interest towards language modeling, where such models are studied not only for their expressive generation capabilities, but also their ability to capture rich contextual features (Qiu et al., 2020; Xia et al., 2020; Zhou et al., 2023; Min et al., 2021). I describe two influential neural language models below, which are studied throughout this thesis.

ELMo

ELMo (Embeddings from Language Models) is a language model-based representation learning technique proposed by Peters et al. (2018). Given a sequence of words S, ELMo computes a vector representation for each word $w_i \in S$ as a function of the entire input sentence. This allows the model to capture the context-dependent meaning of the word, which can then be applied to downstream tasks.

In formal terms, let $x_i \in \mathbb{R}^d$ denote the i-th word embedding in a sequence of word embeddings $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ over S. ELMo first computes a

[5]The softmax function is a non-linearity ensuring that the predicted probabilities sum to 1.

forward and backward representation for w_i by passing $\mathbf{x}$ through a BiLSTM, as in Eq. 2.3. The Bidirectional Language Model (BiLM) is then trained to jointly minimize the negative log likelihood of the next token as predicted from both forward and backward directions:

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{BiLM}(w_i|w_1, w_2, \ldots, w_{i-1}) + \log P_{BiLM}(w_i|w_i+1, w_i+2, \ldots, w_n)$$

Peters et al. (2018) train the BiLM on the 1B Word Benchmark (Chelba et al., 2013), using a 2 layer BiLSTM with 4096 units in each layer. After training, contextualized word representations can be obtained by making a forward pass through the model and extracting the hidden vectors output by the BiLSTM. These vectors can then be employed as input features to a downstream task, such as question answering or natural language inference. In order to consolidate information across all layers of the BiLSTM, Peters et al. (2018) propose to learn a scalar mixture of representations:

$$h_i = \gamma \sum_{j=0}^{L} s_j BiLM_{i,j} \tag{2.5}$$

where L is the number of layers, s is a task-specific, softmax-scaled weight vector and γ is a learnable parameter employed to aid optimization. They demonstrate that this technique, when applied to a downstream task, can lead to sizable performance gains with respect to static word embedding baselines. Specifically, they report state-of-the-art performance across a variety of NLP tasks, such as question answering, natural language inference, and semantic role labeling. Che et al. (2018) demonstrate that this framework can likewise be applied to a variety of languages, leading to strong performance in multilingual dependency parsing.

BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) is a language model-based representation learning technique that, like ELMo, has lead to significant performance gains across a variety of NLP tasks. Unlike ELMo, however, BERT is built upon the Transformer architecture (Vaswani et al., 2017) — an alternative sequence processing framework that has replaced RNNs as the de-facto standard for neural NLP. In order to understand how BERT works, I first provide a brief description of the Transformer below.

The Transformer is based on multi-head self-attention — a fully connected alternative to recurrence and gating. Formally, a Transformer takes a sequence

of embedding vectors $\mathbf{x} = (x_1, x_2, \ldots x_n)$ as input and applies a positional encoding to them, in order to retain the order of words in a sentence. These inputs are then transformed into query (Q), key (K), and value (V) representations via three separate linear transformations — W_i^Q W_i^K, W_i^V — and passed to an attention mechanism. A single attention head computes scaled dot-product attention between K and Q, outputting a weighted sum of V:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.6}$$

For multihead attention (MHA), the same process is repeated for k heads, allowing the model to jointly attend to information from different representation subspaces at different positions (Vaswani et al., 2017). Ultimately, the output of all heads is concatenated and passed through a linear projection W^O:

$$H_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{2.7}$$

$$\text{MHA}(Q, K, V) = \text{concat}(H_1, H_2, ..., H_k)W^O \tag{2.8}$$

Following this, the output of MHA is passed successively through a LayerNorm (Ba et al., 2016) with residual connections (He et al., 2016), onto a two-layer MLP with ReLU activation, and then another LayerNorm (also with residual connections).

The full Transformer architecture features an encoder and decoder — both of which are composed of multiple layers of the aforementioned sub-modules. However, though these components were initially proposed to accommodate sequence-to-sequence tasks (such as machine translation), they have likewise shown considerable utility when employed in isolation. For example, the GPT family of models (Radford et al., 2018, 2019; Brown et al., 2020) employs the decoder as a means of training autoregressive language models at successively increasing scale. BERT, on the other hand, relies entirely on the encoder as a means of capturing rich, bidirectional feature representations.

The original BERT model is trained on two separate tasks: masked language modeling (MLM) and next sentence prediction (NSP). In MLM, 15% of the tokens in a given input sequence S are selected to be randomly *masked.* Of these, 80% are assigned the `[MASK]` token, 10% are replaced with a randomly sampled word from the vocabulary, and 10% are unchanged. The model is tasked with predicting the identity of the masked tokens given the remaining available context. For example, provided the input sequence *my dog is* `[MASK]`, the model might predict *cute*. The MLM task encourages the model to learn a bidirectional representation of the input sequence, as it must account for both the preceding and following tokens when making predictions.

In NSP, the model is tasked with predicting whether or not a given pair of sentences appear after one another. This amounts to a binary classification task, where a model must deduce that the sentence *he likes to play fetch*

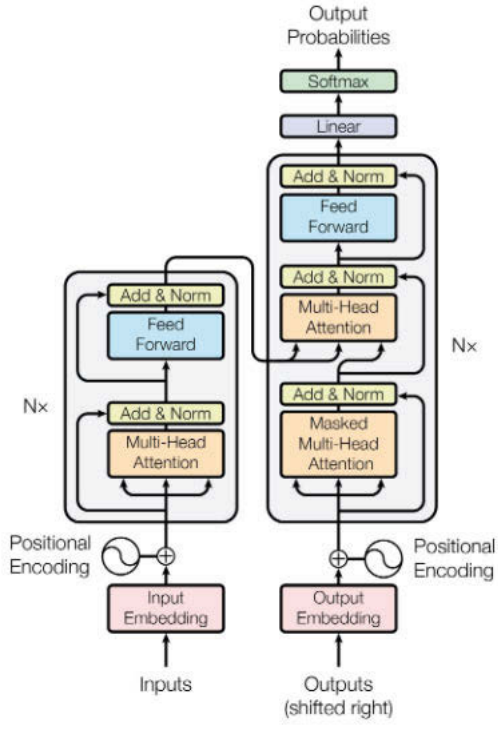

Figure 2.6. The Transformer architecture, as illustrated in Vaswani et al. (2017).

plausibly follows *my dog is cute*. NSP aids the model in capturing any possible relationship between sentences and improves its ability to perform tasks that consolidate information over multiple sequences. Given this formulation, Devlin et al. (2019) train BERT on a combination of English Wikipedia and the BooksCorpus (Zhu et al., 2015b), jointly minimizing the negative log-likelihood of the MLM and NSP objectives. They release a `base` version of the BERT architecture (12 layers, 12 heads, 768 hidden units, 110 million parameters) as well as a `large` version (24 layers, 20 attention heads, 1024 hidden units, 340 million parameters). In addition, they also make a *multilingual* version of the `base` model available, which was trained on the concatenation of the 104 largest Wikipedias (by language, as of 2019).

Devlin et al. (2019) demonstrate the utility of BERT by *fine-tuning* the model on a variety of NLP tasks. This process entails the application of a single classification layer to the reserved `[CLS]` token, which is prepended to every input sequence per requisite. In doing so, the MLM and NSP task losses are replaced by a task-specific loss, which is used to update the model's parameters. Devlin et al. (2019) report that this framework leads to state-of-the-art performance on the GLUE suite, surpassing ELMo and GPT-1 (Radford et al., 2018). They attribute BERT's success to its bidirectional pre-training proce-

dure, which allows the model to learn rich contextualized representations that that can be leveraged across a variety of downstream tasks.

3. Enhancing Dependency Parsers with Language Model Representations

This chapter is dedicated to investigating the behavior of neural dependency parsers. Specifically, I assess the extent to which traditional wisdom about parsing algorithms applies to the neural paradigm — in particular, sophisticated representation learning techniques like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). In this manner, I address the first research question of the dissertation:

RQ1 How do representations from pre-trained language models affect the behavior of neural dependency parsers?

The experiments in this chapter are largely based upon the methodology of McDonald and Nivre (2007, 2011), who characterize the behavior of pre-neural transition-based and graph-based parsers by means of an extensive error analysis. We extend their framework to the neural scenario in Paper I, making use of the transition-based and graph-based parsing algorithms proposed by Kiperwasser and Goldberg (2016). First, we perform a comparative error analysis between the two aforementioned parsers, so as to corroborate whether or not the findings of McDonald and Nivre (2007, 2011) also apply to neural models. Going further, we extend both parsers by incorporating contextualized word embeddings extracted from ELMo and BERT. We then repeat the error analysis for these contextual models in an effort to gauge their overall utility of contextualized embeddings in parsing, as well as the effect they bear on the behavior of the transition-based and graph-based algorithms.

3.1 Characterizing the Error Profiles of Neural Dependency Parsers

As mentioned in Chapter 2, the application of the deep learning paradigm has led to notable leaps in performance within dependency parsing. A key advantage of this new approach is that it obviated the need for feature engineering, leaving researchers and practitioners to focus on the design of model components instead. This freedom manifested itself in many ways, with researchers proposing new structured learning objectives (Le and Zuidema, 2014; Dyer et al., 2015; Zhu et al., 2015a), tree-based sequence processing algorithms

(Bowman et al., 2016; Dyer et al., 2016), syntax-oriented embeddings (Levy and Goldberg, 2014), among other approaches. However, accuracy-wise, many of these methods did not significantly outperform the simple, modular solution of Kiperwasser and Goldberg (2016), who showed that an accurate dependency parser of any type can be built on top of a strong feature extraction mechanism, such as a BiLSTM. Indeed, this paradigm — word embeddings coupled with a BiLSTM feature extractor and a task-specific output layer — would serve as the basis for a vast variety of NLP tasks for several years, ranging from POS-tagging (Plank et al., 2016) to natural language inference (Conneau et al., 2017) to summarization (Nallapati et al., 2016).

One key implication of the work of Kiperwasser and Goldberg (2016) relates to the BiLSTM's role as an all-purpose feature extraction mechanism. This is highlighted via the strong performance of both transition-based and graph-based algorithms, despite the stark differences between the two paradigms (McDonald and Nivre, 2007, 2011). As an illustration, recall that local, greedy transition-based algorithms require access to rich, structural features that relate contextual information beyond the single word being processed. This is not an issue for graph-based parsers, which conduct an exhaustive search over all possible arcs in the sentence, thus taking global sentence structure into account. As such, the simple transition-based parser of Kiperwasser and Goldberg (2016) stands to have more to gain from the BiLSTM mechanism, which must be capable of encoding the necessary context for every incremental decision.

In Paper I, we explore this intuition by employing the experimental design of McDonald and Nivre (2007, 2011) in the context of the neural parsers proposed by Kiperwasser and Goldberg (2016). McDonald and Nivre (2007, 2011) investigate the behavior of two popular pre-neural parsers — MaltParser (transition-based) and MSTParser (graph-based) — across a variety of factors carefully selected to isolate their strengths and weaknesses. These include sentence length, projective and non-projective dependencies, dependency length, distance to root, and various linguistic categories. In our experiments, we employ these same factors in order to assess how they influence the performance of neural transition-based and graph-based parsers across 13 typologically diverse treebanks.

In general, we find that the neural transition-based parser performs slightly worse than its graph-based counterpart, yielding aggregate LAS scores of 80.5 vs. 81.5. This is a bigger gap than what is reported in Kiperwasser and Goldberg (2016) (although we evaluate on different treebanks), leading us to believe that the transition-based parser still partially suffers from the issues outlined in McDonald and Nivre (2007, 2011). This is confirmed in the error analysis, where we observe that the transition-based parser fares worse in the expected settings, such as longer sentences, dependencies of longer length, and dependencies closer to the root. However, the extent to which the two parsers diverge is significantly less stark than what is observed in McDonald and Nivre (2007, 2011). For example, the gap between the transition-based and graph-

based parsers is much smaller across increasing dependency length, and the former degrades much more slowly than expected. The latter point can also be made for sentence length, where the transition-based parser only experiences a sharp drop in accuracy for sentences consisting of more than 50 words — a rare phenomenon in any situation.

Given the above findings, it is difficult to say whether the comparatively worse performance of the transition-based parser can be attributed to the BiLSTM or the transition scoring component itself. Indeed, the parser employed in Paper I does not employ the extended feature set of Kiperwasser and Goldberg (2016), which would provide additional context to the MLP. However, we were not inclined to experiment with this option given the lack of improvement reported in Kiperwasser and Goldberg (2016) with regard to its inclusion. In any case, the error profile described above suggests that error propagation remains a problem for this particular transition-based parser, despite the use of context-preserving mechanisms, such as the BiLSTM and a dynamic oracle (Goldberg and Nivre, 2012, 2013). This points to the potential limitation of the LSTM, which — like the transition-based model itself — is a sequence processor. This necessitates that the representation of a given token h_i only captures the history of the preceding context: $h_{1:i-1}$. Although the bidirectional component is designed to rectify this by incorporating the remaining, unseen context $h_{i+1:n}$, it is unclear precisely how much information about a possible dependent w_j is retained in the incremental hidden state — especially if the distance between head and dependent is large.

One potential improvement to the BiLSTM encoding can be incorporated at the embedding level. Notably, the baseline parsers employed in Paper I utilize pre-initialized FastText word embeddings (Bojanowski et al., 2017), which are updated along with the network's parameters. However, such embeddings are *not* contextual, and it is therefore the role of the BiLSTM to incorporate sentence-level context into each hidden state representation. As discussed above, this alone appears to be insufficient for the transition-based parser explored in Paper I.

