IWPT 2025

18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025)

Proceedings

August 26, 2025

The IWPT organizers gratefully acknowledge the support from the following sponsors.

Organized by

As part of SyntaxFest 2025

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
317 Sidney Baker St. S
Suite 400 - 134
Kerrville, TX 78028
USA
Tel: +1-855-225-1962
acl@aclweb.org

ISBN 979-8-89176-294-7

Introduction

Welcome to the 18th International Conference on Parsing Technologies (IWPT 2025). Since its inaugural meeting in 1989 in Pittsburgh, PA, USA, the IWPT meeting series has been hosted by the Special Interest Group on Natural Language Parsing of the Association for Computational Linguistics as SIGPARSE's primary specialized forum for research on natural language parsing. This year, for the first time, IWPT is held as part of SyntaxFest 2025 in Ljubljana, Slovenia, which brings together five related but independent events:

- 18th International Conference on Parsing Technologies (IWPT 2025)
- 8th Universal Dependencies Workshop (UDW 2025)
- 8th International Conference on Dependency Linguistics (DepLing 2025)
- 23rd Workshop on Treebanks and Linguistic Theories (TLT 2025)
- 3rd Workshop on Quantitative Syntax (QUASY 2025)

In addition, a pre-conference workshop organized by the COST Action CA21167 – Universality, Diversity and Idiosyncrasy in Language Technology (UniDive) was held prior to the main event, with dedicated sessions on the 1st UniDive Shared Task on Morphosyntactic Parsing and the 2nd Workshop on Universal Dependencies for Turkic Languages.

SyntaxFest 2025 continues the tradition of SyntaxFest 2019 (Paris, France), SyntaxFest 2021 (Sofia, Bulgaria), and GURT/SyntaxFest 2023 (Washington DC, USA) in bringing together multiple events that share a common interest in using corpora and treebanks for empirically validating syntactic theories, studying syntax from quantitative and theoretical points of view, and training machine learning models for natural language processing. Much of this research is increasingly multilingual and cross-lingual and requires continued systematic analysis from various theoretical, applied, and practical perspectives. By co-locating these workshops under a shared umbrella, SyntaxFest fosters dialogue between overlapping research communities and supports innovation at the intersection of linguistics and language technology.

As in previous editions, all five workshops at SyntaxFest 2025 shared a common submission and reviewing process, with a unified timeline, identical submission formats, and a shared program committee. During submission, authors could indicate one or more preferred venues, but the final assignment of papers was determined by the collective program chairs, composed of the individual workshop chairs, based on thematic alignment. All accepted submissions were peer-reviewed by at least three reviewers from the shared program committee.

In total, SyntaxFest 2025 received 94 submissions, of which 73 (78%) were accepted for presentation. The final program included a total of 47 long papers, 21 short papers, and 5 non-archival contributions, distributed across the five workshops: 5 papers were presented at IWPT (2 long, 3 short); 20 at UDW (14 long, 5 short, 1 non-archival); 16 at DepLing (12 long, 2 short, 2 non-archival); 18 at TLT (10 long, 7 short, 1 non-archival); and 14 at QUASY (9 long, 4 short, 1 non-archival).

Our sincere thanks go to everyone who made this event possible. We thank all authors for their submissions and the reviewers for their time and thoughtful feedback, which contributed to a diverse and high-quality program. Special thanks go to the local organizing team at the University of Ljubljana and the Slovene Language Technologies Society for hosting the event, and to the sponsors for their generous support. Finally, we gratefully acknowledge ACL SIGPARSE for endorsing the event and the ACL Anthology for publishing the proceedings.

Kenji Sagae, Stephan Oepen (IWPT 2025 Chairs)
Gosse Bomma, Çağrı Çöltekin (UDW 2025 Chairs)
Eva Hajičová, Sylvain Kahane (DepLing 2025 Chairs)
Heike Zinsmeister, Sarah Jablotschkin, Sandra Kübler (TLT 2025 Chairs)

Xinying Chen, Yaqin Wang (QUASY 2025 Chairs)
Kaja Dobrovoljc (SyntaxFest 2025 Organization Chair)

Ljubljana, August 2025

Organizing Committee

TLT Chairs

Heike Zinsmeister, University of Hamburg
Sarah Jablotschkin, University of Hamburg
Sandra Kübler, Indiana University

DepLing Chairs

Eva Hajičová, Charles University, Prague
Sylvain Kahane, Université Paris Nanterre

UDW Chairs

Gosse Bomma, University of Groningen
Çağrı Çöltekin, University of Tübingen

IWPT Chairs

Kenji Sagae, University of California, Davis
Stephan Oepen, University of Oslo

QUASY Chairs

Xinying Chen, University of Ostrava
Yaqin Wang, Guangdong University of Foreign Studies

Publication Chair

Sarah Jablotschkin, University of Hamburg

Local SyntaxFest 2025 Organizing Committee

Kaja Dobrovoljc, University of Ljubljana, SDJT
Špela Arhar Holdt, University of Ljubljana
Luka Terčon, University of Ljubljana
Marko Robnik-Šikonja, University of Ljubljana
Matej Klemen, University of Ljubljana
Sara Kos, University of Ljubljana
Timotej Knez, University of Ljubljana, SDJT
Tinca Lukan, University of Ljubljana

Special Thanks for designing the SyntaxFest 2025 logo to

Kim Gerdes, Université Paris-Saclay

Program Committee

Shared Program Committee

V.S.D.S.Mahesh Akavarapu, Eberhard-Karls-Universität Tübingen
Leonel Figueiredo de Alencar, Federal University of Ceará (UFC)
Patricia Amaral, Indiana University
Giuseppe Attardi, University of Pisa
John Bauer, Stanford University
David Beck, University of Alberta
Laura Becker, Albert-Ludwigs-Universität Freiburg
Aleksandrs Berdicevskis, Gothenburg University
Ann Bies, University of Pennsylvania
Igor Boguslavsky, Universidad Politécnica de Madrid
Bernd Bohnet, Google
Cristina Bosco, University of Turin
Gosse Bouma, University of Groningen
Miriam Butt, Universität Konstanz
G. A. Celano, Universität Leipzig
Heng Chen, Guangdong University of Foreign Studies
Xinying Chen, University of Ostrava
Jinho D. Choi, Emory University
Çağrı Çöltekin, University of Tuebingen
Daniel Dakota, Leidos
Stefania Degaetano-Ortlieb, Universität des Saarlandes
Kaja Dobrovoljc, University of Ljubljana
Jakub Dotlacil, Utrecht University
Gülşen Eryiğit, Istanbul Technical University
Kilian Evang, Heinrich Heine University Düsseldorf
Pegah Faghiri, CNRS
Ramon Ferrer-i-Cancho, Universidad Politécnica de Cataluna
Marcos Garcia, Universidade de Santiago de Compostela
Kim Gerdes, Université Paris-Saclay
Loïc Grobol, Université Paris Nanterre
Bruno Guillaume, INRIA
Carlos Gómez-Rodríguez, Universidade da Coruña
Eva Hajicova, Charles University
Dag Trygve Truslew Haug, University of Oslo
Santiago Herrera, University of Paris Nanterre
Richard Hudson, University College London
Maarten Janssen, Charles University Prague
Jingyang Jiang, Zhejiang University
Mayank Jobanputra, Universität des Saarlandes
Sylvain Kahane, Université Paris Nanterre
Václava Kettnerová, Charles University Prague
Sandra Kübler, Indiana University
Guy Lapalme, University of Montreal
François Lareau, Université de Montréal
Miryam de Lhoneux, KU Leuven
Zoey Liu, University of Florida

Teresa Lynn, Dublin City University
Jan Macutek, Slovak Academy of Sciences
Robert Malouf, San Diego State University
Marie-Catherine de Marneffe, UCLouvain
Nicolas Mazziotta, Université de Liège
Alexander Mehler, Johann Wolfgang Goethe Universität Frankfurt am Main
Maitrey Mehta, University of Utah
Wolfgang Menzel, Universität Hamburg
Marie Mikulová, Charles University
Aleksandra Miletić, University of Helsinki
Jasmina Milićević, Dalhousie University
Simon Mille, Dublin City University
Yusuke Miyao, The University of Tokyo
Noor Abo Mokh, Indiana University
Simonetta Montemagni, Institute for Computational Linguistics "A. Zampolli" (ILC-CNR)
Jiří Mírovský, Charles University Prague
Kaili Müürisep, Institute of computer science, University of Tartu
Anna Nedoluzhko, Charles University Prague
Ruochen Niu, Beijing Language and Culture University
Joakim Nivre, Uppsala University
Stephan Oepen, University of Oslo
Timothy John Osborne, Zhejiang University
Petya Osenova, Sofia University "St. Kliment Ohridski"
Agnieszka Patejuk, Polish Academy of Sciences
Lucie Poláková, Charles University Prague
Prokopis Prokopidis, Athena Research Center
Mathilde Regnault, Universität Stuttgart
Kateřina Rysová, University of South Bohemia
Magdaléna Rysová, Charles University Prague
Tanja Samardzic, University of Zurich
Giuseppe Samo, Beijing Language and Culture University
Haruko Sanada, Rissho University
Nathan Schneider, Georgetown University
Djamé Seddah, Sorbonne University
Anastasia Shimorina, Orange
Maria Simi, University of Pisa
Achim Stein, University of Stuttgart
Daniel G. Swanson, Indiana University
Luka Terčon, Faculty of Arts, University of Ljubljana
Giulia Venturi, Institute for Computational Linguistics "A. Zampolli" (ILC-CNR)
Veronika Vincze, University of Szeged
Yaqin Wang, Guangdong University of Foreign Studies
Pan Xiaxing, Huaqiao University
Chunshan Xu, Anhui Jianzhu University
Nianwen Xue, Brandeis University
Jianwei Yan, Zhejiang University
Zdenek Zabokrtsky, Faculty of Mathematics and Physics, Charles University Prague
Eva Zehentner, University of Zurich
Amir Zeldes, Georgetown University
Daniel Zeman, Charles University Prague
Šárka Zikánová, Charles University Prague

Heike Zinsmeister, Universität Hamburg

Keynote

What can we learn from language models?