3.2 Incorporating Contextualized Embeddings

Models like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) provide a solution to the above problem by computing *contextualized representations* for every token in a sentence. A key distinction between this type of model and their static counterparts is that, while the latter approach employs a lookup table in order to retrieve cached vectors, the former computes representations dynamically by means of a pretrained encoder. Typically, this is either a stacked LSTM (ELMo) or Transformer (BERT) that is trained for language modeling on large amounts of unlabeled data. Pre-trained encoders are leveraged by either 1) making a forward pass through the model and extract-

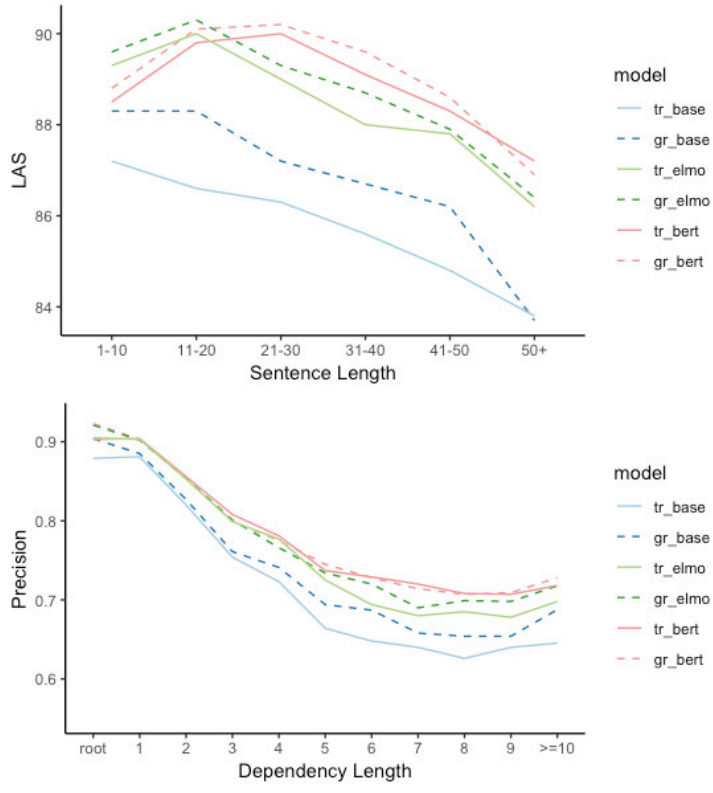

Figure 3.1. Error profiles for transition-based and graph-based parsers, across various sentence length (top, measured in LAS)) and dependency length (bottom, precision) bins. `tr` and `gr` refer to transition-based and graph-based parsers, respectively.

ing the hidden states from the encoder or 2) updating the encoder parameters with respect to the task in question. These two processes are often referred to as *contextual embedding* (Peters et al., 2018) and *fine-tuning* (Howard and Ruder, 2018; Devlin et al., 2019), respectively. In both cases, utilizing a pre-trained encoder downstream has been empirically shown to boost performance across a wide array of NLP tasks. This is generally understood to relate to richness of the hidden representations, which capture language usage across millions of diverse contexts in heterogeneous domains.

In Paper I, we show that ELMo and BERT representations provide a significant boost to parsing accuracy when employed as contextual embeddings. This amounts to an average LAS improvement of +4.0 and +2.8 to the transition-based and graph-based parsers when using ELMo, respectively, as well as +4.4 and +3.1 when using BERT. Indeed, we clearly see a stronger benefit to the transition-based parser when incorporating either model, which now performs on par with the graph-based parser (on average across treebanks). This indicates that the ELMo and BERT representations supplement the parser

with relevant, global-level features that, in the case of the transition-based parser, could not be directly learned with the BiLSTM alone. We corroborate this in terms of error profile, where the ELMo-based transition model closes the gap with its transition-based counterpart across all settings, while the BERT-based models perform virtually identically. Figure 3.1 depicts this phenomenon across various sentence and dependency length settings.

These findings carry numerous implications. Primarily, they demonstrate that neural dependency parsers are able to achieve strong performance via generic encoder mechanisms, such as BiLSTM, which automatically extract features in a task-specific manner. This represents a large step away from the classical dependency parsing paradigm, wherein features were manually selected according to limitations of the parsing algorithm of choice. Moreover, the observed benefit of incorporating pre-trained contextual embeddings implies that the observations put forth by McDonald and Nivre (2007, 2011) hold less relevance for neural parsers. This is a striking result, suggesting that the choice of input representation and the method by which it is incorporated bears more weight on parsing accuracy than the parsing algorithm itself. Indeed, more recent parsing studies seem to demonstrate similar insights, focusing on efficient and accurate methods for leveraging pre-trained language models, rather than the development of new parsing algorithms (Kondratyuk and Straka, 2019; Üstün et al., 2020; Glavaš and Vulić, 2021).

The extent and nature of the ELMo- and BERT-based accuracy boost raises further questions about the information encoded in their respective representations. In one sense, it is not surprising that incorporating such models could provide a performance boost over the use of static embeddings like FastText: while BERT was trained 104 concatenated Wikipedias, FastText only accounts for a single Wikipedia per language[1]. As such, a given word is likely to be represented by a much wider array of contexts in the larger corpora, thus increasing the potential robustness of its representation. Likewise, the ELMo and BERT architectures are well equipped to capture such diverse language usage, given that they comprise 94 and 110 million trainable parameters, respectively. In any case, it is difficult to ascertain the nature of the information captured by these models, both of which employ variations of a self-supervised language modeling objective. On one hand, they likely encode distributional features, acting as noisy estimators of mutual information between predicted words and their contexts (Kong et al., 2020). On the other hand, given their demonstrated utility for dependency parsing, it is possible that they likewise capture abstract structural concepts like grammatical roles or, by extension, parse trees[2]. In

[1]ELMo was also trained on individual Wikipedias per model, but this data was likewise concatenated with all common crawl data available for each language (Che et al., 2018).

[2]Indeed, the murky relationship between word co-occurrence and syntactic categories has been previously investigated in the context of (unsupervised) parsing (Yuret, 1998; Klein and Manning, 2004).

Chapter 4, we detail two experiments that attempt to investigate evidence of the latter.

3.3 Summary

In this chapter, we investigated RQ1 in the context of the transition-based and graph-based parsers proposed by Kiperwasser and Goldberg (2016). First, we employed the methodology of McDonald and Nivre (2007, 2011) in order to characterize the error profiles of each parser. To this end, we showed that the transition-based parser performs generally worse than its graph-based counterpart, particularly in the expected settings: longer sentences, dependencies of longer length, and dependencies closer to the root. However, we also observe that both parsers diverge much less drastically in their error profiles than what is reported McDonald and Nivre (2007, 2011). We surmise that this is likely due to the BiLSTM mechanism, which is able to capture rich contextual features for each token in the input, thus benefiting both the local, greedy transition-based parser and the global, exhaustive graph-based parser.

Going further, we incorporate contextualized embeddings extracted from ELMo and BERT into both parsers. We find that this yields a considerable performance boost, with BERT-based models yielding the highest accuracy overall. In terms of error profile, we observe that the transition-based model benefits most from the contextualized features, closing the gap with its graph-based counterpart across all experimental settings. These findings corroborate the richness of language model representations, suggesting that they are capable of encoding syntactic information that is beneficial for downstream parsing.

4. Searching For Syntax

In Chapter 3, we demonstrated that contextualized embeddings extracted from ELMo and BERT provide a sizable boost to the accuracy of neural dependency parsers. In particular, we showed that the richness of these representations is able to close a performance gap between transition-based and graph-based parsers, which behave almost identically after their incorporation. To further our understanding of the syntactic knowledge of language models, this chapter is dedicated to addressing our second research question:

RQ2 To what extent is syntax encoded in language model components, and how does this vary across different languages?

To do so, we hone in on two individual components of interest: hidden state representations and self-attention distributions. Regarding the former, we perform a series of probing experiments wherein we attempt to decode dependency structure from the hidden states of ELMo and multilingual BERT across 13 languages. For the latter, we investigate the extent to which the same structure is reflected in the attention distributions of multilingual BERT's 144 self-attention heads.

4.1 Probing Hidden State Representations

The success of language models across disparate NLP tasks prompted many researchers to ponder whether such systems possess generalized linguistic knowledge, and if so, how they come to acquire it. The Generalized Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) was proposed as a means of validating such questions across a suite of high level "language understanding" tasks, all of which presuppossed a sophisticated degree of linguistic reasoning. ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), and other related models yielded strong, sometimes superhuman performance across the GLUE suite (Nangia and Bowman, 2019; Yang et al., 2019), leading many researchers to answer the first part of the above question affirmatively, but without a strong intuition about how to pursue the second. Indeed, the nature of such models makes the prospect of identifying and localizing their knowledge a difficult endeavor. Specifically, they feature hundreds of millions of parameters and non-linearities, are randomly initialized, and trained end-to-end with a generic word prediction objective. This makes

it challenging to ascertain which, if any, aspects of linguistic structure are actually encoded by these networks and which components are responsible for capturing them.

Early pursuits in this line of inquiry largely concerned *probing* — a method designed to verify the information contained in hidden state representations (Adi et al., 2017; Ettinger et al., 2016; Belinkov et al., 2017; Conneau et al., 2018; Hupkes et al., 2018). As a technique, probing presupposes white-box access to a pre-trained model, such that hidden state vectors (computed with respect to some input) can readily be extracted. These vectors are typically selected vis-a-vis a specific model component, such as the output of BERT at some specific layer, in an attempt to localize the model's processing patterns. They are then employed as input features to a *probe*: a classifier fit on training data for a task corresponding to a linguistic property, such as number or tense. If such a probe yields higher accuracy on a held-out test set than a random or non-contextual baseline, the representations on which it was trained can thus be interpreted as encoding the selected property.

Numerous studies have applied the probing paradigm to pre-trained language models, seemingly confirming the hypothesis that they encode linguistic information in their hidden state representations (Liu et al., 2019; Tenney et al., 2019b). Out of this work has emerged a consensus observation that such representations likewise *evolve* as they propagate up the network, with lower layers encoding lexical information like parts of speech, middle layers representing syntactic relationships, and upper layers capturing higher-level semantic features pertaining to semantic roles and co-reference (Tenney et al., 2019a). Such a phenomenon, viewed through the lens of probing, led Tenney et al. (2019a) to posit that BERT "rediscovered the classical NLP pipeline": in other words, that the model could learn to perform ostensibly supervised linguistic tasks in an entirely self-supervised manner. Tangentially, Hewitt and Manning (2019) also showed that dependency tree structure could be extracted from such models by applying a linear transformation over their hidden states. These findings implied that these models captured hierarchical syntactic structure in continuous space, wherein standard measures like vector distances and norms could be employed as proxies for the symbolic notions of tree distance and node depth. Crucially, their results failed to generalize to various non-contextual baselines, such as static word embeddings, indicating that models like BERT and ELMo manage to encode some semblance of syntactic structure in their hidden states.

In Paper II, we apply the structural probe of Hewitt and Manning (2019) to BERT and ELMo models trained on 13 different languages (represented by UD treebanks). Our aim in doing so is to assess the extent to which their results — which are reported only in the context of English — hold up in the face of typological diversity. In general, we corroborate many of the findings of Hewitt and Manning (2019): namely, that BERT strongly outperforms ELMo and that probing accuracy for either model varies highly by layer. We also observe that incorporating information across all layers by means of a linear

mixture yields the highest probing accuracy overall (71.30 and 55.23 UAS for BERT and ELMo, respectively).

The above findings lead us to conclude that aspects of syntactic structure are indeed encoded in these models, albeit to varying extents and most reliably by BERT. However, when compared to the supervised parsing experiments discussed in Chapter 3, we nonetheless observe a large gap between those results and the probing accuracy described here. Given that the parsers in Paper I were trained on a different version of UD and did not incorporate the full network's representations (only layers 4–8), we fine-tune BERT using the parser of Kondratyuk and Straka (2019) in order to assess the exact extent of disparity between the two paradigms. To this end, we obtain an average supervised UAS of 89.89 across the 13 treebanks, which corresponds to an improvement of 18.60 over BERT's average probing accuracy. We interpret these findings to mean that, while syntactic structure can partially be decoded from these models' internal representations, it either a) is not an exact mapping to the UD framework (in the same way that representation learned by a parser might be), or b) cannot be fully extracted by means of this particular probe. As such, a specialized parsing architecture is required in order to tailor such representations for UD parsing and obtain the highest performance.

In addition to the variability in probing accuracy in terms of layer and model, we likewise observe that performance varies drastically across languages. For example, for both models, Turkish (58.08, 44.62 for BERT, ELMo) and Chinese (61.87, 45.51) return the lowest UAS overall, while Hindi returns the highest (80.38, 65.22). In terms of standard deviation, this amounts to 7.15 and 7.64 UAS for the BERT and ELMo linear mixtures, respectively. Such a high typological variability can inspire several interpretations. Straight-forwardly, one can opine that the low scoring languages are not well represented in the feature spaces of such models. In the case of multilingual BERT, which was trained on 104 Wikipedias, this could itself be a symptom of lack of pretraining data for a particular set of languages. Turkish, for example, ranks 27th with approximately 540,000 articles (compared to 6 million for English). However, such an observation is at odds with the fact that Hindi ranks even lower at 59th (155,000 documents), despite yielding the highest overall decoding accuracy.

Another explanation could point to the fact that the size of the training data across different treebanks varies significantly, thereby affecting the potential quality of the probe's fit. Indeed, the low-scoring Turkish and Chinese treebanks contain 3,664 and 3,997 sentences, respectively, while the high-scoring Hindi and Italian both contain 13,304 and 13,121. However, to this end, we did not observe significant changes in probing performance even when controlling for treebank size. Likewise, size did not appear to be a factor for supervised parsing, where variance in performance was even smaller. Other possible explanations could factor in typological reasons, the design of the probe, or even the nature of UD as a syntactic framework. We address these topics in Chap-

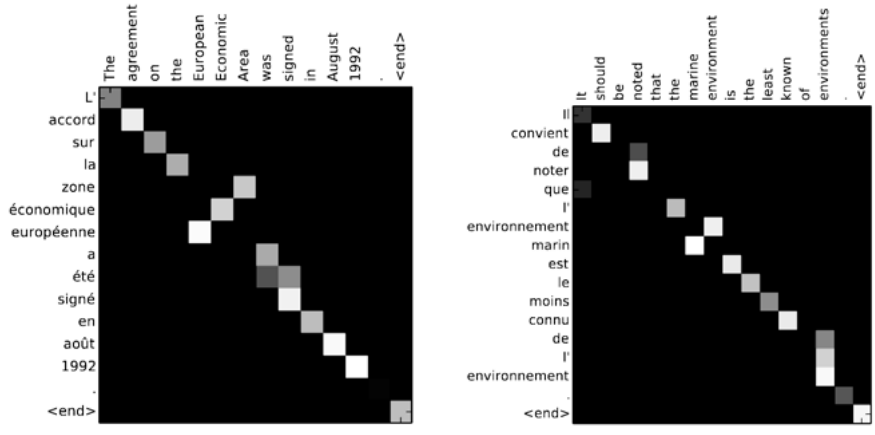

Figure 4.1. Automatically induced attention-based alignment between English and French sentences, as shown in Bahdanau et al. (2015).

ter 5. Alternatively, one could point to the nature of the training data itself, in terms of difficulty, sample diversity, and annotation consistency. This is the focus of Chapter 6.

4.2 Extracting Syntactic Structure From Attention Heads

An alternative means of examining the syntactic structure encoded by language models has been to analyze attention distributions. This methodology fundamentally differs from the probing paradigm in that no auxiliary model is required in order to assess the correspondence between input tokens. After all, the attention mechanism first proposed by Bahdanau et al. (2015) was designed, in part, to be *interpretable*. This meant that one could extract an attention matrix after training and directly examine the extent of "attraction" between tokens. To this end, Bahdanau et al. (2015) showed that, for RNN-based machine translation models, attention encoded word alignment across source and target sentences. Figure 4.1 demonstrates this phemenon in practice, where the French word *européenne* is shown to accurately attend to its English translation *European* — despite differences in word order.

Though variants of attention have been proposed for numerous models, much of the focus in understanding the mechanism has been directed towards Transformers. This is due to the fact that Transformers rely exclusively on attention as a means of incorporating contextual information within and across sentences (as described in Section 2.4.2). In a typical sequence-to-sequence Transformer network, attention is incorporated at three distinct levels: via self-attention in the encoder, via cross-attention between encoder and decoder hid-

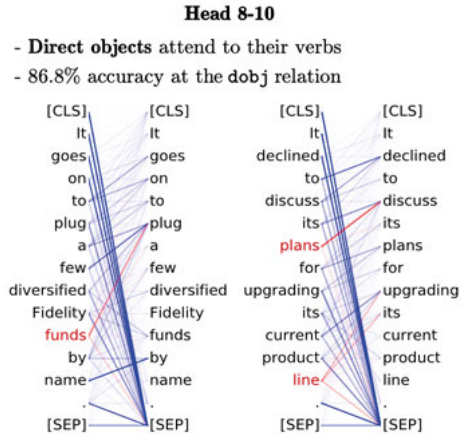

Figure 4.2. Visualization of head 8–10 in BERT base specializing in the `dobj` relation. Incoming/outgoing attention corresponding to this relation is highlighted in red, all other attention in blue. Figure taken from Clark et al. (2019).

den states, and via masked attention in the decoder. RNNs, in contrast to this, only consolidate inter-sentence context via recurrence and gating, reserving cross-attention for sequence-to-sequence tasks like machine translation.

Out of the Transformer's three attention mechanisms, self-attention has arguably been analyzed most widely. This is due to the fact that it can be computed with respect to a single input sentence and does not presuppose conditioning on another sequence. The output of a single self-attention head is thus a normalized matrix Att $\in \mathbb{R}^{N \times N}$ (where N is the sequence length), wherein a single cell conveys the relative importance of one input token to another. In essence, self-attention encodes an input sentence as a fully-connected graph, the edge weights of which are then employed for computing intermediate hidden state representations. This property has been observed by various researchers to bear resemblance to the score matrices employed for graph-based parsing, from which spanning trees can be decoded via algorithms like Chu-Liu-Edmonds (Chu, 1965; Edmonds et al., 1967).

Raganato and Tiedemann (2018) were among the first to attempt to decode tree structure from Transformers' self-attention matrices. In their study, they investigated the extent to which maximum spanning trees extracted from the attention heads of machine translation models aligned with gold-annotated UD treebanks. They showed that, while the decoding accuracy (measured via UAS) was often significantly higher than a random tree baseline, it was rarely much better than exclusively right-branching trees. Indeed, the highest score reported was merely 36.08 UAS overall, extracted from the 5th attention head in Layer 2 of the English $\rightarrow$ Russian model. In addition to this, the authors reported strong variance in decoding accuracy across layers and attention heads, with most heads returning near-random accuracy and only a select few per-

forming on par with the right-branching baseline. Many similar insights were reported by Clark et al. (2019) in the context of BERT, which also performed on par with a right-branching baseline and exhibited high variance in decoding accuracy per layer and head. However, Clark et al. (2019) likewise demonstrated that BERT's attention heads tended to individually "specialize" in select dependency relations, sometimes returning higher accuracy than positional baselines. For example, they showed that head 10 in layer 8 captured direct object dependencies with an accuracy of 86.8, compared to the 40.0 baseline score when accounting for the most frequent offset at which the words in this relation occurred (what they refer to as a "positional baseline").

The aforementioned findings, inter alia, paint a complicated picture of how syntactic structure is encoded by Transformer-based models' attention heads. On one hand, it does not appear — in either BERT or MT models — that there exists a single generalist head capable of capturing sentence-level syntactic structure that closely aligns with UD. On the other hand, shades of structural elements do tend to be distributed throughout the network, with various heads learning to specialize in capturing syntactic relations. In either case, interpretation of such findings is complicated by the fact that all experiments are conducted in a monolingual scenario, focusing on English. This is potentially problematic, since the language's rigid word order and lack of inflectional morphology may instill a bias that confounds position with structure. Though Clark et al. (2019) make note of this and accordingly compare their relation-wise results with respect to positional baselines, nearly all of the relations they evaluate are left-branching and adjacent. In many such cases, the decoding accuracy is not much higher than the positional baseline (though the aforementioned direct object relation and several others are notable exceptions). In such cases, the specialist heads may be mischaracterized as syntactic rather than positional, when, in fact, they simply attend to adjacent tokens (Voita et al., 2019).

In Paper III, we replicate Clark et al. (2019)'s tree decoding experiments for multilingual BERT in the context of a Parallel Universal Dependencies (PUD) treebank set representing 18 typologically diverse languages (Zeman et al., 2017). Our intention in doing so is to forego the confounds introduced by the typological characteristics of English and investigate the extent to which the findings of Clark et al. (2019) generalize across different languages. Interestingly, we are able to corroborate many of that study's results, even in a multilingual setting. For example, although we fail to localize a generalist head, we nonetheless observe that numerous heads are able to decode full trees more accurately than an adjacent-branching baseline[1]. For the most part, layers 6 and 7 tend to perform unilaterally well across heads, although above-baseline heads are distributed across the network. Interestingly, though sev-

[1] We measure decoding accuracy in terms of UUAS, since self-attention is not explicitly directed. For instance, a head can allocate most of its attention to a single dependent, or vice versa.

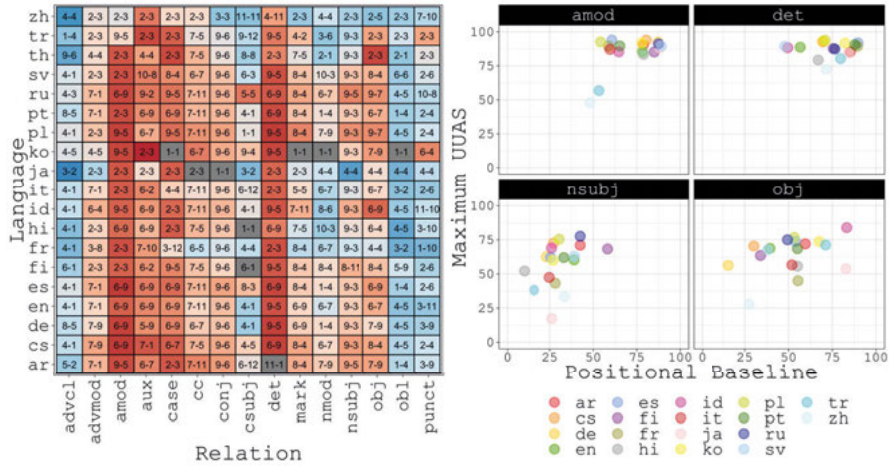

Figure 4.3. Left: UUAS per relation across languages (best layer/head combination indicated in cell). *Right*: Best UUAS as a function of best positional baseline (derived from the treebank), selected relations.

eral languages outperform their respective baselines by a wide margin (Czech: +13; German: +13; English: +11; Spanish +10; Swedish +12), other languages perform as well as exclusively right-branching trees (French, Italian, Japanese, French). Though we could not diagnose the exact reason for why the French and Italian treebanks return such low decoding accuracies, we suspect that Chinese and Japanese performance is influenced by multilingual BERT's tokenization scheme.

In honing in on individual dependency relations, we are likewise able to corroborate the phenomenon of specialist attention heads. For example, Figure 4.3 (right) demonstrates that modifier relations like determiners (`det`) and adjectives (`amod`) can respectively be decoded with accuracies of 88±6 and 85±12 across languages — significantly outperforming positional baselines. We observe that layer-head combinations of 2–3, 6–9, and 9–5 are largely responsible for encoding these relations (Figure 4.3, left), though the highest scoring head tends to vary slightly by language. We report a similar trend for more difficult "core" relations like nominal subjects (`nsubj`) and objects (`obj`), where decoding accuracy is likewise consistently higher than our baselines. This is especially interesting in the context of typological word order variation: while an SVO language like English might often place the subject directly to the left of the verb, an SOV language like Hindi might have it several positions further away, with an object and its potential modifiers intervening. Indeed, this is reflected in the positional baselines, where the most frequent offset in English returns an accuracy of 39%, compared to a mere 10% in Hindi. Despite this, decoding accuracy for both languages amounts to 60% and 52%, respectively, indicating that one particular head is devoted to tracking and composing sub-

jects with their associated verbs. Indeed, Figure 4.3 shows that attention head 9–3 is responsible for encoding the `nsubj` relation in 14 out of 18 languages.

While these results are suggestive of the role of attention in tracking syntactic structure and syntactic relations, they are far from conclusive. Indeed, though attention distributions appear to encode dependency trees more accurately than adjacent-branching baselines (in most cases), these results are still a far cry from the performance expected of supervised parsers. This is likely due to a variety of reasons. Namely, the distributed nature of BERT's attention suggests that no single head can act in a generalist capacity. Clark et al. (2019) experiment with this notion by consolidating all of English BERT's attention heads. To this end, they propose a probe that learns linear combinations of weights extracted from the entire network. However, though this approach yields higher accuracy than an individual head, it nonetheless significantly underperforms Hewitt and Manning (2019)'s structural probe (though the reported metrics are not strictly comparable).

Another explanation for the low decoding accuracy could be that BERT's internal representation of syntax (expressed via attention) does not align well with the UD framework. Limisiewicz et al. (2020) address this hypothesis by proposing several modifications to the UD scheme, which are tailored to BERT's observed attention patterns. They report moderate gains to decoding accuracy when incorporating such changes, and demonstrate that ensembling over several selected heads provides a further boost. However, even in the best setting where 48 heads are ensembled, their overall tree-decoding accuracy for English remains comparatively low at 52 UAS. In conjunction with Clark et al. (2019)'s probing results, this finding suggests that — even in favorable, supervised settings — BERT's self-attention can only encode a limited view of syntactic structure.

To this end, one may wonder if self-attention *needs* to encode syntactic structure in order to succeed at the masked language modeling objective. After all, such a task is mostly lexical in nature: a network is tasked with guessing the identity of missing tokens in a corrupted sequence. Though some missing context can presumably be inferred via syntactic structure, there is no aspect of the training regime that is explicitly tasked with encoding it. Htut et al. (2019) experiment with this notion by decoding dependency trees from BERT's attention matrices *after* it has been fine-tuned on the COLA (grammatical acceptability) (Warstadt et al., 2019) and MNLI (recognizing textual entailment) (Williams et al., 2018) tasks. In doing so, they hypothesize that the respective syntactic and semantic nature of such tasks could predispose the network towards encoding syntactic information as a byproduct of the fine-tuning process. However, though they corroborate the existence of distributed specialist heads in their experiments, they are unable to articulate any meaningful difference in performance across models when decoding attention via MST. Taken together with the aforementioned studies, such findings imply that attention, as a mechanism, is incapable of encoding syntactic structure in a readily in-

terpretable, generalist fashion — regardless of training objective. However, to this end, it is also worthwhile to ponder whether the tasks chosen by Htut et al. (2019) *require* syntactic knowledge in order to be solved. In fact, much prior work suggests that they *do not*: models trained for MNLI, for example, have been shown to heavily rely on heuristics and lexical information, to the extent that input sentences can be re-ordered, truncated, or otherwise corrupted without performance significantly degrading (Gupta et al., 2021; Sinha et al., 2021a,b; Clouatre et al., 2022). With this in mind, it seems unreasonable to expect a model that behaves in such a manner to transparently encode syntactic structure. In Chapter 5, we investigate whether or not this remains the case after fine-tuning BERT on an explicit, dependency parsing-oriented objective.

4.3 Summary

In this chapter, we investigated RQ2 via two separate experiments concerning hidden state representations and self-attention distributions, respectively. In the first experiment, we applied the structural probe of Hewitt and Manning (2019) to ELMo and BERT representations in an attempt to decode dependency structure. We found that, while the probe cannot match the performance of a supervised parser, it can nonetheless decode dependency structure with reasonable accuracy. To this end, we note that probing accuracy varies highly across models, with BERT strongly outperforming ELMo, as well as across layers within models. Per the latter point, we observe that BERT's middle layers (6, 7, and 8) return the highest accuracies overall. In applying the probe to treebanks across 13 languages, we observe significant variation in probing accuracy. This variation is much stronger than what is observed for supervised parsing, warranting further investigation.

In the second experiment, we attempted to decode dependency structure from the the self-attention heads of multilingual BERT. We were able to corroborate the phenomenon of specialist heads, which tracked select dependency relations more accurately than various positional baselines. We also observed that such specialist heads reflected the same relations across various languages, signaling a degree of multilingual alignment. We were, however, unable to localize any single generalist head which was capable of performing holistic parsing over an entire sentence. Though we observed that dependency trees were generally decodable above the rate of a adjacent-branching baseline, we failed to identify an overall trend across heads, layers, and languages.

5. On the Role of Experimental Variables

In Chapter 4, we investigated the extent to which syntax is encoded by language model components. In focusing on hidden state representations, we found that dependency structure can indeed be extracted when utilizing the structural probe of Hewitt and Manning (2019). Likewise, we showed that syntactic information is distributed across the entire multilingual BERT network, with attention heads specializing in select grammatical relations, such as nominal subjects. However, though these findings were largely affirmative, we also made note of numerous trends that warranted further investigation. For example, we observed significant variation by language in the probing experiments, as well as a significant gap in accuracy between probing and supervised parsing. Furthermore, in the attention experiments, we were unable to localize a generalist head that was capable of encoding full tree structure in its weights. Taken together, these findings imply that the probing and attention decoding methodologies paint a potentially misleading picture of the syntactic capabilities language models.

In this chapter, I aim to shed a nuanced light on the findings described in Chapter 4 by answering the third research question:

RQ3 How is our understanding of language model components affected by various experimental variables, such as the choice of alternative syntactic frameworks or learning objectives?