Isabel Papadimitriou
Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard University

Abstract: This talk will examine how linguistic theory can benefit from the recent surprising successes of language models in modeling human language production. Language models provide linguists with an unprecedented empirical tool to expand and test our theoretical hypotheses about language. I will go over two main methodologies for taking advantage of language models as an empirical tool. Firstly, examining language model internals as functional theories for how linguistic information can be represented in ways that lead to linguistic capabilities. Secondly, using model training as an empirical testbed, examining what kinds of environments make statistical language learning possible or harder. Both methodologies showcase the importance of developing empirical paradigms that narrow the gap between computational methods and linguistic concerns in order to make language models able to help us expand theoretical horizons.

Bio: Isabel Papadimitriou is a Kempner Fellow at the Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard, and incoming as an assistant professor of linguistics at the University of British Columbia. She is interested in analyzing how large language models learn and represent abstract structural systems, and in how experiments on language models can help enrich the hypothesis space around what makes the learning and representation of language possible.

Table of Contents

An Efficient Parser for Bounded-Order Product-Free Lambek Categorial Grammar via Term Graph

Jinman Zhao and Gerald Penn
Dept. of Computer Science
University of Toronto
CANADA
{jzhao,gpenn}@cs.toronto.edu

Abstract

Lambek Categorial Grammar (LCG) parsing has been proved to be an NP-complete problem. However, in the bounded-order case, the complexity can be reduced to polynomial time. Fowler (2007) first introduced the *term graph*, a simple graphical representation for LCG parsing, but his algorithm for using it remained largely inscrutable. Pentus (2010) later proposed a polynomial algorithm for bounded-order LCG parsing based on cyclic linear logic, yet both approaches remain largely theoretical, with no open-source implementations available. In this work, we combine the term-graph representation with insights from cyclic linear logic to develop a novel parsing algorithm for bounded-order LCG. Furthermore, we release our parser as an open-source tool.

1 Introduction

Many studies have shown that transformer-based models such as large language models (LLMs) effectively capture certain aspects of syntactic structure (Niu et al., 2022; Strobl et al., 2024; Ramesh et al., 2024; Cagnetta and Wyart, 2024). Coming to terms with better representations of syntax could play a significant role in future LLM research, contributing to advancements in areas such as mitigating hallucinations (Wu and Liu, 2025) and reasoning (Barke et al., 2024).

While most current research on syntax in NLP primarily focuses on context-free grammars (CFGs), categorial grammar (CG) deserves greater attention due to its unique advantages. Unlike CFGs, which rely on a predefined set of production rules, CG is inherently lexicalized, meaning that all grammatical variations are captured within the lexicon itself. This allows syntactic processing to be driven directly by the lexical categories present in a sentence, rather than by a global rule set. Additionally, CG strongly adheres to the principle of compositionality, as seen in Montague grammar, ensuring that syntactic and semantic derivations align closely. This property makes semantic interpretation more transparent and directly extractable from syntax, and could be particularly beneficial for improving the still fraught understanding of the interplay between structure and meaning by the neural language modellling community.

Downstream tasks that leverage CG's syntactic representations to interpret sentence structure generally involve two stages: 1) supertagging (Bhargava and Penn, 2020; Tian et al., 2020; Kogkalidis and Moortgat, 2023), in which each word is assigned a syntactic category, and 2) sequent derivation (Yamaki et al., 2023; Clark, 2015; Fowler, 2007), which organizes these categories into a coherent graphical structure that captures the sentence's grammatical composition.

Like other CG formalisms, Lambek Categorial Grammar (LCG) parsing is amenable to this two-step process. A useful supertagger (Zhao and Penn, 2024) for LCG has already been proposed, allowing us to focus on the second sequent derivation step. But LCG sequent derivation has been proved to be NP-complete (Pentus, 2006). Fortunately, it becomes polynomially solvable under a bounded-order assumption (Fowler, 2007; Pentus, 2010). This assumption is not only theoretically appealing but also empirically justified: in practical scenarios, the syntactic category order tends to remain low. For instance, in both the CCGbank (Hockenmaier and Steedman, 2007) and LCGbank corpus (Bhargava et al., 2024), the maximum order is only 5 (Fowler, 2008), suggesting that bounded-order parsing is sufficient for most real-world applications. Fowler (2007) introduced the *term graph*, a simple graphical representation for LCG parsing. While it also proposed a polynomial-time algorithm with complexity $O(n^3)$ for bounded-order parsing, that approach was never properly explicated and its proof of correctness is overly complex. Pentus (2010) developed an alternative $O(n^4)$ algo-

Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), pages 1–10
August 26, 2025

rithm based on cyclic linear logic. In this work, we prove that the insights from cyclic linear logic also work for term graphs and use these to propose an efficient yet simple algorithm for bounded-order LCG parsing using term graphs that remains $O(n^3)$. We release our parser and demonstrate our parser on LCGbank.[1] For expository purposes, only the recognition (yes/no) version of the algorithm is presented in the text.

2 Related Work

LCG was first introduced by Lambek (1958), and since then, numerous variants have been developed, including ones that are product-free, using only $/$ and $\backslash$ as connectives, unidirectional, with only one of $/$ or $\backslash$, and lexicalized, that prohibit the derivation of the empty sequent, among others. The parsing complexity of LCG has been an ongoing topic of research, leading to the introduction of various frameworks aimed at addressing parsing challenges, such as proof nets (Roorda, 1991), LC-Graphs (Penn, 2004), term graphs (Fowler, 2007), and cyclic linear logic (Girard, 1989; Yetter, 1990).

A key milestone in this line of work was the proof proposed by Pentus (2006) that derivability in the original LCG is NP-complete. Subsequent studies further demonstrated that derivability in the product-free (Savateev, 2012) and semidirectional (Dörre, 1996) LCGs is also NP-complete, while unidirectional (Savateev, 2009) derivability has been shown to be solvable in polynomial time. Despite this theoretical complexity, in practical settings, both the original LCG (Pentus, 2010) and its product-free (Fowler, 2007) variant have been proved to be polynomial-time solvable under reasonable constraints, making them more feasible for real-world applications.

3 Preliminary

3.1 Lambek Categorial Grammar

A *Lambek Categorial Grammar* (LCG) is a formal system used to model natural language syntax through category-based inference. The set of categories C is built from a set of *atomic categories* (e.g., $\{S, NP, N, PP\}$) along with three *binary connectives*: the *forward slash* ($/$), the *backward slash* ($\backslash$) and the *product* ($\cdot$), which encode directional function application. In this work, we focus on the **product-free** (resulting in only two connectives) LCG since the product connective has limited contribution to linguistic.

A *Lambek grammar* G is defined as a four-tuple:

$$G = \langle \Sigma, A, R, S \rangle$$

where Σ is a finite alphabet of symbols (lexical items). A is a set of atomic categories from which complex categories are constructed. R is a relation that maps symbols in Σ to categories in C. S is the set of sentence categories, determining well-formed sentence structures.

Lambek calculus L has the following rules of inference:

$$\frac{\Gamma X \to Y}{\Gamma \to Y/X}\ (/R) \quad \Gamma \text{ is not empty}$$

$$\frac{X\Gamma \to Y}{\Gamma \to X\backslash Y}\ (\backslash R) \quad \Gamma \text{ is not empty}$$

$$\frac{\Gamma \to X \quad \Delta Y \Theta \to Z}{\Delta Y/X\Gamma\Theta \to Z}\ (/L)$$

$$\frac{\Gamma \to X \quad \Delta Y \Theta \to Z}{\Delta\Gamma X\backslash Y\Theta \to Z}\ (\backslash L)$$

$$\frac{\Gamma \to X \quad \Delta X \Theta \to Y}{\Delta\Gamma\Theta \to Y}\ (\text{CUT})$$

Lambek calculus allowing empty premises, denoted as L^*, is a special case where Γ can be empty.

Sequent derivability problem is to determine whether a sequent $\Gamma \vdash s, s \in S$ is derivable under L (or L^*).

3.2 Bounded-Order

We define the *order* of a category, denoted $o(\alpha)$, as a measure of the depth of argument implication nesting. The definition proceeds recursively as follows:

- $o(\alpha) = 0$, if α is a basic (atomic) category;
- $o(\alpha/\beta) = o(\beta\backslash\alpha) = \max(o(\alpha), o(\beta) + 1)$, for complex categories.