To this end, I describe follow-up experiments to each of the probing and attention decoding studies discussed in Chapter 4. With regard to the former, we compare the UD-based probing results to the accuracy obtained by a probe trained on SUD: an alternative annotation framework representing a "surface syntax" dependency analysis, described in Section 2.2. In addition to this, we revisit the attention decoding experiments by investigating the extent to which generalist heads emerge after fine-tuning BERT on a supervised parsing objective.

5.1 Issues with Probing

Though probing has enjoyed widespread usage as a means of investigating the hidden state representations of neural models, it has likewise received considerable criticism as a paradigm. Many of its shortcomings are outlined by

Belinkov (2022), who conducts a widespread survey of the paradigm and its usages. Chief among these is the issue how models interface with the data employed towards attesting linguistic properties. More generally, the argument of Belinkov (2022) concerns the supervised nature of most probing tasks, which presuppose the use of an appropriate model to serve as a probe. In the process of being fit on a finite dataset, the probe inevitably learns to correlate input samples X with their corresponding labels Y. This means that the probe is not only tasked with associating properties of the input representation with the linguistic task in question, but also with memorizing regularities within the training data itself. Ravichander et al. (2021) illustrate the danger inherent to such a pursuit, showing that a probe can learn to decode linguistic properties from representations, despite the underlying model not having been exposed to the same properties during pre-training. They conjecture that, while "probing only indicates whether some property correlated with a linguistic property of interest is encoded in the [...] representation, it cannot isolate what the property might be, whether the correlation is meaningful, or how many such properties exist". Certainly, such insights relate to the notion of "spurious correlation", wherein two or more variables can be shown to be associated, despite not being causally related (Jo and Bengio, 2017; Shah et al., 2020). This has been shown to affect NLP tasks of various stripes, with NLI-based models being particularly vulnerable in correlating factors like sentence length, negation, and sub-sequence overlap with entailment labels (Gururangan et al., 2018; Poliak et al., 2018; McCoy et al., 2019).

Concerns about spurious correlations are naturally related to model selection — another critique raised by Belinkov (2022). Indeed, the prospect of probing models for linguistic properties hinges on the capability of the probing model itself. As such, a principled appraisal of various candidate models should be conducted so as to ensure that the selected model performs accurately, does not overfit, and can generalize accordingly. In theory, this entails an exhaustive search over (possibly innumerable) hyperparameters, such as model complexity, training regime, linearity, and general architecture. In practice, however, researchers are generally limited by experimental constraints (such as lack of access to computing resources) and therefore resort to "best practices" for model selection offered by previous work. Unfortunately, insights put forth by such studies are often conflicting in and of themselves. For example, while numerous studies urge researchers to employ simple, linear probes in order to demonstrate that properties are easily extractable (Alain and Bengio, 2017; Hupkes et al., 2018; Liu et al., 2019), others advocate for the use of complex probes that best optimize information theoretic measures like mutual information and pareto efficiency (Pimentel et al., 2020a,b). Further yet, the performance of the chosen probe must itself be considered with respect to numerous viable baselines. Kunz and Kuhlmann (2020) offer a taxonomy of thereof, and demonstrate various experimental settings (for example, number of training examples) wherein non-contextual baselines perform comparably

to fully-contextualized representations. Everything taken together, it becomes difficult to ascertain that the success or failure of a given probe is, in fact, conclusive, or if it is simply representative of the experimental variables employed in the study.

Paper II demonstrates how the aforementioned concerns manifest in practice when analyzing the behavior of Hewitt and Manning (2019)'s structural probe. We motivate our investigation of these issues by noting a potential caveat related to the aforementioned probing results: that they are reported in the context of the UD annotation scheme. One can argue that this is a subjective experimental choice, in that UD imposes a set of design principles that have been deemed as not representative of syntactic structure in the traditional theoretical sense (Gerdes et al., 2018; Osborne and Gerdes, 2019). As such, it remains an open question whether or not the structural probe will yield the same performance when tasked with decoding an alternative — yet equally plausible — tree structure from the same set of input strings.

To explore this question, we employ the Surface-Syntactic Universal Dependencies (SUD) framework, which departs from UD in its treatment of adpositions, copula, coordination, among other relations. In motivating SUD, Gerdes et al. (2018) conjecture that UD's prioritization of content word heads departs from traditional theoretic analyses, which assign function words and auxiliaries as the heads of dependencies and phrases. As such, SUD is presented as an alternative, *surface-syntactic* representation, which serves as a complement to the *deep syntax* captured by UD. Gerdes et al. (2018) offer an automatic conversion algorithm for UD, which we apply to the 13 treebanks studied in Paper II in order to obtain their respective SUD counterparts. We note that this results in a characteristically different set of treebanks, particularly in terms of graph properties. For example, while average dependency length decreases when converting from UD to SUD (3.14→3.0), tree height increases significantly (4.20→5.91). Also, we observe that, as expected, the average percentage of direct relations between content words (noun → noun, verb → noun) per sentence noticeably drops from 32% to 22%.

In terms of probing accuracy, the SUD tree structure proves to be slightly more difficult to decode than UD. Specifically, we observe that the linear mixture for BERT and ELMo yields 68 and 52 UAS, respectively, compared to UD's 72 and 55. This trend is likewise visible across layers, where the frameworks diverge in accuracy between BERT's 3rd and 12th layers. Here, the 7th layer shows the greatest disparity with 3 UAS in favor of UD. A similar trend persists for ELMo, with the 2nd layer also yielding a 3-point preference for UD. This is an interesting result, given that supervised parsers do not appear to exhibit such a preference when trained on UD and SUD treebanks. There, the performance gap amounts to less than 0.5 UAS in favor of UD, though this difference is not statistically significant.

On the surface, these results demonstrate that UD trees are easier to decode from BERT and ELMo's representations. By extension, we might conclude

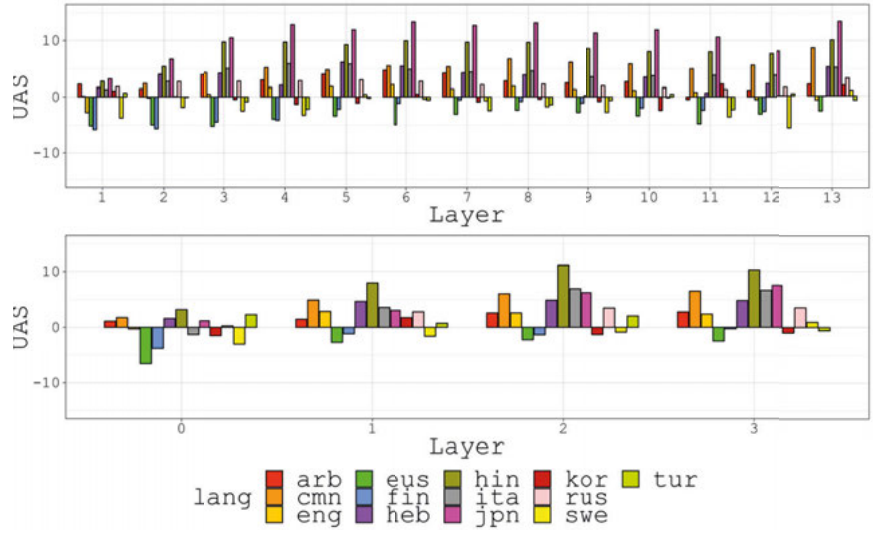

Figure 5.1. Difference in probing accuracy across UD (+) and SUD (−) for BERT (top) and ELMo (bottom), per layer and treebank.

that the syntactic structure captured by such models resembles the "deep syntax" of UD more than the "surface syntax" of SUD. However, in examining the difference in probing accuracy across treebanks, we notice a much more nuanced picture emerging. Specifically, we can observe from Figure 5.1 that while some treebanks exhibit a strong preference for UD in terms of probing accuracy (Japanese, Hindi, Chinese, Italian, Hebrew), others either prefer SUD (Basque, Finnish, Turkish), or show no preference at all (Korean).

In looking further into this phenomenon, we make note of a strong correlation ($\rho = 0.82, p < 0.001$) between the observed preference for UD exhibited by our treebanks and the difference in tree height between UD and SUD. We interpret this to mean that, the higher a tree becomes after conversion to SUD, the more difficult it is for the probe to decode it from a model's representations. This is most clearly evident for Japanese, which exhibits the strongest preference for UD overall, as well as the greatest increase in tree height. The SUD-preferring treebanks, on the other hand, exhibit a much weaker rate of tree height increase following the conversion to SUD. Lastly, Korean demonstrates no preference between the frameworks, which is reflected by the fact that it returns the smallest increase in height overall.

Though graph properties like tree height clearly bear influence on the performance of the structural probe, we make a further note of how they themselves may stem from the typological factors. To illustrate this, we observe that the three treebanks shown to prefer SUD (Basque, Finnish, Turkish) likewise share a typological property in common: agglutinative morphology. In other words, these languages rely on morphological case marking in order to express func-

tional relations, often foregoing the use of standalone function words. This property is corroborated by treebanks statistics, which show that these particular treebanks exhibit the shortest average dependency length, and the lowest average usage of adpositions per sentence. With in this mind, is of little surprise that these treebanks are least affected by the conversion to SUD, since their associated tree shape remains mostly unchanged. By extension, this allows for the language-specific properties that *are* affected by the conversion to be more easily decodable by the SUD probe — for example, the prolific usage of auxiliary verbs in Basque.

In addition to typological properties, we also show that annotation decisions can affect probe performance. This is especially relevant to Japanese and Korean, which exhibit outlier tendencies in our experiments. Regarding the Japanese GSD treebank, it has been previously noted that the tokenization scheme utilized therein resembles morphological segmentation, where inflectional affixes are split off from their nominal or verbal stems and re-attached as direct dependents (Omura et al., 2021). As a result, this particular treebank features a proliferation of functional elements, which ensures that the UD $\rightarrow$ SUD conversion results in a drastic increase in tree height. It follows that the structural probe, which is demonstrably affected by tree height, exhibits a clear-cut preference for UD in this case. In contrast to this, the Korean treebank employs a drastically different tokenization strategy, where tokens represent as larger chunks (Noh et al., 2018). As a result, content words comprise the majority of elements in this treebank, making functional elements comparatively rare. For this reason, the Korean treebank is hardly affected by the UD $\rightarrow$ SUD conversion, thus explaining why the probe shows no preference.

5.2 Issues with Attention

Similarly to probing, the prospect of interpreting attention weights has been at the center of much debate. This was arguably initiated by Jain and Wallace (2019), who argue that attention cannot serve as a faithful explanatory mechanism for the relative importance of inputs. To make their case, they first demonstrate that attention distributions in BiRNN models trained on various tasks do not correlate with other common measures of feature importance. Furthermore, they demonstrate that counterfactual attention distributions, including *adversarial* ones that resemble the original distributions as little as possible, tend to pose little-to-no effect on the model, which often generates the same prediction regardless. In a contemporaneous study, Serrano and Smith (2019) reach many similar conclusions regarding the interpretability of attention in text classification tasks. There, they show that removing the most highly attended-to input token bears a similar effect on the model as removing random tokens, and that gradient-based feature attribution methods are a much better indicator of input token saliency in this regard. To this end, such findings have

led some researchers in the field to advocate for alternative feature attribution methods in place of attention (Bastings and Filippova, 2020). However, numerous evaluations of these approaches have themselves yielded mixed results with regard to utility and consensus (Atanasova et al., 2020; Ding and Koehn, 2021; Bastings et al., 2022), thus putting the definition of faithfulness into further focus (Jacovi and Goldberg, 2020).

Wiegreffe and Pinter (2019) offer a counterpoint to the "attention as explanation" debate by revisiting the results reported in Jain and Wallace (2019). Primarily, they demonstrate that text-classification models tend to yield similar attention distributions when trained across numerous random seeds, indicating that such models arrive at similar "solutions" with respect to input token importance. In another experiment, they train an adversarial model that attempts to mimic a pretrained, base model's predictions, while simultaneously diverging as much as possible from its attention distributions. In applying the base and adversarial models' trained attention weights to a non-contextual classifier, they show that the base weights aid performance, while the adversarial weights degrade it. Consolidating these findings, Wiegreffe and Pinter (2019) argue that, while attention may not necessarily provide an exclusive *faithful* explanation for a model's decisions, it nonetheless offers a meaningful *plausible* one.

This debate is relevant to our investigation of syntax encoded via self-attention in that it offers a conceptual framework by which our results can be interpreted. Indeed, the tree decoding experiments outlined in Chapter 4 cannot, in principle, be described in terms of faithfulness. After all, the model which we investigate — multilingual BERT — is not trained on a task that presupposes the incorporation or prediction of tree structure (unlike parsing, for example). As such, the search for generalist or specialist heads amounts to seeking a plausible explanation for how the model employs attention in composing representations and, by extension, its output. To this end, we can indeed view our findings so far as suggestive of the fact that BERT does incorporate dependency syntax when computing self-attention weights (albeit, to a limited extent). However, to answer whether or not self-attention, as a standalone mechanism, is *capable* of faithfully encoding syntactic structure requires the the use of a model that is trained on a task for which such structure is explicitly necessary. Given such a model, a self-attention head can be said to generate faithful explanations if its weights align with the exact tree that the model is tasked with predicting.

To answer this question, we revisit the findings described in Section 4.2 through the lens of a dependency parser. Specifically, we fine-tune the aforementioned multilingual BERT encoder for dependency parsing, using Dozat and Manning (2017)'s biaffine algorithm. We train this parser on the concatenation of all PUD treebanks, so as to ensure that the trees we aim to decode will be guaranteed to have been seen by the model. In doing so, we hope to bypass the potential confounds associated with cross-treebank transfer, such

as imbalanced training data or tokenization mismatches. Following the fine-tuning process, we attempt to extract undirected, unlabeled dependency trees from each multilingual BERT attention head, via the same MST-based decoding procedure employed by Raganato and Tiedemann (2018) and Clark et al. (2019). We then compare the decoding accuracy returned by the parsing-tuned model with that of the pre-trained MLM.

In general, we observe a noticeable boost in decoding accuracy when fine-tuning multilingual BERT on a parsing-based objective. This amounts to an increase of over 20 points for most languages when accounting for the best-performing head, with Spanish yielding the highest boost overall: 50→77 UUAS. Interestingly, the same treebanks which posed issues for the pre-trained model (French, Japanese, Turkish, Chinese) are likewise unaffected by the fine-tuning procedure, returning little-to-no improvement over the adjacent-branching baseline. This leads us to believe that annotation or tokenization issues may be at play for these particular treebanks. Beyond this, we find that the fine-tuning procedure generally bears little effect on heads extracted from the lower layers of the model, all of which tend to behave similarly to their pre-trained counterparts. However, we observe a marked increase in decoding accuracy across heads in layers 10 and 11, indicating that the network begins to specialize for the parsing task in this region, thereby affecting the behavior of attention heads in the process.