We have $o(\text{NP}) = 0$, $o((\text{NP}\backslash\text{S})/\text{NP}) = 1$, and $o((\text{S}/\text{NP})\backslash(\text{S}/\text{NP})) = 2$ as examples.

The maximum order of a category can also be interpreted as the depth of the corresponding term frame structure

3.3 Term Graph

Proof nets (Roorda, 1991; Buch, 2009) are a widely used graphical framework for representing derivations. One key advantage is their ability to merge ambiguous derivations, effectively capturing multiple syntactic structures that share the same semantic interpretation. For instance, as shown in

[1] https://github.com/zhaojinm/LCG_parser.git

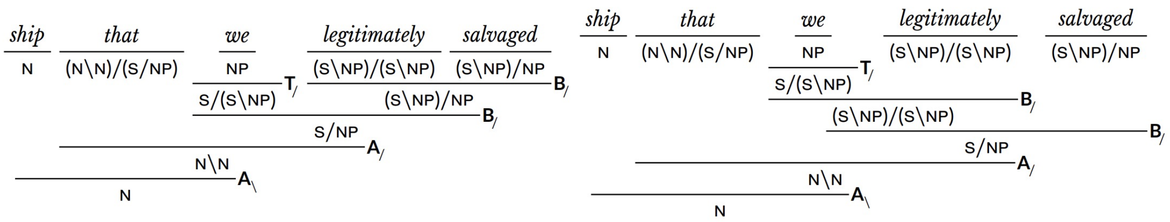


Figure 1: An example of LCG sequent derivation with distinct derivation but same semantics (Bhargava et al., 2024).

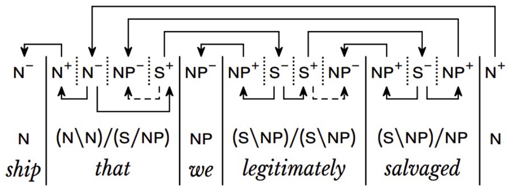


Figure 2: An example of proofnet (Bhargava et al., 2024).

Figure 1, although the two derivations appear structurally different, they convey the same meaning as Figure 2 demonstrated.

There are in fact two algorithmic formalizations of proof nets subsequent to Roorda (1991), however. The more conservative one (Penn, 2004) in terms of natural deduction is what led to term graphs (Fowler, 2007), a simplification that requires less structure to be explicitly maintained. Pentus (2010), which comes from the other formalization, never really embraced all of its advantages.

Constructing a term graph for a given sequent follows a two-step process. Let us use the following sequent as an example:

$$S/(S\backslash NP) \quad (S\backslash NP)/NP \quad NP \quad \vdash S$$

Step 1: Graph Frame Construction

The first step is deterministic and begins by assigning a polarity to each category in the sequent:

- **Negative** polarity is assigned to antecedents(left-hand category).
- **Positive** polarity is assigned to the succedent(right-hand category).

After polarity assignment, the above sequent become:

$$S/(S\backslash NP)^{-} \quad (S\backslash NP)/NP^{-} \quad NP^{-} \quad \vdash S^{+}$$

Each polarized category is treated as a node, and categories containing slashes are decomposed according to the following rewriting rules recursively until no rule can be applied:

$$(\alpha/\beta)^{-} \Rightarrow \alpha^{-} \rightarrow \beta^{+}$$
$$(\beta\backslash\alpha)^{-} \Rightarrow \beta^{+} \leftarrow \alpha^{-}$$
$$(\alpha/\beta)^{+} \Rightarrow \beta^{-} \dashleftarrow \alpha^{+}$$
$$(\beta \backslash \alpha)^{+} \Rightarrow \alpha^{+} \dashrightarrow \beta^{-}$$

In these transformations, the left-hand side of each rule determines the neighborhood of α. The *dashed* edges introduced in this step are referred to as *Lambek edges*, while other connections are called *regular edges*. This process effectively translates syntactic categories into tree structures.

Next, rooted Lambek edges are introduced, connecting the root of the succedent tree to the roots of the antecedent trees. This makes the whole sequent from a forest to a bigger tree. See Figure 3 as a frame for the sequent.

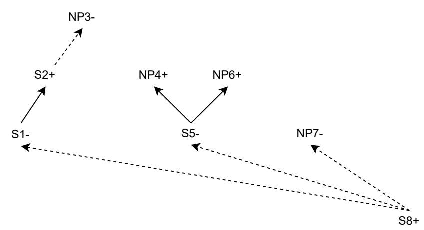


Figure 3: An example of term graph frame.

Step 2: Atom Matching

The second step is *non-deterministic* and involves computing a complete matching of the polarized atoms. The matching must satisfy two constraints:

1. **Planarity**: The edges connecting atoms must not cross when visualized.

2. **Opposite Polarity Pairing**: Every atomic category instance must be paired with exactly one instance of the same category but with *opposite polarity*.

These pairings, called *matches* or *links*, are represented by *regular* edges, are directed from positive atoms to negative atoms.

An example of a *term graph* for a sequent derivation is shown in Figure 4.

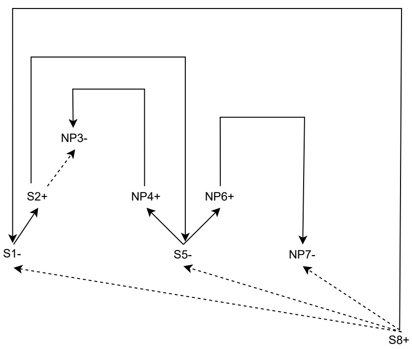


Figure 4: An example of term graph.

Correctness criteria

A term graph G is considered L^**-integral* if it satisfies the following conditions:

1. T(0): It is regular acyclic, for all vertices, there is no regular path to itself.
2. T(1): For every Lambek edge $\langle s, t\rangle$ in G, there exists a regular path from s to t.

A term graph is called *integral* if it is L^**-integral* and additionally satisfies:

3. T(CT): For every Lambek edge $\langle s, t\rangle$ in G, there exists a regular path from s to some vertex x in G. If x has a non-rooted Lambek in-edge $\langle s', x\rangle$, then there must not be a regular path from s to s'.

Theorem 3.1. *A sequent is derivable in L if and only if it has an integral term graph. A sequent is derivable in L^* if and only if it has an L^*-integral term graph.*

Proof. Fowler (2007) □

If we use the naive chart-based parser, the complexity would be NP-complete. Fowler (2007) proposed a method that claims $O(n^3)$ complexity for LCG bounded order sequent derivability, however, this algorithm is very complex and difficult to understand.

3.4 Cyclic linear logic

Cyclic linear logic framework, which, while easy to understand, involves numerous steps. Due to space constraints, we refer readers to Pentus (2010) for a more detailed explanation. Here, we provide only a brief overview of the key idea: they transform sequent derivability in Lambek calculus into sequent derivability in cyclic linear logic. Through a series of transformations, they further convert sequents into a tree-like structure (Figure 5 as an example), allowing for axiom matching.

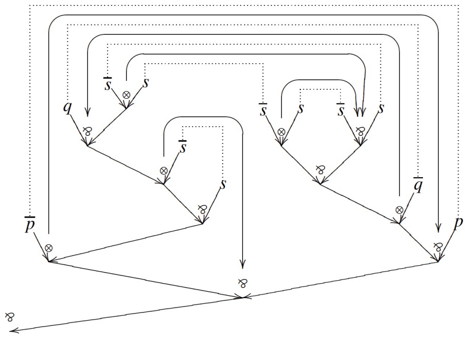


Figure 5: An example of CMLL-based framework (Pentus, 2010). Note that unlike term graph, *order* in CMLL does not equal to the depth of the framework, CMLL's depth is unbounded.

The core idea of their algorithm is that for a span (i, j) in the subtree, it is unnecessary to store a subgraph containing all vertices from i to j. Instead, only the information from two paths in the tree is relevant: one from the root to $axiom_i$ and another from the root to $axiom_j$. Since the depth remains constant under the bounded order condition, the chart-based parser achieves cubic complexity $O(n^3)$ for L^*. However, for L, additional information is required, increasing the complexity to $O(n^4)$.

4 Bounded Order Parser

We combine the strengths of two existing frameworks and propose a simple and easily understandable algorithm based on term graphs. Our approach retains the cubic-time complexity for both $L*$ and L, making it both efficient and practical.

In this section, we first introduce the naive chart parser and our proposed algorithm, followed by a

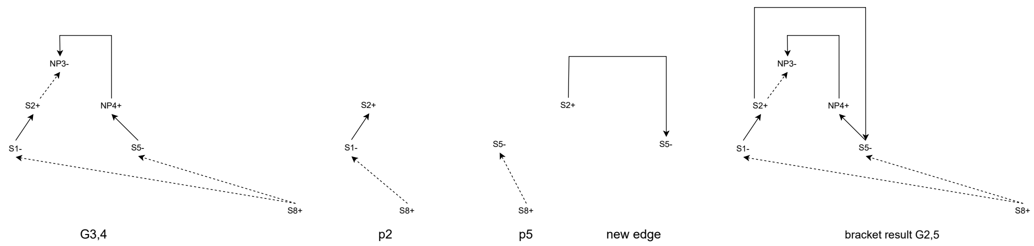


Figure 6: An example of naive bracket operation.