These results imply that self-attention can be *steered* towards reflecting syntactic structure if the model's underlying objective calls for the incorporation of tree structure. By extension, they also indicate that, if such structure was useful to the BERT pre-training objective (masked language modeling), it would be transparently decodeable from self-attention. The results described in Section 4.2 indicate that this is not the case, given our inability to localize a generalist head across languages (in the same way that we can for the fine-tuned parser, at least). Beyond this, it is also important to acknowledge that self-attention does not correspond to a single parameter, but rather a combination of numerous model components (Vaswani et al., 2017). As such, we are likewise interested in the extent to which these constituent parts — namely, the key, query, value, and feed-forward parameters — influence the representation of syntactic structure in the fine-tuned model as a whole. To investigate this, we repeat the fine-tuning procedure, where we *freeze* all network parameters throughout training, except those corresponding to each of the aforementioned components. In other words, only the selected components are subject to gradient updates throughout training, while all other parameters represent the original pretraining task. We then compare the MST-based decoding accuracy of each of these ablated models to the original, pre-trained model, as well as the fully-tuned parser.

Overall, we observe a substantial decrease in decoding accuracy when compared to the fully-tuned network, corresponding to a difference of approximately 10 UUAS points across components. Interestingly, the Key compo-

nent returns the worst accuracy overall when its parameters are tuned in isolation, whereas the related Query component (with which it is composed) performs on par with all other ablated components. When tuned together, the Key and Query components yield comparable decoding accuracy to the standalone Query parameters, notably dropping in accuracy in the uppermost layer. We note that the Value parameters return the highest decoding accuracy across all ablated components, despite the fact that the Value representations are composed *after* the product of Key and Query components is scaled. This indicates that the Value component is able to extract the relevant syntactic information encoded in the self-attention matrix and propagate it up the network, affecting the Key and Query parameters, despite them being frozen.

5.3 Summary

In this chapter, we investigated RQ3 in the context of hidden state probing and attention head decoding. Regarding the former, we demonstrated that the choice of annotation framework bears a clear influence when probing BERT and ELMo for dependency structure across 13 typologically diverse treebanks. Crucially, however, we also noted that the observed performance disparity between UD and SUD is highly influenced by experimental variables employed in the study. These include the selection of the probing model, the typological characteristics of the languages under investigation, and annotation properties of their representative treebanks.

Regarding attention head decoding, we showed that the self-attention heads of fine-tuned BERT models can reflect syntactic structure when imbued with a supervised dependency parsing objective. To this end, we observed much higher decoding accuracies after fine-tuning than what we reported for the off-the-shelf, pre-trained model in Chapter 4. We interpreted this to mean that, if dependency tree structure was explicitly necessary for BERT's pre-training objective (masked language modeling), it would be transparently encoded in its attention heads. In addition, we showed that every self-attention component implemented in BERT (Keys, Queries, Values, Feed-Forward) is capable of encoding syntactic structure when fine-tuned in isolation, though the highest decoding accuracies are obtained when the network is trained in full.

6. Exploring and Characterizing UD Treebanks

In the previous two chapters, we experimented with the prospect of decoding syntactic structure — specifically, dependency trees — from the components of neural language models. Although our preliminary findings were encouraging, a careful analysis revealed that the interpretation of these results was confounded by a host of experimental factors. Prominent among these was the issue of *data*, which was employed as a means of eliciting the structure we hoped to investigate. Notably, our probing study revealed that treebank properties like tokenization scheme could bear a sizable influence on tree shape, irrespective of the widely understood properties of the underlying language, as in the case of Japanese and Korean. This, in turn, led to an apparent preference for one syntactic framework over another, despite the difference between them being, in this case, non-linguistic. Insights such as these highlight the complicated relationship between linguistic tasks and the data collected to elicit them. Unfortunately, it is often difficult to isolate properties that are characteristic of the former, yet independent of the latter — especially in typologically diverse settings where native speaker judgements are unavailable.

This chapter is focused on the data employed in the previously mentioned studies: UD treebanks. Specifically, we address the fourth and final research question:

RQ4 How can we characterize treebank properties that affect parser learnability, such as annotation quality or tokenization decisions?

We do so by exploring three different dataset analysis methods — dataset cartography (Swayamdipta et al., 2020), $\mathcal{V}$-information (Ethayarajh et al., 2022), and minimum description length (Perez et al., 2021). Specifically, we extend each method to the task of dependency parsing, using the biaffine parser of Dozat et al. (2017), trained on 88 different UD treebanks. In doing so, we find that each of these methods is able to shed a nuanced light on the inherent characteristics and idiosyncracies of treebanks, which would otherwise remain undetected by traditional, accuracy-based performance metrics.

6.1 Accuracy and Its Alternatives

The proliferation of neural language models like ELMo and BERT necessitated a comprehensive means of evaluation by which they could be reliably

compared, and through which state-of-the-art performance could be articulated. For a time, the GLUE benchmark (Wang et al., 2018) fulfilled this purpose, offering a set of seven complex NLP tasks that were motivated as assessing models' capacity for "language understanding". The models in question quickly exceeded the performance of the accompanying human baselines, prompting researchers to investigate the data itself and what exactly the models managed to learn from it. In many cases, it was shown that models came to rely on heuristic cues and shortcuts present in the data (yet invisible to the human observer), such that entire input sequences could be corrupted without drastically degrading performance (Gupta et al., 2021; Sinha et al., 2021b; Clouatre et al., 2022). This, in turn, prompted the community to consider more challenging benchmarks, new perspectives on crowd-sourced data, and alternative means of evaluating how models interfaced with the data on which they were fit (Nangia and Bowman, 2019; Wang et al., 2019; Bowman and Dahl, 2021).

Per the latter point, various new methods were introduced in order to provide a nuanced view of how models fared with the occasionally idiosyncratic properties of datasets (Swayamdipta et al., 2020; Kunz and Kuhlmann, 2021; Perez et al., 2021; Rodriguez et al., 2021; Vania et al., 2021; Ethayarajh et al., 2022). Much of this work was inspired by the shortcomings of accuracy-based metrics (such as precision, recall, and F-score) — the evaluation paradigm around which the majority of NLP tasks are centered. Indeed, although it is straightforward to calculate and interpret, accuracy leaves much to be desired when it comes to understanding model *behavior* with respect to a certain dataset. For example, it is most often reported with respect to one checkpoint of a model's entire training regime, which typically consists of numerous epochs and parameter updates. In honing in on one particular checkpoint (usually the best with respect to validation loss or accuracy), one cannot readily assess whether the model was easily fit on the data, or if training was stopped prematurely. Furthermore, in choosing the arg max over the output distribution, one inevitably loses information about it: was the model confident in making its prediction? Or was the distribution highly entropic? Also relevant is the train/test distinction: in evaluating on the latter, one can gauge a model's ability to generalize, but generally cannot assess the goodness-of-fit on the former, nor its sample efficiency. Likewise, accuracy cannot, in principle, adequately assess the *quality* of the training data: can the model learn from all instances therein? Or does the data contain a substantial amount of noise due to, for example, annotation inconsistencies? With this in mind, we provide a brief description of three accuracy-free dataset analysis methods below, each of which seek to mitigate the aforementioned shortcomings.

6.1.1 Dataset Cartography

One prominent accuracy-free method for exploring datasets was proposed by Swayamdipta et al. (2020), who called their approach *dataset cartography* (DC). Provided a training partition of a dataset, DC tracks the probabilities that a model assigns to the *gold label* for every instance therein. The mean and standard deviation for these probabilities — referred to as *confidence* (Conf) and *variability* (Var), respectively — is then calculated across a designated number of training epochs and plotted in a 2D diagram. Swayamdipta et al. (2020) refer to such plots as *data maps*, since they visualize the underlying dataset through the perspective of a particular model. They demonstrate that data maps tend to follow a familiar, boomerang-shaped distribution across datasets, with three distinct regions corresponding to *easy*, *hard* and *ambiguous* instances.

Swayamdipta et al. (2020) demonstrate that easy instances, characterized by high Conf and low Var, do not equip the model to generalize well — either in-domain (ID) or out-of-domain (OOD). Ambiguous instances (high Var), however, appear to be especially effective for this purpose: training on 33% of the data, sorted by high Var, yields higher OOD accuracy than training on the full data for several commonsense reasoning and textual entailment tasks. Furthermore, it appears that this capacity can be further expanded by swapping some samples in the ambiguous subset with easy instances, clarifying the role of the latter in optimization. Lastly, Swayamdipta et al. (2020) show that hard instances (low Conf, low Var) tend to correspond to dataset noise (in the form of annotation errors, for example), though they generally constitute a small percentage of the dataset and their removal does not yield significant improvements.

6.1.2 V-Information

An alternative approach for quantifying dataset difficulty is proposed by Ethayarajh et al. (2022), who leverage the concept of $\mathcal{V}$-information (henceforth V-Info). Introduced by Xu et al. (2020), V-Info is a framework for estimating the amount of information between random variables X and Y that is *usable* by a model in family $\mathcal{V}$ — for example, a sentiment classifier or syntactic parser. Here, *usability* is measured with respect to a corrupted form of the input (in other words, all tokens replaced with "_"), from which $\mathcal{V}$ must nonetheless attempt to predict Y — essentially a label-only baseline. High V-Info values indicate that the dataset contains much information that cannot be inferred by naive baselines, while values close to zero indicate that the model would not fare worse when picking a class at random.

Ethayarajh et al. (2022) propose numerous use cases of V-Info. Primarily, they demonstrate that V-Info can be used to compare different datasets with respect to a single model. For example, they report that MultiNLI (Williams et al., 2018) contains less BERT-usable information than SNLI (Bowman et al.,

2015), making it a more difficult dataset for the same task. Similarly, they show that V-Info can be measured with respect to corrupted or transformed versions of datasets, so as to isolate the influence of specific properties. To this effect, they report that token identity — isolated by shuffling input words — contains the most usable information for SNLI across all models. Lastly, they demonstrate that *pointwise* V-Info can be measured at the instance level with negative values bearing a strong correspondence to mislabeled examples.

6.1.3 Minimum Description Length

Minimum description length (MDL) (Rissanen, 1978) is another information-theoretic metric that has recently seen usage within NLP. Put briefly, MDL concerns the transmission of information (data) through a specified channel (a probabilistic model). Ideally, a model that is fit well on some data will learn to transmit — or *compress* — it using as few bits as possible. Blier and Ollivier (2018) propose online coding for measuring MDL: a technique which transmits the data and model without explicitly compressing the latter's parameters[1]. Effectively, online coding entails splitting the dataset into a designated set of partitions (or blocks) and measures the fit of the model on each successive partition. Computed this way, MDL expresses the ability of a model to generalize with respect to a dataset: models that learn efficiently from limited instances will yield shorter codelengths (lower MDL).

Perez et al. (2021) propose using MDL as a means of analyzing NLP dataset characteristics. To demonstrate its effectiveness, they first show that MDL can reveal the utility of subquestions in various question–answering tasks. Compared to employing the question in isolation, incorporating relevant subquestions often results in a further reduction of description length. Furthermore, they show that MDL is capable of isolating linguistic properties that are characteristic of certain datasets. For example, they report that the MDL for SST-2 (Socher et al., 2013) is dramatically *increased* when masking out adjectives in the input, as opposed to masking the same proportion of random words. They interpret this to indicate that adjectives are more prevalent in the SST-2 dataset than other parts of speech, which is intuitive given the sentiment analysis task which SST-2 represents. In a related experiment, they demonstrate that MDL can reveal the relative importance of word order across the datasets collected in GLUE. To this end, they show that randomly shuffling words in COLA (Warstadt et al., 2019) leads to a substantial increase in MDL with respect to the original word order, implying that the task is particularly sensitive to this property. In contrast, datasets corresponding to other GLUE tasks like STS-B, QQP, and RTE yield minimal difference in MDL between shuffled and original input.

[1] Unlike variational coding, for instance, which is likewise proposed in Blier and Ollivier (2018).

6.2 Treebanking Concerns

It should be noted that much of the aforementioned discussion pertains largely to classification tasks. In this thesis, however, our focus lies largely on treebanks and parsing, which represent a different class of problem: structured prediction. As such, it remains an open question whether or not the surveyed issues and methods proposed to address them are relevant solely to the former paradigm, or if they apply in a more general sense. Moreover, the GLUE suite — and most work centered around it — is monolingual, representing English in isolation. Our work, on the other hand, concerns numerous languages that were sampled with typological diversity in mind. In a sense, working with such data is more challenging, given that we lack direct native speaker insight for the majority of associated treebanks and cannot readily appraise the quality thereof. As such, the utility of the aforementioned dataset analysis methods becomes especially vital in our limited setting, as well as across the wide scope of UD overall.

In working with UD, it is important to note that, despite its impressive language coverage, not all treebanks are created equal. For example, though several treebanks have been natively annotated with respect to UD's official guidelines, many others had been migrated from treebanks in older schemes and converted accordingly — often by means of deterministic algorithms and heuristics (for example, see Bouma and van Noord (2017); Çöltekin et al. (2017)). Though such conversions are usually highly accurate, they sometimes require rigorous correction, and, other times, re-annotation (Türk et al., 2019) — assuming such issues have been identified in the first place. Compounded with conversion from existing schemes, varying interpretations of UD guidelines are likely to manifest across a globally distributed research community. Indeed, numerous annotation inconsistencies have been recognized across typologically similar languages (Han et al., 2020), as well as across treebanks for the same language (Aggarwal and Zeman, 2020; Dönicke et al., 2020). Absent an in-depth knowledge of a particular language, identifying such anomalies is difficult and time-intensive.

In the previous chapters, we demonstrated that such concerns are pertinent to our research. For example, the divergent tokenization schemes employed for Japanese and Korean treebanks bore a clear, likewise diverging effect on the probing accuracy for these languages. Furthermore, we observed consistently low performance for Turkish IMST across all experiments described so far, which raised questions about the nature of treebank as a whole. Here, we offer a brief description of the tokenization scheme applied to Japanese and Korean, as well as the automatic conversion process applied to Turkish, so as to illustrate the challenges posed for treebank annotation.