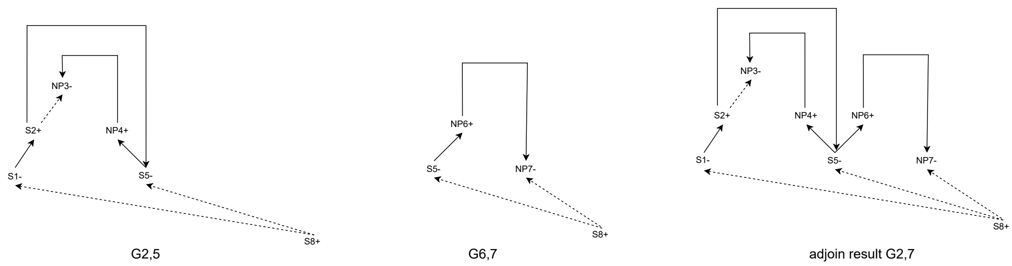


Figure 7: An example of naive adjoin operation.

proof of its correctness.

4.1 Naive Chart Parser for $L*$

A natural approach is to use dynamic programming, following the standard chart parsing paradigm. For each span (i, j), we consider two possible operations: `bracket` and `adjoin`. The `bracket` operation introduces a new link between positions i and j, on top of the subgraph constructed over the inner span $(i+1, j-1)$. Figure 6 illustrates an example of the `bracket` operation. Mathematically, the `bracket` operation can be viewed as the union of graphs:

$$G_{i,j} = G_{i+1,j-1} \cup p_i \cup p_j \cup new\_edge$$

The other operation is `adjoin`, which merges two adjacent subgraphs $G_{i,k}$ and $G_{k+1,j}$ into a larger graph $G_{i,j}$. Figure 7 illustrates an example of the `adjoin` operation. From a mathematical perspective, this corresponds to the composition or union of the two subgraphs:

$$G_{i,j} = G_{i,k} \cup G_{k+1,j}$$

We can apply certain early stopping heuristics during parsing. For example, if a candidate graph $G_{i,j}$ contains a cycle, we can immediately discard it from the chart. However, despite such pruning strategies, the number of possible graphs that may be stored in each chart entry $F_{i,j}$ can still be exponential in the worst case. This is because the number of nodes in $G_{i,j}$ is unbounded, and thus the number of possible subgraph configurations grows exponentially. As a result, the overall complexity of this naive chart parser remains exponential.

4.2 Efficient Parser for $L*$

The key bottleneck of the naive chart parser described above lies in its failure to prune intermediate nodes: each chart entry may store an exponential number of subgraph variants due to the unbounded number of nodes.

We begin by introducing the core insight behind our parser design. Our approach incrementally merges pairs of subgraphs. However, unlike standard methods that may retain the full internal structure of each subgraph, we observe that it is sufficient to preserve only the node information along the two outermost boundary paths. Nodes in the interior of the merged span will no longer be accessed by any subsequent operations from outside the span and thus can be safely ignored. This simplification significantly reduces the complexity.

Frame construction Our method's first step aligns with Section 3.3, and we start from Figure 3.

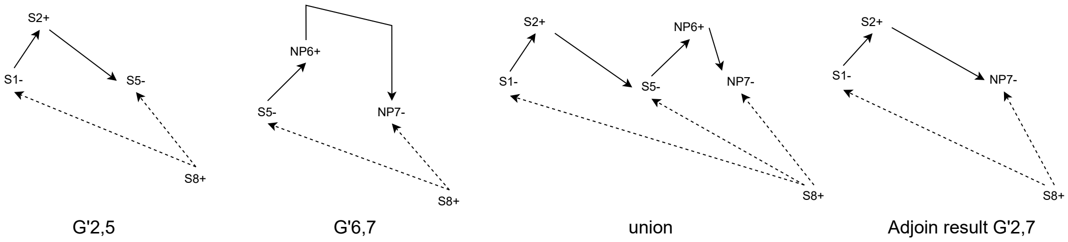


Figure 8: An example of updated adjoin operation. We only illustrate the new `adjoin` operation. However, the graph G'_{25} shown here can be seen as an updated version of the bracket-derived G_{25} from Figure 6. Additionally, the edge $\mathrm{S}_2 \to \mathrm{NP}_7$ in G'_{27} exists because there is a regular path $\mathrm{S}_2 \rightsquigarrow \mathrm{NP}_7$ in the union graph. Similarly, the dash edge $\mathrm{S}_8 \dashrightarrow \mathrm{S}_5$ in the union graph corresponds to $\mathrm{S}_8 \dashrightarrow \mathrm{S}_1$ in G'_{27}, since S_1 (i.e., w in line 9 of Algorithm 1) satisfies $\mathrm{S}_1 \rightsquigarrow \mathrm{S}_5$.

4.2.1 Chart Parser

We continue to define our parser in terms of two operations: `adjoin` and `bracket`, each corresponding to the union of subgraphs. However, since each chart entry $F_{i,j}$ now stores a *simplified graph*, we denote it by $G'_{i,j}$. For the both operation, the update rule remains structurally the same with extra `Simplify` function, for `adjoin`:

$$G'_{i,j} = G'_{i,k} \cup G'_{k+1,j}$$

$$G'_{i,j} = \texttt{Check\_and\_Simplify}(G'_{i,j})$$

and for `bracket`:

$$G'_{i,j} = G'_{i+1,j-1} \cup p_i \cup p_j \cup new\_edge$$

$$G'_{i,j} = \texttt{Check\_and\_Simplify}(G'_{i,j})$$

Here, `Check_and_Simplify` removes redundant internal nodes and preserves only the necessary boundary information. The function `Check_and_Simplify` (Algorithm 1) performs two key check steps and one simplify step sequentially:

1. For all node pairs (u, v) in $G'_{i,j}$, if both $u \rightsquigarrow v$ and $v \rightsquigarrow u$ hold, then the cyclic constraint T(0) is violated. In this case, the function returns `"CYCLIC"` and terminates further computation for this span. (line 1)

2. For every dash edge $u \dashrightarrow v$, if $u \rightsquigarrow v$, then all incoming dash edges to v (i.e., $w \dashrightarrow v$) are deleted. This ensures that the T(1) condition is satisfied for node v and no need to keep this in the future. (line 2)

3. Simplify graph(after line 3). Line 3-4 initialize a new graph where V are nodes on two boundary paths and E is empty. For all (u, v) where $u \rightsquigarrow v$ we can add edge $u \to v$ in the result graph (line 5-7). Line 8-14 deal with the dashed edge, from Lemma 4.7, if $u \dashrightarrow v$, then the dashed path $u \rightsquigarrow v$ is unique and it must pass through w since $w \rightsquigarrow v$. Note that S contains all the nodes the u access to. So the dashed constraint becomes a bunch of constraints; either node in S access to w will make T(1) satisfy for v. Lines 8–14 and line 2 should be considered as a unified process: both are designed to enforce the T(1) constraint.

Figure 8 is the update result for Figure 6 and 7.

Algorithm 1: `Check_and_Simplify`$(G'_{i,j})$

Input: $G'_{i,j}$
Output: A simplified graph
1 T0 check;
2 T1 check;
3 $V = node(pi \cup pj)$
4 $new\_G = \{V, E = \emptyset\}$
5 **foreach** $u, v \in V$ **do**
6   **if** $u \rightsquigarrow v \in G'_{i,j}$ **then**
7         $E.append(u \to v)$;
8 **foreach** $u \dashrightarrow v$ *in* $G'_{i,j}$ **do**
9   Follow v backwards along regular edges to the furthest $w \in V$ such that $w \rightsquigarrow v$.
10     $S = \{s | s \in V, u \rightsquigarrow s\}$
11   **if** $S = \emptyset$ *or* w *is NULL* **then**
12   **return** `'NO REGULAR ACCESS'`
13   **foreach** $s \in S$ **do**
14         $E.append(s \dashrightarrow w)$;
15 **return** `"VALID"`, new_G

4.2.2 Correctness

Let G be the original term graph, $G_{i,j}$ be the partial term graph with span (i, j). And let $G^{'}_{i,j}$ be the simplified graph that only contains nodes in two paths p_i and p_j where p_i is the path in Figure 3 from root to i. We want to prove that $G^{'}_{i,j}$ stores enough information for LCG parsing as $G_{i,j}$.

Lemma 4.1. *If the term graph G is acyclic, the method will not return 'CYCLIC'.*

Proof. By induction, for each edge (u, v) in $G^{'}_{i,j}$, there must exist a path that $u \rightsquigarrow v$ in $G_{i,j}$. Therefore, no cycle in G indicates no cycle can be detected by $G^{'}$. □

Lemma 4.2. *If $G_{i,j}$ is the adjoin of $G_{i,k}$ and $G_{k+1,j}$, then there is no cross match (a, b) such that $i \leq a \leq k$ and $k + 1 \leq b \leq j$.*

Proof. This can be easily proved by induction where the base case is $i = j - 1$. □

Lemma 4.3. *For each $G_{i,j}$, for all $u, v \in p_i \cup p_j$, if $u \rightsquigarrow v$ in $G_{i,j}$, then $u \rightsquigarrow v$ in $G^{'}_{i,j}$.*

Proof. Prove by Induction.
Base: $G_{i,i}$ = $G^{'}_{i,i}$, trivial case.
Induction Step:
Assume $G^{'}_{i+1,j-1}$, $G^{'}_{i+1,k}$ and $G^{'}_{k,j}$ are both satisfiable.
Case 1: `bracketing` where:

$$G_{i,j} = G_{i+1,j-1} \cup p_i \cup p_j \cup new\_edge$$

Assume there is an edge $(u, v) \in p_i \cup p_j$ such that $u \rightsquigarrow v$ in $G_{i,j}$, and $u \not\rightsquigarrow v$ in $G^{'}_{i,j}$. By IS, $u, v \notin (p_i \cap p_{i+1}) \cup (p_{j-1} \cap p_j)$. Thus, there must be a node $x \in (p_i \cap p_{i+1}) \cup (p_{j-1} \cap p_j)$ and $w \in G_{i,j} \setminus (p_i \cup p_{i+1} \cup p_{j-1} \cup p_j)$ such that $x \to w$ or $w \to x$, and such edge does not exists based on our graph construction rule. Contradiction.
Case 2: `adjoin`, where:

$$G_{i,j} = G_{i,k} \cup G_{k+1,j}$$

Since both subgraphs are satisfied by IS and there is no cross-match by Lemma 4.2, the result also holds for $G_{i,j}$. □

Lemma 4.4. *If the term graph G is cyclic, the method will return 'CYCLIC'.*

Proof. If there is a cycle in G, let i be the left most and j be the right most, then by Lemma 4.3, $G_{i,j}$ must contains two edges of (i, j) and (j, i). Then, the cycle check will return 'CYCLIC'. □

Theorem 4.5. *(T(0)) Term graph G is regular acyclic if and only if the method returns 'ACYCLIC'.*

Proof. By Lemma 4.1 and Lemma 4.4. □

Lemma 4.6. *If a dashed edge $u \dashrightarrow v$ in G has no regular access, then the method will return 'NO REGULAR ACCESS'.*

Proof. By induction, for each edge (u, v) in $G^{'}$, there must exist a path that $u \rightsquigarrow v$ in G. Therefore, no regular path in G indicates no regular path can be detected by $G^{'}$. □

Lemma 4.7. *Regular in degree is 1 for all nodes.*

Proof. Induction on the term graph construction. □

Lemma 4.8. *If the method returns 'NO REGULAR ACCESS' for a dashed edge $u \dashrightarrow v$, then this edge has no regular access in G.*

Proof. By Lemma 4.7, such a regular path $u \rightsquigarrow v$ is unique. According to the construction procedure of G', the constraint $u \dashrightarrow v$ is initialized as the pair $(\{u\}, v)$, and is iteratively updated through `adjoin` or `bracket` operations. Suppose an intermediate step yields a span $[i, j]$; the result is then represented as a pair (S, w), where S is a set of nodes and w is a single node, both of which lie on the path $p_i \cup p_j$. This representation implies that at least one node in S must maintain *regular access* to w.

If, during an `adjoin` or `bracket` operation, the method determines that regular access does not hold, then one of the following conditions must be true:

1. No node on $p_i \cup p_j$ has access to w; or

2. No node in S has access to any node on $p_i \cup p_j$.

In either case, by Lemma 4.2, it follows that there exists no path of regular access from S to w. □

Theorem 4.9. *(T(1)) For each dashed edge $u \dashrightarrow v$ in term graph G, $u \rightsquigarrow v$ in G if and only if the method does not return 'NO REGULAR ACCESS'.*

Proof. By Lemma 4.6 and Lemma 4.8. □

4.2.3 Complexity of L*

All operations strictly follow the procedure of a chart parser. Therefore, the overall complexity is

$$O(n^3) \cdot |F_{ij}| \cdot \max(O(\texttt{adjoin}), O(\texttt{bracket}))$$

where $|F_{ij}|$ denotes the number of possible graph configurations for the span $[i, j]$ stored in the chart entry F_{ij}. Since the category order is bounded(i.e., constant), the number of nodes within each G'_{ij} is constant, and the number of possible configurations is also bounded. Moreover, the cost of each graph operation (such as `adjoin` and `bracket`) deals with the constant number of nodes, so both $O(\texttt{adjoin})$ and $O(\texttt{bracket})$ are bounded.

Therefore, the overall complexity is $O(n^3)$.

4.3 Parser for L

While the constraint T(CT) may initially appear to be a condition on Lambek edges, a closer examination reveals that it is in fact a constraint on each positive node. Specifically, we can restate T(CT) as:

- T(CT): for every positive node s^+, there must exist a negative node x^- such that either $root \dashrightarrow x$, or $s \not\rightsquigarrow \texttt{dash\_parent}(x)$.

Moreover, for any positive node s^+, the search for a negative node x^- satisfying the T(CT) condition can terminate as soon as one such x^- is found. This is because T(CT) can no longer be violated once the condition holds for any such x^-. Specifically, by T(1), we have $\texttt{dash\_parent}(x) \rightsquigarrow x$. Suppose at some point we observe that $s^+ \rightsquigarrow x$ but $s^+ \not\rightsquigarrow \texttt{dash\_parent}(x)$. The only way this could occur is if $\texttt{dash\_parent}(x) \rightsquigarrow s^+ \rightsquigarrow x$, which would imply a cycle. However, by T(0), cycles are disallowed, so it must be the case that $s^+ \not\rightsquigarrow \texttt{dash\_parent}(x)$.

This formulation of T(CT) is naturally compatible with our parsing algorithm. For each positive node x_i, we maintain a set $\texttt{CT}_{x_i} = \{x_i\}$, initialized with the node itself. During parsing, whenever an element $v \in \texttt{CT}_{x_i}$ no longer appears on the boundary paths, we update $\texttt{CT}_{x_i}$ by replacing v with all nodes that are reachable from v via regular access and that lie on the current boundary path.

If $\texttt{CT}_{x_i}$ ever becomes empty, this indicates that T(CT) can no longer be satisfied for x_i. On the other hand, once any node in $\texttt{CT}_{x_i}$ finds a matching negative node that satisfies the T(CT) condition, we consider the constraint for x_i satisfied and can safely discard $\texttt{CT}_{x_i}$ from further tracking.

Complexity for L It is worth noting that maintaining the T(CT) constraint introduces only minimal overhead. Since the number of nodes in each simplified subgraph $G'_{i,j}$ is bounded, the number of elements in each $\texttt{CT}_{x_i}$ set remains constant throughout the parsing process. Consequently, the additional computation required to update and check T(CT) constraints does not affect the overall complexity. The total time complexity remains $O(n^3)$.

4.4 Number of Derivations

An additional advantage of our chart-based parser is its ability to naturally track all valid derivations for a given sequent. During parsing, the algorithm maintains distinct derivation paths, allowing it to enumerate all possible syntactic analyses.

4.5 Experimental Results

To illustrate the practical performance and correctness of our parser, we apply it to LCGbank (Bhargava et al., 2024), a Lambek Categorial Grammar variant of the Penn Treebank (Marcus et al., 1993) containing 44,870 labeled sentences. Our parser successfully derives every sentence in the dataset, demonstrating both the efficiency and robustness of the proposed algorithm.

5 Conclusion

In this work, we presented a practical and efficient parser for bounded-order, product-free Lambek Categorial Grammar, based on a refined term-graph framework. Inspired in part by the theoretical insights from cyclic linear logic (Pentus, 2010), our algorithm achieves the same asymptotic complexity as Fowler (2007)'s chart-based parser at $O(n^3)$, while offering significantly improved simplicity and implementability. In contrast, Pentus's original method incurs a higher $O(n^4)$ complexity due to the need for additional structural tracking in cyclic linear logic.

A key innovation in our parser lies in the use of boundary-only representations, which eliminate the need to track internal nodes and allow for aggressive graph simplification without sacrificing correctness. Our method unifies term graph derivability conditions with a lightweight dynamic programming architecture, resulting in a parser that is both fast and easy to understand. Crucially, we provide the first open-source parser that is bounded-order polynomial, enabling further research and application of Lambek grammar in modern NLP workflows.

References

Shraddha Barke, Emmanuel Anaya Gonzalez, Saketh Ram Kasibatla, Taylor Berg-Kirkpatrick, and Nadia Polikarpova. 2024. HYSYNTH: Context-free LLM approximation for guiding program synthesis. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Aditya Bhargava, Timothy A. D. Fowler, and Gerald Penn. 2024. LCGbank: A corpus of syntactic analyses based on proof nets. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10225–10236, Torino, Italia. ELRA and ICCL.

Aditya Bhargava and Gerald Penn. 2020. Supertagging with CCG primitives. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 194–204, Online. Association for Computational Linguistics.

Armin Buch. 2009. Mildly non-planar proof nets for ccg. *OF THE EUROPEAN SUMMER SCHOOL FOR LOGIC, LANGUAGE, AND INFORMATION*, page 160.

Francesco Cagnetta and Matthieu Wyart. 2024. Towards a theory of how the structure of language is acquired by deep neural networks. *Preprint*, arXiv:2406.00048.

Stephen Clark. 2015. The java version of the c&c parser: Version 0.95.

Jochen Dörre. 1996. Parsing with semidirectional Lambek grammar is NP-complete. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 95–100, Santa Cruz, California, USA. Association for Computational Linguistics.

Timothy AD Fowler. 2007. A polynomial time algorithm for parsing with the bounded order lambek calculus. In *Conference on Mathematics of Language*, pages 36–43.