6.2.1 Tokenization Principles

Japanese GSD (McDonald et al., 2013) employs the *short word unit* (SUW) tokenization scheme (Murawaki, 2019), which itself operationalizes the minimum unit standard (Den et al., 2008): a Japanese lexicon categorized by *word types*, such Chinese-origin words, non-Chinese loan words, and numerals. By default, this scheme classifies any candidate word as a SUW if it corresponds to an item in the minimum unit lexicon. If a candidate word can be split into two minimum units of the same *type*, it is likewise considered a SUW (though this does not apply to splits of more than two units). SUWs thus comprise both function and content words in UD, with adpositions, auxiliary verbs, and affixes all bearing the same categorization as words. Such an approach carries the attractive property of minimizing a corpus' vocabulary size and type–token ratio, which can aid representation learning and parsing. Conversely, it likewise leads to a proliferation of functional elements that are arguably morphological in nature, which can, by consequence, inflate parsing accuracy.

Omura et al. (2021) propose an alternative tokenization scheme called *long word unit* (LUW), which they argue to be a better fit for UD's syntactic word criterion. Rather than building upon minimum units, they opt instead to split the *bunsetsu* unit: a base phrase comprised of a "compound independent word" and its dependents (affixes, adpositions, auxiliaries). With this in mind, Omura et al. (2021) define LUWs to be the individual constituents that comprise a bunsetsu. Given this formulation, LUW more accurately represents the agglutinative nature of Japanese, where many dependent elements that would otherwise be classified as individual SUWs are now represented as parts of the word they modify. Omura et al. (2021) re-tokenize GSD via the LUW scheme, showing that its application amounts to an expected increase in type-token ratio and a decrease in number of tokens. Furthermore, they demonstrate that, when using gold tokenization, there is no significant difference in parsing accuracy between SUW and LUW.

Unlike Japanese, where sentences are represented as a continuous string of characters, Korean employs whitespace to delimit *eojeols*. These units typically comprise a noun or verb stem as well as any corresponding postpositions and inflectional/derivational particles — making them similar to the aforementioned bunsetsu. The internal complexity of the eojeol makes it difficult to categorize it as a standalone syntactic unit, and Noh et al. (2018) demonstrate how this poses additional problems for the assignment of part-of-speech tags and dependency relations. For example, they show that the GSD treebank unilaterally tags eojeols containing verbal elements as `VERB`, regardless of the type of ending applied to the verb stem. This results in a broad, coarse-grained miscategorization of adjectival, adverbial, and auxiliary elements, with 97% of the latter constructions requiring revision per the authors' perspective.

Such issues are endemic of the eojeol's general opacity, which has led various researchers to propose alternative tokenization strategies. For example,

Chen et al. (2022) propose to apply morphological segmentation at the eojeol level, where morphemes and particles are segmented as isolated units, yet remain direct dependents of the stem. The authors demonstrate that this approach resolves the opacity issue, affording a more straightforward assignment of part-of-speech tags and dependency relations, which leads to higher parsing accuracies. However, in treating morphemes as standalone units, this approach likewise fails to accurately represent the agglutinative nature of Korean, which makes it privy to the same problems posed by SUW for Japanese.

6.2.2 Conversion Artefacts

Aside from tokenization, another prevalent issue faced by UD concerns the conversion of existing dependency treebanks to the framework. Typically, this is accomplished either by manual re-annotation, or by an automated conversion process followed by human validation. Turkish IMST (Sulubacak et al., 2016a) belongs to both categories, having been converted to UD from a language-specific framework, which itself was a re-annotation of an older Turkish treebank: METU-Sabancı (MST) (Oflazer et al., 2003).

The creation of the IMST treebank was motivated by the various annotation inconsistencies present in the original MST treebank. Sulubacak et al. (2016a) categorized these issues as pertaining to semantic incoherence, hierarchy and overlap, annotation ambiguity, and a reliance on omissible tokens. In the manual re-annotation effort, Sulubacak et al. (2016a) reduced the dependency label set from 24 to 16, and showed that their proposed revisions yielded a 10-point LAS increase for MaltParser.

The IMST treebank would later undergo its own conversion process to the UD framework, as described in Sulubacak et al. (2016b). Similarly to Japanese and Korean, the chief challenge in this process entailed segmentation: namely, the application of UD's syntactic word criterion to Turkish — also an agglutinative language. Doing so necessitated the removal of the inflectional group (IG) formalism applied by both MST and IMST for segmenting morphosyntactic units along Turkish words' derivational boundaries. In its place, Sulubacak et al. (2016b) proposed several changes, including the splitting of some derivational morphemes to act as dependents of their associated stem (behaving similarly to clitics), and the merging of others that were deemed to be insufficiently productive (*lexicalization*, in their words). On top of this, Sulubacak et al. (2016b) proposed UD mappings for language-specific part-of-speech tags, morphological features, and dependency relations, making note of various edge cases where this was not straightforward (for example, assigning case markers to verbal heads). After implementing the segmentation and mapping adjustments, they applied a series of post-processing steps in order to ensure that the conversion yielded valid dependency trees: for example, assignment of root tokens, or removal of cycles.

Though IMST has often featured as the representative Turkish treebank in a variety of typological and parsing studies, the numerous conversions leading up to its present form have led several researchers to question the annotation consistency present therein. To this end, Türk et al. (2019) outline various problems associated with the present IMST UD treebank, which they attribute to the automated conversion procedure described above. For example, they show that an ambiguity between an accusative case marker and a possessive suffix leads to a systematic misclassification of `nsubj` relations as `obj`. By their measure, this particular mismatch occurs 276 times throughout the corpus and is a product of an incorrect morphological analysis, which was performed automatically by Sulubacak et al. (2016b). Türk et al. (2019) also introduce 10 previously unused relation types, including `advcl` and `iobj`. In total, their re-annotation effort amounts to manually revising 5,635 sentences, which yields increased parsing accuracies over the updated treebank (+3 UAS for transition-based and graph-based parsers).

6.3 Exploring UD Treebanks

In the previous sections, I outlined the various issues concerning the use of accuracy metrics in understanding NLP datasets, as well as three alternative methods proposed to mitigate them. In addition to this, I provided a description of the challenges posed for the UD framework — namely tokenization and conversion — as well as how these factors pertained to the treebanks employed in our experiments. With this in mind, Paper IV details the application of the three aforementioned dataset analysis methods to the UD suite, so as to better understand the inherent characteristics of the treebanks that comprise it. More specifically, we extend dataset cartography, $\mathcal{V}$-information, and minimum description length to the dependency parsing task, evaluating the behavior of Dozat and Manning (2017)'s biaffine parser as it is trained on 88 UD treebanks.

In terms of the dataset cartography metrics, we find that both Conf and Var — when interpreted as complements to each other — are capable of painting a nuanced picture of how *easy* or *hard* treebanks might be to parse. Specifically, we find that Japanese GSD ranks among the *easiest* treebanks to parse (along with English Atis and Hindi HDTB), returning the 2nd-highest average Conf and 4th-lowest Var. We corroborate this by zooming in on the Japanese GSD data map (Figure 6.1, left), where we observe that nearly 70% of all tokens in the treebank lie in the easy-to-parse region — e.g. Conf ≥ 0.95 and Var ≤ 0.1. This corroborates the findings of Nivre and Fang (2017), who attribute the high parsing accuracies associated with this treebank to the proliferation of easy-to-parse, functional elements present therein.

On the other end of the spectrum, we observe that Turkish IMST ranks among the *hardest* to parse treebanks (along with Uyghur UDT and Vietnamese

VTB), returning the lowest overall Conf. Interestingly, unlike UDT and VTB, it does not rank among the top-10 highest Var treebanks, indicating that treebank size – doubtless a factor for UDT and VTB (the 2nd and 3rd smallest treebanks overall) — is not the main explanation here. Indeed, in analyzing the Turkish IMST data map, we observe that nearly 10% of its tokens reside in the *hard-to-learn* region — e.g. Conf ≤ 0.25 and Var ≤ 0.1. This is more than double the density of hard-to-learn points in UDT and VTB, which confirms to the potential conversion issues described by Türk et al. (2019).

In measuring V-Info across treebanks, we observe an entirely different ranking than what was returned by the dataset cartography metrics. Specifically, we note that Japanese GSD, which was ranked among the easiest to parse treebanks by DC, returns the 30th lowest V-Info score (out of 88). We observe a similar effect for English Atis, which drops further to 11th. Turkish IMST, meanwhile, remains near the bottom, yielding the 4th-lowest V-Info in aggregate. On the other end of the spectrum, we make note of other top-10 "easy-to-parse" treebanks (per Conf and Var) among the top 3: Latin LLCT, Romanian SiMoNERo, and Catalan AnCora. Provided these observations, we can interpert V-Info as a means of simultaneously penalizing regularity and stochasticity in data. In other words: homogenous, single-genre treebanks with limited vocabularies (such as English Atis) are likely to produce low V-Info scores, given the uniformity of syntactic structures present therein. Furthermore, treebanks with potential annotation inconsistencies (such as Turkish, where 10% of tokens are "unlearnable") will likewise yield low overall V-Info, with many of the parser's decisions amounting to random "guesses". As an illustration of this, refer to Figure 6.1 (right), where arc-level V-Info scores the three lowest scoring treebanks are overwhelmingly distributed around 0, manifesting as a density "spike". Conversely, the V-Info scores for the three highest scoring treebanks are much more evenly distributed, resembling a normal distribution. To this end, we surmise that consistently annotated treebanks with diverse vocabularies and varied usage of syntactic structures are likely to return high V-Info, given that the baseline parser will have difficulty learning such patterns.

For MDL, we find that the aforementioned "easy" treebanks (Japanese GSD, English Atis, Turkish Atis) likewise yield the lowest scores. This suggests that they are the easiest to compress, or, alternatively, that they are the most sample efficient. Given the nature of these treebanks, such findings align with our observations above. In contrast to this, we observe a new set of treebanks appearing towards the high end of the MDL score spectrum, with Finnish TDT, Chinese GSD, and Latin PROIEL rounding out the top 3. Intuitively, a high MDL in the case of parsing might suggest that the model is exposed to a larger diversity of token types during training, which could hinder it in learning various types of dependencies. To this end, we observe that the aforementioned treebanks — as well as others yielding high MDL (Estonian, Russian, Turkish) — tend to be morphologically rich. Interestingly, Korean GSD also ranks highly in this regard, indicating that the segmentation issues discussed in Sec-

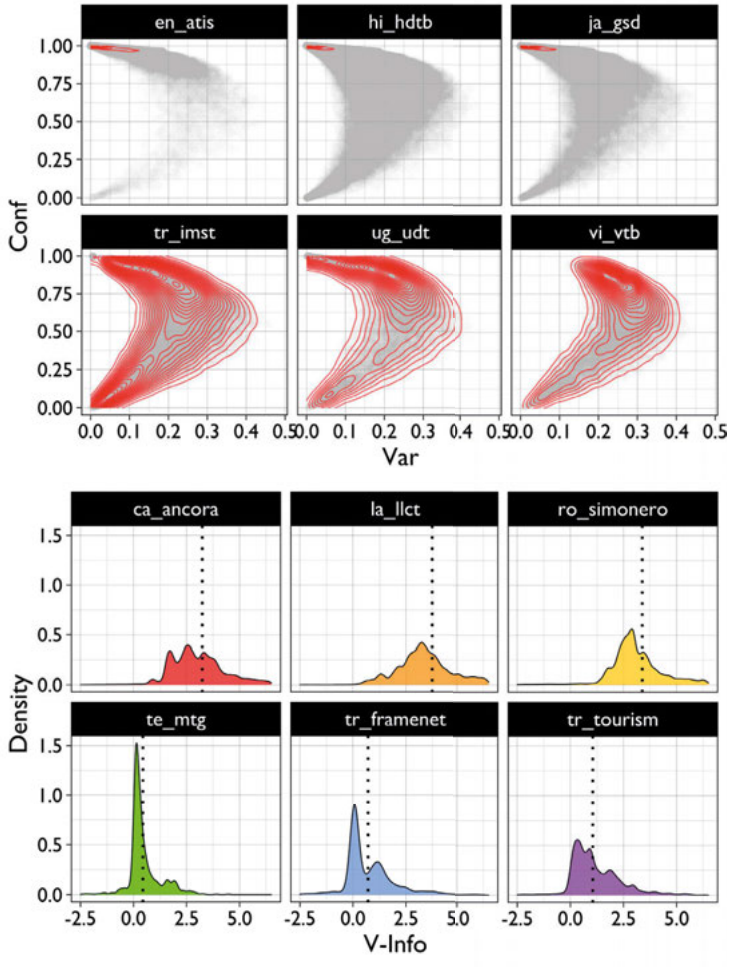

Figure 6.1. Top: Data maps for the three easiest (top) and hardest (bottom) to parse treebanks, according to Conf (y-axis) and Var (x-axis). Points correspond to individual arcs in the training set; red overlay represents a 2D point density. *Bottom*: Arc-level V-Info score distributions for the top-3 highest (top) and lowest (bottom) scoring treebanks.

tion 6.2.1 bear an influence in the representation of the morphology of the language. In an attempt to quantify the correspondence between a treebank's attested morphological complexity and its MDL, we compute a series of proxy metrics, including type-token ratio (TTR), number of feature types, and feature entropy. We find that each of these metrics yield strong correlations with MDL, with TTR being the highest ($\rho = 0.58, p < 0.001$). We interpret these findings to mean that MDL is highly influenced by vocabulary usage, which itself follows from typological (morphology) or annotation (tokenization) factors.

6.4 Summary

In this chapter, we investigated RQ4 in the context of Universal Dependencies. Specifically, we extended the dataset cartography (Swayamdipta et al., 2020), $\mathcal{V}$-information (Ethayarajh et al., 2022), and minimum description length (Perez et al., 2021) methods to the structured prediction scenario, training Dozat et al. (2017)'s parser on a subset of 88 treebanks (including the 13 treebanks discussed in Chapters 3, 4, and 5). We found that each of these methods provided nuanced perspectives on each surveyed treebank that would otherwise be obscured by accuracy-based measures. To this end, we found that Japanese GSD appeared among the easiest to parse and the most sample efficient treebanks, due to the tokenization strategy applied therein. Conversely, Turkish ISMT ranked consistently among the hardest to parse treebanks, with 10% of its training data proving impossible to fit by the parser. We interpreted this to indicate that this particular treebank contains annotation inconsistencies arising from the numerous conversion process it has undergone (Sulubacak et al., 2016b; Türk et al., 2019). Overall, such insights resonate with the findings reported in previous chapters, demonstrating the need for alternative evaluation metrics and other means of understanding how models interface with the data on which they are trained.