Timothy AD Fowler. 2008. Efficiently parsing with the product-free lambek calculus. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 217–224.

Jean-Yves Girard. 1989. Towards a geometry of interaction. *Categories in computer science and logic*, 92:69–108.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

Konstantinos Kogkalidis and Michael Moortgat. 2023. Geometry-aware supertagging with heterogeneous dynamic convolutions. In *Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD)*, pages 107–119, Gothenburg, Sweden. Association for Computational Linguistics.

Joachim Lambek. 1958. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Jingcheng Niu, Wenjie Lu, and Gerald Penn. 2022. Does BERT rediscover a classical NLP pipeline? In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3143–3153, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Gerald Penn. 2004. A graph-theoretic approach to sequent derivability in the lambek calculus. *Electronic Notes in Theoretical Computer Science*, 53:274–295. Proceedings of the joint meeting of the 6th Conference on Formal Grammar and the 7th Conference on Mathematics of Language.

Mati Pentus. 2006. Lambek calculus is np-complete. *Theoretical Computer Science*, 357(1):186–201. Clifford Lectures and the Mathematical Foundations of Programming Semantics.

Mati Pentus. 2010. A polynomial-time algorithm for lambek grammars of bounded order. *Linguistic Analysis*, 36(1):441–471.

Rahul Ramesh, Ekdeep Singh Lubana, Mikail Khona, Robert P. Dick, and Hidenori Tanaka. 2024. Compositional capabilities of autoregressive transformers: A study on synthetic, interpretable tasks. *Preprint*, arXiv:2311.12997.

Dirk Roorda. 1991. *Resource Logics: Proof-Theoretical Investigations*. Ph.D. thesis.

Yu V Savateev. 2009. Recognition of derivability for the lambek calculus with one division. *Moscow University Mathematics Bulletin*, 64:73–75.

Yury Savateev. 2012. Product-free lambek calculus is np-complete. *Annals of Pure and Applied Logic*, 163(7):775–788. The Symposium on Logical Foundations of Computer Science 2009.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. 2024. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561.

Yuanhe Tian, Yan Song, and Fei Xia. 2020. Supertagging Combinatory Categorial Grammar with attentive graph convolutional networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6037–6044, Online. Association for Computational Linguistics.

Katherine Wu and Yanhong A Liu. 2025. Lp-lm: No hallucinations in question answering with logic programming. *arXiv preprint arXiv:2502.09212*.

Ryosuke Yamaki, Tadahiro Taniguchi, and Daichi Mochihashi. 2023. Holographic CCG parsing. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 262–276, Toronto, Canada. Association for Computational Linguistics.

David N Yetter. 1990. Quantales and (noncommutative) linear logic. *The Journal of Symbolic Logic*, 55(1):41–64.

Jinman Zhao and Gerald Penn. 2024. LLM-supertagger: Categorial grammar supertagging via large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 697–705, Miami, Florida, USA. Association for Computational Linguistics.

Step-by-step Instructions and a Simple Tabular Output Format Improve the Dependency Parsing Accuracy of LLMs

Hiroshi Matsuda **Chunpeng Ma**
Megagon Labs, Tokyo,
Recruit Co., Ltd.
{hiroshi_matsuda, ma.chunpeng}@megagon.ai

Masayuki Asahara
National Institute for Japanese
Language and Linguistics
masayu-a@ninjal.ac.jp

Abstract

Recent advances in large language models (LLMs) have enabled impressive performance in various tasks. However, standard prompting often struggles to produce structurally valid and accurate outputs, especially in dependency parsing. We propose a novel step-by-step instruction strategy, where universal part-of-speech tagging precedes the prediction of syntactic heads and dependency labels, and a simplified CoNLL-U like output format, our method achieves state-of-the-art accuracy on Universal Dependencies datasets across 17 languages without hallucination or contamination. We further show that multilingual fine-tuning simultaneously improves cross-language generalization performance. Our results highlight the effectiveness of explicit reasoning steps in LLM-based parsing and offer a scalable, format-consistent alternative to bracket-based approaches.

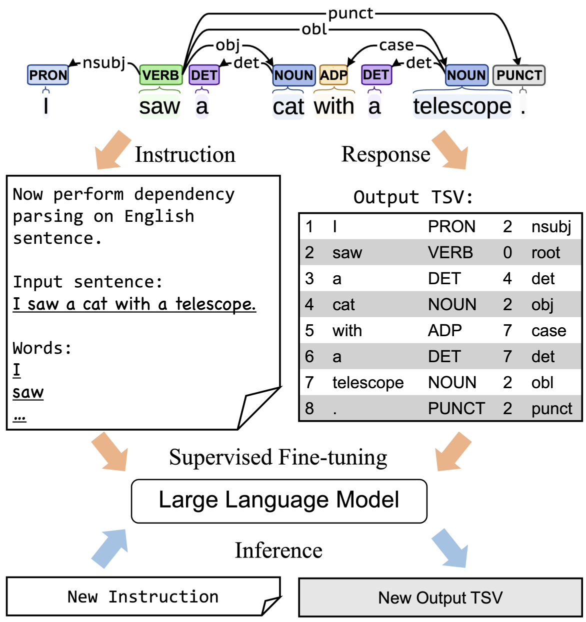


Figure 1: Framework of the proposed method.

1 Introduction

Recent advances in large language models (LLMs) have dramatically reshaped the landscape of natural language processing; however, their potential for syntactic analysis – particularly dependency parsing – remains underexplored. Furthermore, it is desirable to systematically investigate prompting and fine-tuning techniques that enhance the performance of LLM-based dependency parsing.

In this work, we examine how fine-tuned LLMs can be effectively guided to perform accurate dependency parsing using simple, structured instruction prompts. Specifically, we design a single-turn supervised fine-tuning setup where the input sentence is accompanied by a tabular output format based on a minimal subset of the `CoNLL-U`[1], which is the standard format of Universal Dependencies (UD) treebanks (Nivre et al., 2020) as in Figure 1. This table-based representation not only improves format validity and readability, but also facilitates learning non-projective structures.

The results of our preliminary experiments using `UD_English-EWT`[2] are summarized in Table 1. First, we found that performing SFT with a single-step prompt yielded accuracy comparable to or better than that of `UDPipe 2.0` (Straka, 2018). Next, we introduced a step-by-step prompting strategy in a Chain-of-Thought style (Wei et al., 2022). Specifically, we first predict UPOS tags, then syntactic heads and dependency relations. We observed that step-by-step prompts leads to substantial gains in both unlabeled attachment score (UAS) and labeld attachment socre (LAS).

Despite using a very simple prompt, we observed fairly high parsing accuracy, prompting us to investigate the possibility of data contamination (refer Appendix B for details). Based on our analysis, we found no evidence of contamination in

[1] https://universaldependencies.org/format.html

[2] https://universaldependencies.org/treebanks/en_ewt/

Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), pages 11–19
August 26, 2025

	Token Recall	UPOS	UAS	LAS
`UDPipe 2.0`	100.0	97.5	93.4	91.5
`gpt-4o-mini-2024-07-18`: Chain-of-Thought Steps				
1 (UPOS+HEAD+DEPREL)	98.5	98.4	93.3	91.4
2 (UPOS; HEAD+DEPREL)	99.1	**98.6**	94.8	93.1
3 (UPOS; HEAD; DEPREL)	**99.6**	**98.6**	**95.1**	**93.5**

Table 1: Preliminary experiment on evaluating Chain-of-Thought effect in `UD_English-EWT r2.15`. We performed all steps within a single-turn prompt. The example prompts are presented in Appendix C.

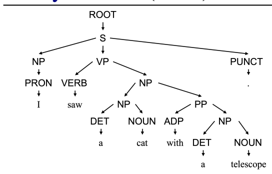
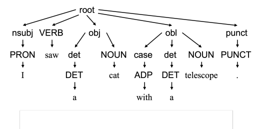
Vinyals et al. (2015)	Hromei et al. (2024)
	
(ROOT♦(S♦ (NP♦PRON♦)$_{NP}$♦ (VP♦VERB♦(NP♦ (NP♦DET♦NOUN♦)$_{NP}$♦ (PP♦ADP♦ (NP♦DET♦NOUN♦)$_{NP}$♦)$_{PP}$♦)$_{NP}$♦)$_{VP}$♦ PUNCT♦)$_{S}$♦)$_{ROOT}$	[root♦[nsubj♦[PRON♦[I♦]]] ♦[VERB♦[saw♦]]♦ [obj♦[det♦[DET♦[a♦]]]♦ [NOUN♦[cat♦]]]♦ [obl♦[case♦[ADP♦[with♦]]] ♦[det♦[DET♦[a♦]]]♦ [NOUN♦[telescope♦]]]♦ [punct♦[PUNCT♦[.♦]]]]

Table 2: Comparison of bracket-based linearization methods. Syntactic elements are separated by "♦".

the prediction of syntactic heads, and dependency relations by the models used in this study for the test set of `UD_English-EWT r2.15`. However, we suspect that the part-of-speech tagging may have been exposed to the models during its pre- and mid-training[3] phases.

2 Related Work

Linearization techniques are essential for both constituency parsing (Vinyals et al., 2015; Ma et al., 2017) and dependency parsing (Li et al., 2018; Hromei et al., 2024) using sequence-to-sequence model with bracket-based representations, illustrated in Table 2.

In generative parsing using bracket-based representations, the tree structure in the output text is often invalid, which is one of the factors that reduces the accuracy of parsing, resulting in additional recovery procedure (Bai et al., 2023), or even redesign the topology of neural networks to ensure the output validity (Dyer et al., 2015; Gómez-Rodríguez and Vilares, 2018).

[3]https://vintagedata.org/blog/posts/what-is-mid-training

3 Approach

In this section, we describe a table-based representation of dependency structures, similar to the `CoNLL-U` format, and explain how to construct instruction prompts for dependency parsing.

3.1 Table-based representation

Recent large language models (LLMs) have significantly improved their ability to output in structured formats such as JSON or CSV, enabling function calling for flexible interaction with external services[4]. This capability facilitates the direct handling of tabular structures such as `CoNLL-U`, potentially allowing LLMs to generate parse results with higher structural validity compared to the bracket-based representations employed in prior studies.

In this work, we adopt a table-based representation that extracts only the essential fields – ID, FORM, UPOS, HEAD, and DEPREL – from the `CoNLL-U` format, as illustrated in the output TSV in Figure 1. A further advantage of the table-based approach is its ability to naturally represent non-projective dependency structures using index-based head references. However, it should be noted that table-based representations can represent circular references and multiple roots. As we demonstrate in the next section, the tabular outputs generated by the LLMs were mostly well-formed, and the validity errors were fairly rare on the `UD_English-EWT` r2.15 test set. Furthermore, the table-based representation offers an advantage in recovery processing, as it can accurately recover word indices and forms as long as the number of records and the field structure are correctly output.

3.2 Step-by-step instruction prompts

We began our preliminary experiments using the simple single-step prompt illustrated in Figure 1. Through iterative refinement, we found that parsing the UPOS tags first, followed by the HEAD and DEPREL fields in a step-by-step manner, led to improved accuracy. Accordingly, the experiments presented in next chapter employ a three-step Chain-of-Thought prompting strategy, processing the elements in the order of UPOS, HEAD, and DEPREL. Representative examples of these prompt templates are provided in the Appendix C.

[4]https://platform.openai.com/docs/guides/function-calling

Models	# of Parameters Trainable vs. Transformer	UD_English-EWT r2.2 Token Recall	UPOS	UAS	LAS	UD_English-EWT r2.15 Token Recall	UPOS	UAS	LAS
Baselines:									
UDPipe 2.0	30.1M† / 168M	-	-	-	-	-	97.5	93.4	91.5
Hexatagger bert-base	222M / 178M	-	-	91.4	88.7	-	-	93.5	91.4
(+ Gold POS)	224M / 178M	-	-	(93.3)	(91.1)	-	-	(93.8)	(92.1)
U-DepP Llama-2-13b-hf*	31.6M / 13.0B	96.3	-	88.9	86.6	95.3	-	92.3	90.1
U-DepP Llama-3.1-8B*	22.0M / 8.05B	98.4	-	90.7	88.4	97.8	-	92.0	90.2
Our methods:									
gpt-4o-mini-2024-07-18	(undisclosed)	99.6	97.7	93.2	91.1	99.5	98.3	94.9	93.3
gpt-4o-2024-08-06	(undisclosed)	99.7	**98.1**	93.7	91.7	**100.0**	98.5	95.2	93.5
gemma-2-2b*	10.4M / 2.62B	99.7	98.0	93.2	91.1	99.9	98.3	94.4	92.7
Qwen2.5-7B*	20.2M / 7.64B	99.5	97.9	93.1	91.0	99.5	98.3	94.7	93.0
Llama-3.1-8B*	21.0M / 8.05B	99.4	97.8	93.4	91.3	**100.0**	98.4	94.8	93.1
gemma-2-9b*	27.0M / 9.27B	**99.8**	**98.1**	**93.8**	**91.9**	**100.0**	**98.6**	**95.5**	**94.1**

Table 3: Evaluation of various models in UD_English-EWT r2.2 and r2.15. Best scores are highlighted in **bold**. The scores for UDPipe 2.0 are taken from its official documentation. The scores for Hexatagger and U-DepPLLaMA are the results of our reproduction experiments. The scores in the row (+ Gold POS) are provided for reference, as they use gold POS tags. The LoRA-SFT models are marked by "*". "†" indicates that the value is estimated from the size of distributed model archive.

Dataset	Baselines: UDPipe 2.0 UPOS	UAS	LAS	Hexatagger(+Gold POS) UAS	LAS	Ours: Monolingual: gemma-2-2b* UPOS	UAS	LAS	Qwen2.5-7B* UPOS	UAS	LAS	Llama-3.1-8B* UPOS	UAS	LAS	gemma-2-9b* UPOS	UAS	LAS	Ours: Multilingual: gemma-2-9b* UPOS	UAS	LAS
ar_padt	97.0	88.1	83.7	86.8 (87.9)	81.8 (83.9)	95.4	87.7	83.1	96.1	88.7	84.3	**97.1**	**90.0**	**85.5**	96.2	89.8	85.4	96.2	89.9	**85.7**
bg_btb	99.3	95.3	92.6	94.7 (95.4)	92.0 (92.4)	99.3	95.3	92.9	99.2	95.2	92.7	99.1	95.7	93.2	**99.5**	**96.9**	**94.7**	**99.5**	**97.0**	**94.8**
ca_ancora	99.2	94.9	93.4	94.8 (95.1)	93.1 (93.7)	98.9	94.5	92.9	99.0	94.7	93.1	99.1	95.1	93.6	**99.3**	**95.7**	**94.5**	**99.3**	**95.7**	94.3
cs_pdt◇	99.4	95.7	94.3	94.3 (94.7)	92.5 (92.9)	99.3	95.1	93.6	99.3	95.2	93.8	99.3	95.5	94.2	99.4	95.7	**94.4**	99.3	94.9	93.3
de_gsd	97.1	89.2	85.5	87.9 (90.2)	83.8 (86.1)	97.0	89.1	85.3	97.1	89.7	85.9	97.1	89.7	86.1	**97.4**	**90.2**	**86.5**	**97.4**	**90.3**	**86.7**
en_ewt	97.5	93.4	91.5	93.5 (93.8)	91.4 (92.1)	98.3	94.4	92.7	98.3	94.7	93.0	98.4	94.8	93.1	**98.6**	**95.5**	**94.1**	**98.6**	**95.5**	**94.1**
es_ancora	99.1	94.0	92.4	93.4 (94.6)	91.6 (92.9)	99.1	93.7	92.0	99.1	94.1	92.6	99.1	94.2	92.7	99.1	**94.6**	**93.2**	**99.2**	**95.1**	**93.7**
fr_gsd	98.5	95.0	93.2	94.4 (96.0)	93.3 (94.6)	98.6	95.6	94.0	98.7	95.9	94.4	98.5	95.6	93.9	98.7	**96.7**	**95.3**	**98.8**	**96.7**	95.2
it_isdt	**98.8**	95.1	93.4	95.7 (96.5)	94.0 (95.2)	98.6	95.6	94.0	98.4	94.8	93.2	98.7	95.8	94.4	98.7	**96.0**	**94.6**	**98.8**	**96.4**	**94.9**
ja_gsd	98.6	95.1	94.2	94.6 (95.2)	93.3 (94.6)	98.7	94.4	93.1	98.8	94.8	93.6	98.5	94.8	93.6	**99.0**	**95.7**	**94.9**	98.9	95.5	94.5
ko_gsd	96.7	88.9	85.2	87.4 (87.9)	83.6 (85.1)	96.5	87.4	83.9	96.8	89.3	86.5	96.9	88.8	85.9	**97.2**	**90.3**	**87.3**	**97.2**	**90.6**	**87.6**
nl_alpino	98.2	94.9	92.9	93.3 (94.1)	90.6 (91.5)	98.1	94.0	91.3	97.4	94.0	91.6	97.3	93.9	91.6	**98.4**	**95.7**	**93.9**	**98.5**	95.6	93.7
no_bokmaal	98.6	94.7	93.4	**95.9** (96.5)	94.3 (94.8)	98.7	94.7	93.5	98.5	94.2	92.9	98.7	95.0	93.8	**98.8**	95.7	**94.8**	98.7	95.5	94.6
ro_rrt	98.1	92.6	89.1	91.9 (92.9)	87.9 (89.0)	98.1	92.5	88.9	97.9	92.7	89.2	97.7	93.1	89.7	98.1	**94.4**	**91.3**	**98.4**	94.3	**91.3**
ru_syntagrus	98.8	94.5	92.3	94.1 (95.1)	91.7 (92.7)	98.8	95.2	93.1	98.9	95.3	93.4	98.9	95.5	93.5	**99.0**	**95.8**	**93.8**	98.6	95.0	92.7
sl_ssj	98.8	94.5	92.9	94.0 (94.8)	92.3 (93.2)	98.5	94.0	92.4	98.7	94.1	92.4	98.9	94.7	93.2	**99.0**	**95.9**	**94.6**	**99.0**	**95.9**	**94.7**
zh_gsdsimp	95.8	86.7	83.6	87.4 (90.1)	84.3 (87.9)	96.4	86.5	83.6	96.5	87.2	84.4	96.8	87.7	84.7	**97.5**	**89.3**	**86.9**	97.3	**89.7**	**87.3**

Table 4: Evaluation results on various UD r2.15 datasets. For each language, best scores among the baselines and our monolingual models are shown in **bold**, with ties and second-best scores underlined. Additionally, scores from our multilingual model that outperform the baselines and monolingual models are also highlighted. The scores for UDPipe 2.0 are taken from its official documentation. The scores for Hexatagger are the results of our reproduction experiments. The scores in the brackets are provided for reference, as they use gold POS tags. LoRA-SFT models are marked by "*". "◇" indicates use of a language-specific pre-trained model in UDPipe 2.0.