7. Conclusion

Throughout this dissertation, I investigated the extent to which syntactic knowledge is — or can be — captured by neural language models. I explored this question from the perspective of dependency parsers — structured prediction models specialized for predicting syntactic structure. Doing so allowed me to compare self-supervised language models to fully supervised parsers, which were trained on the exact syntactic structure we hoped to elicit. To this end, I were able to articulate the utility of language model features in downstream syntactic parsing, the differences in how language models and parsers encode syntax, and the influence treebank data can bear on the interpretation of our results. Ultimately, I was able to address the research questions posed at the beginning of this thesis, which I revisit below.

7.1 Research Questions

The research questions posed in Chapter 1 were as follows:

RQ1 How do representations from pre-trained language models affect the behavior of neural dependency parsers?

We find that contextualized embeddings extracted from pre-trained language models can provide a substantial boost to parsing accuracy. This applies to both transition-based and graph-based parsers, with the former enjoying greater improvement with respect to a static embedding baseline. Moreover, we observe that the error profiles associated with each model become virtually identical, equalizing the perceived pros and cons of either paradigm as observed in the pre-neural era.

RQ2 To what extent is syntax encoded in language model components, and how does this vary across different languages?

We show that a structural probe can decode UD-based dependency structure from the hidden states of neural language models. Overall, BERT strongly outperforms ELMo in terms of probing accuracy, with its intermediate layers (6, 7, 8) seemingly encoding the most syntactic information. Likewise, we observe considerable variation across the 13 surveyed languages, which suggests that typological or treebank factors

can affect probing results.

In addition to hidden state representations, we also show that dependency structure can be partially decoded from the attention weights of Transformer-based models. We demonstrate that such structure is distributed across the network, with several attention heads specializing in select dependency relations. This trend generally persists across most languages (of 20), though several languages fail to outperform basic positional baselines. In general, though we fail to localize a generalist head that can perform holistic parsing over full sentences, we nonetheless show that trees can be consistently extracted with an accuracy above that of a right-branching baseline.

RQ3 How is our understanding of language model components affected by various experimental variables, such as the choice of alternative syntactic frameworks or learning objectives?

We find that experimental variables bear a sizeable influence in the interpretation of the aforementioned results. Regarding the probing experiments, we show that UD trees can be decoded more accurately in aggregate than those belonging to SUD — an alternative, parallel syntactic framework. We further demonstrate that the preference for UD or SUD varies highly per language, with Japanese, Hindi, and Italian yielding higher scores for the former, and Basque, Finnish, and Turkish preferring the latter. Ultimately, we find that decoding accuracy for either framework is correlated with graph properties such as tree shape (to which the probe appears sensitive), which are themselves influenced by typological properties and treebank annotation factors.

Regarding the attention decoding experiments, we find that fine-tuning BERT with a dependency parsing objective leads to considerable gains in tree decoding accuracy. This mostly applies to the higher layers of the network, where we observe attention heads to reflect the exact tree structure on which the parser was trained.

RQ4 How can we characterize treebank properties that affect parser learnability, such as annotation quality or tokenization decisions?

We show that metrics like dataset cartography, $\mathcal{V}$-information, and minimum description length can reveal interesting properties of treebanks that would otherwise remain obscured by accuracy-based measures. Dataset cartography, for example, can pinpoint training samples that are either trivial or impossible for a parser to learn. Likewise, $\mathcal{V}$-information can express how a parser performs with respect to a

naive baseline, highlighting samples that require generalization beyond random guessing and memorization. Finally, minimum description length can reveal the sample efficiency of a treebank, which is related to morphological complexity and vocabulary usage.

Taken together, our findings suggest that, while language models' syntactic knowledge can be operationalized by means of downstream tasks, clarifying the nature of how it is encoded is a challenging endeavor. Indeed, the features extracted from language models are undoubtedly beneficial for supervised parsing. Likewise, it is clear that the components of Transformer-based models like BERT *can* be steered to reflect syntactic structure, provided that the fine-tuning objective is of the same nature. Such insights reflect the richness of language model representations, as well as the flexibility of the Transformer architecture and the fine-tuning paradigm. However, in the absence of a syntax-oriented objective function, isolating the components which track syntax remains difficult. Indeed, this is mostly due to the highly distributed and non-linear nature of such models, where hundreds of millions of parameters are optimized in unison. The language modeling objective is also relevant in that, unlike parsing, it does not impose any structural restrictions on what token must be predicted — simply that it must be the most *probable* given the context. As such, though methods like probing can demonstrably offer a glimpse into how vestiges of syntax are reflected in select model components, they cannot provide a comprehensive picture of how syntax is operationalized with respect to each individual parameter — or even if it needs to be. Beyond this, our search for syntax is further complicated by the representations chosen to elicit its abstract nature, as well as experimental factors like model and data selection.

7.2 Beyond Parsing

As noted above, the experiments presented in this dissertation were largely conveyed through the dependency formalism. Among other advantages, doing so allowed us 1) to envisage the utility of language model in an established, well-studied NLP task like dependency parsing, and 2) to make use of the expert-annotated, multilingual syntactic data collected in Universal Dependencies. Though this framing was generally informative, it was likewise bound by various methodological constraints. Chief among these was dependency's formal restrictions on what can constitute a well-formed syntactic tree — in other words, that relations manifest directly between words and that such relations are asymmetrical. As such, we did not consider other syntactic formalisms, such as constituency, which would require alternative experimental methodologies, evaluations, and data. In addition to this, the focus of our experiments was directed towards individual model components, such as hidden

state representations and self-attention heads. We did not work with language model output directly, and were therefore unable to say whether such models were even capable of generating grammatical utterances in a human-like manner.

In light of these concerns, Paper V situates our work within the broader scope of the interpretability literature. There, we outline three distinct paradigms employed by researchers as a means of eliciting the syntactic capabilities of language models (or lack thereof). One such paradigm is probing, which we discuss at length in Chapters 4 and 5[1]. Another is *targeted syntactic evaluation* (TSE), which is broadly concerned with comparing the probabilities assigned by language models to grammatical and ungrammatical utterances (Linzen et al., 2016; Gulordava et al., 2018; Marvin and Linzen, 2018). Lastly, we consider *downstream NLU evaluation*, which seeks to measure how models perform on NLU benchmarks after being deprived of (or, alternatively, imbued with) syntactic structure (Gupta et al., 2021; Sinha et al., 2021a,b). In surveying the research comprising these three paradigms, we observe a broad range of insights that, at times, lead to contradictory conclusions with respect to the question: *do language models capture syntax?* In an attempt to reconcile these conflicting insights, we propose a set of distinctions that we deem important toward the study of syntax in language models.

Primarily, we stress that coding properties, in themselves, do not represent syntax as a whole. Though a model may exhibit sensitivity to ungrammatical inflections, or, alternatively, an insensitivity to word order when solving tasks downstream, we must avoid conflating select *instantiations* of hierarchical structure with a model's acquisition of "syntax" as a holistic system. This likewise applies to syntactic representations (such as Universal Dependencies) which offer a perspective on how sentences are hierarchically organized, but themselves are privy to theoretical preferences and formal constraints. Such concerns tie more broadly into the general question of methodology, and whether or not experimental factors bear a confounding influence on the observed results. In Paper II, we showed that this can indeed occur, as performance of the probe appeared to correlate with underlying properties of the UD and SUD representations that were not necessarily linguistic in nature. Depending on our treatment of such factors, the answers we attribute to our research questions could vary drastically. To this end, it is worthwhile to ponder whether asking if language models *can*, *do*, or *need to* learn syntax carry different implications as research questions.

We conclude Paper V by acknowledging the inherent difficulty associated with the search for syntax. Indeed, we must recall that, after all, syntax is an

[1]Specifically, Paper V employs Belinkov (2022)'s taxonomy of parametric and non-parametric probing. The structural probe employed in Paper II falls under the former category and attention decoding (Paper III) under the latter.

abstract concept. Though its traces appear in the surface form of sentences, debates about its exact nature, representation, and learnability rage on to this day. In this light, studying syntax through the lens of language models offers numerous advantages. Among other things, language models provide an interface through which the likelihood, internal representation, and learning conditions of syntactic constructions can be studied. In an ideal scenario, such insights can complement studies involving human participants, potentially shedding light on the computational aspects of our own language faculty. However, in order for this undertaking to bear any empirical value, we must apply the same rigor to the computational study of syntax as we do to other fields of scientific inquiry. In the least, doing so should entail conducting a multi-dimensional evaluation of results, a principled appraisal of linguistic data, and a conscientious investigation of all relevant methodological factors and confounds. If implemented correctly, this paradigm has the potential to provide empirical grounding for explanations that were too long shrouded in theory and reinvigorate the study of syntax.

7.3 Future Work

Throughout this thesis, our focus was primarily placed on contextual embedding and fine-tuning approaches. We have demonstrated the efficacy of such methods, showing, for example, how the bidirectional nature of models like ELMo and BERT imbues them with rich feature representations, which can serve as an excellent basis for downstream, supervised tasks. In the present day, however, the NLP research community has turned its attention away from bidirectional models and towards *large language models* (LLMs) — autoregressive models built upon the Transformer's decoder component (Brown et al., 2020; Chowdhery et al., 2022; Smith et al., 2022; Zhang et al., 2022). LLMs are typically several orders of magnitude larger than BERT-base (the largest model surveyed in this thesis), both in terms of parameters and training data: for example, Google's PaLM model contains 540 billion parameters (BERT-base: 110 million) and was trained on approximately 780 billion tokens (BERT-base: 3.3 billion) (Chowdhery et al., 2022). The increased scale of such models has been shown to not only lower test set perplexity (Kaplan et al., 2020), but also to give rise to numerous groundbreaking phenomena (Wei et al., 2022a,b). Chief among these is *in-context learning*, in which LLMs can generate correct predictions for a task after being conditioned on only a few training examples (Brown et al., 2020). Crucially, this setup is encoded entirely in natural language, wherein a model accepts a conditioning context — a string typically comprising of *exemplars* (input-output pairs for training) and a *prompt* (the input sequence for which an output must be generated) — and generates an output string corresponding to the correct class. Certainly, this is a far cry from the supervised learning

paradigm discussed in this thesis, in that it 1) does not presuppose the use of a task-specific model, 2) does not require fine-tuning or gradient updates, and 3) can generalize via a small number of exemplars instead of the entire training set (also referred to as few-shot learning).

The advent of LLMs presents interesting implications for investigating the syntactic knowledge of neural language models. Out of the three methodologies outlined in Paper V, the probing paradigm likely carries the most dubious application to LLMs. Indeed, the concerns around probing discussed throughout this thesis become increasingly salient in the LLM scenario, as such models are often scaled to more than 100 billion parameters[2]. Therefore, such high parameter counts make it increasingly challenging to characterize and localize the behavior of a model's components with respect to a given linguistic property, and the assumption that this can even be done with a lightweight linear model at such scale is questionable. Conversely, TSE seems to be the most readily applicable to the LLM scenario — provided that conditional probabilities can be obtained for a given model. Indeed, benchmarks like BLiMP (Warstadt et al., 2020) or SyntaxGym (Gauthier et al., 2020) can continue to serve as gauges for the overall fluency of LLMs, especially as they are scaled to increasing sizes. To this end, Liang et al. (2022) demonstrate that the largest variant of the GPT-3 model (175B parameters) yields the best overall Exact Match (EM) accuracy across the BLiMP suite, while its smaller counterparts fare considerably worse. In addition to this, TSE benchmarks can aid in isolating the linguistic phenomena that remain problematic for LLMs, despite their otherwise demonstrable fluency. On this note, Liang et al. (2022) show that state-of-the-art instruction-tuned models such as InstructGPT (Ouyang et al., 2022) appear to struggle in producing irregular morphological forms across BLiMP. In this specific scenario, investigating such failure cases in more detail can shed light on the exact nature of the "syntax" internalized by such models, and how techniques like instruction tuning can influence syntactic generalization.

Beyond TSE, downstream NLU evaluation likewise carries considerable potential for articulating the syntactic capabilities of LLMs. In the case of fine-tuning, it has been shown that BERT and similarly-sized models are often insensitive to coding properties (for example, word order) when making predictions at test time (Gupta et al., 2021; Sinha et al., 2021a,b; Clouatre et al., 2022). While this has lead some researchers to surmise that such models do not process language in the same hierarchical manner that humans do (Sinha et al., 2021b), others have instead argued that such behavior is related to the downstream task in question (Clouatre et al., 2022). In support of the latter argument, numerous studies have demonstrated that models trained (or fine-tuned on) tasks like Natural Language Inference (NLI) (Bowman et al., 2015; Williams et al., 2018) had come to rely on various heuristics in order to make

[2]The 540B PaLM model, for example, contains 118 layers, 48 heads, and a hidden state dimensionality of 18432

predictions (for example, sub-sequence overlap or sentence length) — thereby challenging the notion that supervised learning can assess a model's "understanding" (Gururangan et al., 2018; Poliak et al., 2018; McCoy et al., 2019). The in-context learning paradigm presents an interesting twist to this setup, however, in that it limits all spurious correlations to what is observed in the provided exemplars. Likewise, it does not require any parameter updates. With this in mind, it would be interesting to see whether or not the (in)sensitivity of models to linguistic form likewise persists within in-context learning, and to which tasks, models, or settings this might apply. Specifically, word order would be a particularly compelling setting to revisit, in order to gauge models' robustness to permutations thereof and the extent to which this corresponds to human comprehension (Mollica et al., 2020). Beyond this, studying in-context learning from a syntactic perspective could potentially aid in demystifying the puzzling findings yielded by recent work: namely, that models can produce accurate predictions even when exemplars are semantically irrelevant (Min et al., 2022; Webson and Pavlick, 2022; Wei et al., 2023).

7.4 Final Remarks

It is not controversial to say that, in the present moment, the field of NLP remains captivated by language models. This fascination does not show many signs of waning, given the rapid pace of innovation and, consequently, adoption. Indeed, as language models garner everyday usage in the form of interactive chatbots and content generators, it becomes critical for us — as researchers, developers, and end users — to understand their behavior. To this end, I hope that this thesis adequately illustrated the difficulty inherent to the task of "interpretation" and "understanding". Likewise, I hope to have conveyed the considerable nuance that is essential to this undertaking, as well as the importance of careful experimental design and analysis. Though the experiments described here pertain mostly to an older class of models, I hope that they hold relevance for the current state-of-the-art and beyond. After all, there remains so much to learn.

Bibliography

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, Toulon, France. OpenReview.net.

Akshay Aggarwal and Daniel Zeman. 2020. Estimating POS annotation consistency of different treebanks in a language. In *Proceedings of the 19th International Workshop on Treebanks and Linguistic Theories*, pages 93–110, Düsseldorf, Germany. Association for Computational Linguistics.

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Workshop Track Proceedings*, Toulon, France. OpenReview.net.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online. Association for Computational Linguistics.

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 166–170, New York City. Association for Computational Linguistics.

Giuseppe Attardi and Massimiliano Ciaramita. 2007. Tree revision learning for dependency parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 388–395, Rochester, New York. Association for Computational Linguistics.

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, and Joseph Turian. 2009. Accurate dependency parsing with a stacked multilayer perceptron. *Proceedings of EVALITA*, 9:1–8.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, San Diego, CA, USA.

Jasmijn Bastings, Sebastian Ebert, Polina Zablotskaia, Anders Sandholm, and Katja Filippova. 2022. "Will you find these shortcuts?" a protocol for evaluating the faithfulness of input salience methods for text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 976–991, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jasmijn Bastings and Katja Filippova. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155, Online. Association for Computational Linguistics.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, MIT Press.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Léonard Blier and Yann Ollivier. 2018. The description length of deep learning models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, pages 2220–2230, Montréal, Canada. Curran Associates, Inc.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, MIT Press.

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

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R. Bowman and George Dahl. 2021. What will it take to fix benchmarking in natural language understanding? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4843–4855, Online. Association for Computational Linguistics.

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

Eric Brill, Jimmy Lin, Michele Banko, Susan T. Dumais, and Andrew Y. Ng. 2001. Data-intensive question answering. In *Proceedings of The Tenth Text REtrieval Conference, TREC 2001*, volume 500-250 of *NIST Special Publication*, Gaithersburg, Maryland, USA. National Institute of Standards and Technology (NIST).

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan,

Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, Online. Curran Associates, Inc.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

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

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, Elsevier.

Yige Chen, Eunkyul Leah Jo, Yundong Yao, KyungTae Lim, Miikka Silfverberg, Francis M. Tyers, and Jungyeul Park. 2022. Yet another format of Universal Dependencies for Korean. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5432–5437, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

N. Chomsky. 1956. Three models for the description of language. *I.R.E. transactions on information theory*, 2(3):113–124, The Institute of Radio Engineers, Inc.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David

Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Jayeol Chun, Na-Rae Han, Jena D. Hwang, and Jinho D. Choi. 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).

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Louis Clouatre, Prasanna Parthasarathi, Amal Zouaq, and Sarath Chandar. 2022. Local structure matters most: Perturbation study in NLU. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3712–3731, Dublin, Ireland. Association for Computational Linguistics.

Çağrı Çöltekin, Ben Campbell, Erhard Hinrichs, and Heike Telljohann. 2017. Converting the TüBa-D/Z treebank of German to Universal Dependencies. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 27–37, Gothenburg, Sweden. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 400–407, Salvador, Brazil. ACM.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308, MIT Press.

Yasuharu Den, Junpei Nakamura, Toshinobu Ogiso, and Hideki Ogura. 2008. A proper approach to Japanese morphological analysis: Dictionary, model, and evaluation. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language

Resources Association (ELRA).
Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
Shuoyang Ding and Philipp Koehn. 2021. Evaluating saliency methods for neural language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5034–5052, Online. Association for Computational Linguistics.
Tillmann Dönicke, Xiang Yu, and Jonas Kuhn. 2020. Identifying and handling cross-treebank inconsistencies in UD: A pilot study. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 67–75, Barcelona, Spain (Online). Association for Computational Linguistics.
Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, Toulon, France. OpenReview.net.
Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.
Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.
Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.
Jack Edmonds et al. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.
Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, Copenhagen, Denmark. Association of Computational Linguistics.
Jason M. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. *Advances in probabilistic and other parsing technologies*, pages 29–61, Springer.
Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211, Wiley Online Library.
Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2022. Understanding dataset difficulty with $\mathcal{V}$-usable information. In *International Conference on*

Machine Learning, ICML 2022, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008, Baltimore, Maryland, USA. PMLR.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 134–139, Berlin, Germany. Association for Computational Linguistics.

Richard Futrell, Kyle Mahowald, and Edward Gibson. 2015. Large-scale evidence of dependency length minimization in 37 languages. *Proceedings of the National Academy of Sciences*, 112(33):10336–10341, National Acad Sciences.

Jon Gauthier, Jennifer Hu, Ethan Wilcox, Peng Qian, and Roger Levy. 2020. SyntaxGym: An online platform for targeted evaluation of language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 70–76, Online. Association for Computational Linguistics.

Kim Gerdes, Bruno Guillaume, Sylvain Kahane, and Guy Perrier. 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.

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

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pages 959–976, Mumbai, India. Indian Institute of Technology Bombay.

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

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2015. An efficient dynamic oracle for unrestricted non-projective parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 256–261, Beijing, China. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. 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 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Ashim Gupta, Giorgi Kvernadze, and Vivek Srikumar. 2021. BERT & family eat word salad: Experiments with text understanding. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh*

Symposium on Educational Advances in Artificial Intelligence, EAAI 2021 2021, pages 12946–12954, Online. AAAI Press.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Ji Yoon Han, Tae Hwan Oh, Lee Jin, and Hansaem Kim. 2020. Annotation issues in Universal Dependencies for Korean and Japanese. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 99–108, Barcelona, Spain (Online). Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 770–778, Las Vegas, NV, USA. IEEE Computer Society.

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

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. 2019. Do attention heads in BERT track syntactic dependencies? *CoRR*, abs/1911.12246.

Richard A. Hudson. 1987. Zwicky on heads. *Journal of Linguistics*, 23:109–132.

Dieuwke Hupkes, Sara Veldhoen, and Willem H. Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *J. Artif. Intell. Res.*, 61:907–926.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. 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 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Jason Jo and Yoshua Bengio. 2017. Measuring the tendency of CNNs to learn surface statistical regularities. *CoRR*, abs/1711.11561.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

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

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 478–485, Barcelona, Spain. Association for Computational Linguistics.

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

Lingpeng Kong, Cyprien de Masson d'Autume, Lei Yu, Wang Ling, Zihang Dai, and Dani Yogatama. 2020. A mutual information maximization perspective of language representation learning. In *8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa, Ethiopia. OpenReview.net.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic. Association for Computational Linguistics.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *CoNLL-2002: The 6th Conference on Natural Language Learning 2002*, Taipei, Taiwan.

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

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 507–514, Sydney, Australia. Association for Computational Linguistics.

Marco Kuhlmann and Joakim Nivre. 2010. Transition-based techniques for non-projective dependency parsing. *Northern European Journal of Language Technology (NEJLT)*, 2(1):1–19, Linköping University Electronic Press.

Jenny Kunz and Marco Kuhlmann. 2020. Classifier probes may just learn from linear context features. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5136–5146, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jenny Kunz and Marco Kuhlmann. 2021. Test harder than you train: Probing with extrapolation splits. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 15–25, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 729–739,

Doha, Qatar. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland. Association for Computational Linguistics.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yüksekgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2022. Holistic evaluation of language models. *CoRR*, abs/2211.09110.

Tomasz Limisiewicz, David Mareček, and Rudolf Rosa. 2020. Universal Dependencies According to BERT: Both More Specific and More General. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2710–2722, Online. Association for Computational Linguistics.

Jimmy Lin, Aaron Fernandes, Boris Katz, Gregory Marton, and Stefanie Tellex. 2002. Extracting answers from the web using data annotation and knowledge mining techniques. In *Proceedings of The Eleventh Text REtrieval Conference, TREC 2002*, volume 500-251 of *NIST Special Publication*, Gaithersburg, Maryland. National Institute of Standards and Technology (NIST).

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, MIT Press.

Haitao Liu. 2008. Dependency distance as a metric of language comprehension difficulty. *Journal of Cognitive Science*, 9(2):159–191.

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

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Ryan McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. Ph.D. thesis, University of Pennsylvania.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan. Association for Computational Linguistics.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York City. Association for Computational Linguistics.

Ryan McDonald and Joakim Nivre. 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.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230, MIT Press.

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

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Igor Aleksandrovic Mel'cuk et al. 1988. *Dependency syntax: theory and practice*. SUNY Press.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Workshop Track Proceedings*, Scottsdale, Arizona, USA.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2021. Recent advances in natural language processing via large pre-trained language models: A survey. *CoRR*, abs/2111.01243.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Francis Mollica, Matthew Siegelman, Evgeniia Diachek, Steven T Piantadosi,

Zachary Mineroff, Richard Futrell, Hope Kean, Peng Qian, and Evelina Fedorenko. 2020. Composition is the core driver of the language-selective network. *Neurobiology of Language*, 1(1):104–134, MIT Press.

Yugo Murawaki. 2019. On the definition of japanese word. *CoRR*, abs/1906.09719.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Nikita Nangia and Samuel R. Bowman. 2019. Human vs. muppet: A conservative estimate of human performance on the GLUE benchmark. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4566–4575, Florence, Italy. Association for Computational Linguistics.

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

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics.

Joakim Nivre. 2007. Incremental non-projective dependency parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 396–403, Rochester, New York. Association for Computational Linguistics.

Joakim Nivre. 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.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

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

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

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency

parsing. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 49–56, Boston, Massachusetts, USA. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York City. Association for Computational Linguistics.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan. Association for Computational Linguistics.

Youngbin Noh, Jiyoon Han, Tae Hwan Oh, and Hansaem Kim. 2018. Enhancing Universal Dependencies for Korean. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 108–116, Brussels, Belgium. Association for Computational Linguistics.

Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a turkish treebank. *Treebanks: Building and using parsed corpora*, pages 261–277, Springer.

Mai Omura, Aya Wakasa, and Masayuki Asahara. 2021. Word delimitation issues in UD Japanese. In *Proceedings of the Fifth Workshop on Universal Dependencies (UDW, SyntaxFest 2021)*, pages 142–150, Sofia, Bulgaria. Association for Computational Linguistics.

Timothy Osborne and Kim Gerdes. 2019. The status of function words in dependency grammar: A critique of universal dependencies (UD). *Glossa: a journal of general linguistics*, 4(17).

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1310–1318, Atlanta, GA, USA. JMLR.org.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. Rissanen data analysis: Examining dataset characteristics via description length. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 8500–8513, Online. PMLR.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word

representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Tiago Pimentel, Naomi Saphra, Adina Williams, and Ryan Cotterell. 2020a. Pareto probing: Trading off accuracy for complexity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3138–3153, Online. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020b. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. OpenAI.

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

Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. 2021. Probing the probing paradigm: Does probing accuracy entail task relevance? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3363–3377, Online. Association for Computational Linguistics.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471, Elsevier.

Pedro Rodriguez, Joe Barrow, Alexander Miserlis Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. 2021. Evaluation examples are not equally informative: How should that change NLP leaderboards? 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 4486–4503, Online. Association for Computational Linguistics.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. 2020. The pitfalls of simplicity bias in neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9573–9585. Curran Associates, Inc.

Robert F. Simmons, Sheldon Klein, and Keren McConlogue. 1964. Indexing and dependency logic for answering english questions. *American Documentation*, 15:196–204, Wiley Subscription Services, Inc.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021a. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2888–2913, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Koustuv Sinha, Prasanna Parthasarathi, Joelle Pineau, and Adina Williams. 2021b. UnNatural Language Inference. 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 7329–7346, Online. Association for Computational Linguistics.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 132–140, Prague, Czech Republic. Association for Computational Linguistics.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zheng, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using DeepSpeed and Megatron to train Megatron-Turing NLG 530B, A large-scale generative language model. *CoRR*, abs/2201.11990.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Umut Sulubacak, Gülşen Eryiğit, and Tuğba Pamay. 2016a. Imst: A revisited Turkish dependency treebank. In *Proceedings of TurCLing 2016, the 1st international conference on Turkic computational linguistics*, Konya, Turkey. Ege University Press.

Umut Sulubacak, Memduh Gokirmak, Francis Tyers, Çağrı Çöltekin, Joakim Nivre, and Gülşen Eryiğit. 2016b. Universal Dependencies for Turkish. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3444–3454, Osaka, Japan. The COLING 2016 Organizing Committee.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 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.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019*, New Orleans, LA, USA. OpenReview.net.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Klincksieck.

Ivan Titov and James Henderson. 2007a. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 947–951, Prague, Czech Republic. Association for Computational Linguistics.

Ivan Titov and James Henderson. 2007b. A latent variable model for generative dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 144–155, Prague, Czech Republic. Association for Computational Linguistics.

Utku Türk, Furkan Atmaca, Şaziye Betül Özateş, Balkız Öztürk Başaran, Tunga Güngör, and Arzucan Özgür. 2019. Improving the annotations in the Turkish Universal Dependency treebank. In *Proceedings of the Third Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*, pages 108–115, Paris, France. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 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.

Clara Vania, Phu Mon Htut, William Huang, Dhara Mungra, Richard Yuanzhe Pang, Jason Phang, Haokun Liu, Kyunghyun Cho, and Samuel R. Bowman. 2021. Comparing test sets with item response theory. 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 1141–1158, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In

Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Chang Wang, Aditya Kalyanpur, James Fan, Branimir K Boguraev, and DC Gondek. 2012. Relation extraction and scoring in DeepQA. *IBM Journal of Research and Development*, 56(3.4):9–1, IBM.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392, MIT Press.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, MIT Press.

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Jerry W. Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. 2023. Larger language models do in-context learning differently. *CoRR*, abs/2303.03846.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. 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 11–20, Hong Kong, China. Association for

Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Patrick Xia, Shijie Wu, and Benjamin Van Durme. 2020. Which *BERT? A survey organizing contextualized encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7516–7533, Online. Association for Computational Linguistics.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. 2020. A theory of usable information under computational constraints. In *8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa, Ethiopia. OpenReview.net.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 195–206, Nancy, France.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Deniz Yuret. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, Massachusetts Institute of Technology.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, Hao Peng, Jianxin Li, Jia Wu, Ziwei Liu, Pengtao Xie, Caiming Xiong, Jian Pei, Philip S. Yu, and Lichao Sun. 2023. A comprehensive survey on pretrained foundation models: A history from BERT to chatGPT. *CoRR*, abs/2302.09419.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015a. A re-ranking model for dependency parser with recursive convolutional neural network. 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 1159–1168, Beijing, China. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015b. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015c. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015*, pages 19–27, Santiago, Chile. IEEE Computer Society.

Arnold M. Zwicky. 1985. Heads. *Journal of Linguistics*, 21:1–29, Cambridge University Press.