4 Experiments

We conducted both supervised fine-tuning (SFT) with Low-Rank Adaptation (LoRA) (Hu et al., 2022) and inference experiments for open LLMs on a high-performance cloud service[5,6]. For OpenAI models, SFT was performed via the official web console[7]. We explored SFT hyper-parameters[8] on the UD_English-EWT r2.15 development set and applied them to all experiments. We used simple TSV recovery process only restores the ID and FORM on a row-by-row basis.

[5] Experiments were conducted on a Google Cloud A2 Ultra instance with 8 × NVIDIA A100 GPUs (80GB each), 96-core Intel Xeon CPUs @ 2.20GHz, 1,360GB RAM, and 5TB of SSD storage. The software environment included: Ubuntu 22.04, CUDA 12.1, Python 3.11.9, PyTorch 2.5.1, Transformers 4.49.0, TRL 0.15.2, PEFT 0.14.0, OpenAI 1.68.2, Unsloth 2025.3.18, and vLLM 0.7.2.

[6] The implementation used in the experiments is available on GitHub. https://github.com/megagonlabs/llmpp

[7] https://platform.openai.com/docs/guides/fine-tuning. The cost of fine-tuning the en_ewt-r2.15 train set for 2 epochs was about $52 for gpt-4o-mini and about $430 for gpt-4o.

[8] **Open LLMs:** num_epochs=3, max_seq_length=8192, lr=3e-4, lr_scheduler=cosine_with_min_lr, min_lr=0.1, LoRA: r=8, dropout=0.05, target_modules="all-linear" (embedding layers excluded). **OpenAI:** num_epochs=2, max_seq_length=8192, lr=default.

4.1 Dataset

We mainly used Universal Dependencies treebanks r2.15. For `UD_English-EWT` (`en_ewt`), we also used r2.2 for comparison with baseline methods.

For monolingual SFT. We used datasets for the following 17 languages to evaluate the parsing accuracy for each language: `ar_padt`, `bg_btb`, `ca_ancora`, `cs_pdt`, `de_gsd`, `en_ewt`, `es_ancora`, `fr_gsd`, `it_isdt`, `ja_gsd`, `ko_gsd`, `nl_alpino`, `no_bokmaal`, `ro_rrt`, `ru_syntagrus`, `sl_ssj`, and `zh_gsdsimp`. Statistics for each dataset are provided in the Appendix A.

For multilingual SFT. To train a multilingual parsing model, we constructed a new dataset by gathering training sets from the datasets for the 17 languages above. To reduce training time and costs, we downsampled `cs_pdt` and `ru_syntagrus` by 17% to balance them with other language datasets. The final training data consisted of 182,255 sentences and 3,889,494 tokens, which was used to train a multilingual model (denoted as `17_multi` below). Additionally, we evaluated the following 10 language datasets not included in the multilingual training data: `el_gdt`, `he_htb`, `hi_hdtb`, `hu_szeged`, `id_gsd`, `pt_gsd`, `sv_talbanken`, `tr_imst`, `vi_vtb`, and `zh_gsd`.

4.2 Baseline methods

We compared our method against three strong baselines: `UDPipe 2.0`[9] (Straka, 2018), `Hexatagger`[10] (Amini et al., 2023), and `U-DepPLLaMA`[11] (Hromei et al., 2024). The reported scores for `UDPipe 2.0` were taken from its official documentation, while the results for `Hexatagger` and `U-DepPLLaMA` were reproduced in our environment using their publicly available implementations[12]. For Hexatagger, we report the accuracy under the setting that does not use gold POS tags (the accuracy when using gold POS tags is also provided as a reference).

4.3 Evaluation of various models

We conducted a comparative evaluation of baselines and our SFT method with various LLMs including `gpt-4o-mini`[13], `gpt-4o`[14], `gemma-2-2b`[15], `Qwen2.5-7B`[16], `Llama-3.1-8B`[17], and `gemma-2-9b`[18] using `UD_English-EWT`.

Results are summarized in Table 3. Overall, `gemma-2-9b` achieved the highest performance, followed closely by `gpt-4o`. Beyond Table 3, circular references were rare, with only 3 cases found in the output of `Qwen2.5-7B`, and no multiple roots found in the output of either model on the test set. These results highlight the favorable cost-performance trade-off of open LLMs, leading us to exclude OpenAI models from the subsequent experiments.

From the perspective of model parameter size, the pre-trained LLMs used in this experiment contain 2.6 to 9.3 billion parameters, which is several tens of times larger than the bert-base models used in the baselines. However, the numbers of trainable LoRA parameters are relatively small, ranging from 10 to 27 million. This suggests that LoRA-based SFT effectively leverages the capabilities of large, fixed-weight networks for dependency parsing tasks. Moreover, the parsing accuracy appears to depend on the number of pre-training parameters, given a certain number of trainable parameters.

4.4 Evaluation in 17 languages

Monolingual SFT. We evaluated the proposed method in 17 UD languages to assess its monolingual performance. Table 4 shows the detailed results for each language.

The proposed method achieved the highest LAS in all 17 languages, and the highest UAS in 16, except Norwegian, indicating its overall effectiveness. Among the open LLMs, `gemma-2-9b` demonstrated consistently strong performance, ranking first in 16 languages with the sole exception of Arabic. Due to lower tokenization efficiency in `ar_padt` compared to other languages, the LLMs occasionally failed to output the complete analysis results within the available context length, particularly for long sentences. However, the `Llama-3.1` tokenizer was approximately 20% more efficient at tokenizing Arabic text than the `gemma-2` and `Qwen2.5` tokenizers, which contributing to higher accuracy. This indicates a trade-off between efficiency and accuracy: as the number of Chain-of-Thought steps increases, the allowable input sentence length becomes more constrained by the maximum context length of the LLMs.

[9] https://ufal.mff.cuni.cz/udpipe/2/models

[10] https://github.com/rycolab/parsing-as-tagging

[11] https://github.com/crux82/u-deppllama

[12] The publicly available implementation of U-DepPLLaMA uses the precision as the accuracy, but we followed the UD convention and used F1-measure as the accuracy.

[13] https://platform.openai.com/docs/models/gpt-4o-mini

[14] https://platform.openai.com/docs/models/gpt-4o

[15] https://huggingface.co/google/gemma-2-2b

[16] https://huggingface.co/Qwen/Qwen2.5-7B

[17] https://huggingface.co/meta-llama/Llama-3.1-8B

[18] https://huggingface.co/google/gemma-2-9b

Dataset	gemma-2-9b 17-multi		
	UPOS	UAS	LAS
el_gdt	93.1	92.4	86.3
he_htb	90.1	83.3	70.0
hi_hdtb	76.9	72.9	55.2
hu_szeged	87.1	85.7	75.1
id_gsd	86.5	82.4	66.6
pt_gsd	92.2	86.4	78.0
sv_talbanken	92.9	89.6	82.1
tr_imst	79.2	67.9	53.5
vi_vtb	81.8	72.3	57.7
zh_gsd	97.2	89.5	87.2

Table 5: Evaluation results of our multilingual model on UD r2.15 datasets not used for training.

Multilingual SFT. An additional advantage of the proposed method is its compatibility with multilingual training. The `gemma-2-9b 17-multi` model achieved comparable or higher accuracy than its monolingual counterparts, except in Czech and Russian, likely due to the down-sampling.

Table 5 shows the evaluation results on 10 languages not included in the training data for `17_multi`. Among these, Greek and Swedish exhibited relatively high performance, indicating successful generalization from typologically or linguistically related languages. This highlights the model's ability to generalize across languages, a key strength of our method.

4.5 Analysis

Error analysis. We conducted an error analysis on Simplified Chinese, which showed the lowest UAS in monolingual evaluation. Errors were primarily concentrated in nouns (27.8%), verbs (24.8%), and punctuation marks (16.1%) for `gemma-2-9b`. Most of these errors occurred in sentences containing multiple independent clauses—a structure more frequent in Chinese than in many other languages. Due to the structural parallelism among these clauses, an output that differs from the gold annotation is not necessarily incorrect.

Figure 2 illustrates an example that includes noun, verb, and punctuation errors, highlighting the challenge of analyzing paratactic structures with minimal syntactic markers.

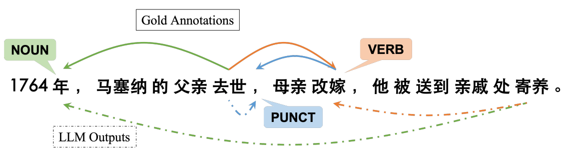


Figure 2: An example illustrating common errors for Chinese dependency parsing.

Performance in other tasks. An LLM fine-tuned for dependency parsing clearly performs worse on other tasks, even if the base model has been instruction-tuned. This performance degradation in general tasks may be mitigated or even reversed by fine-tuning the model on the dependency parsing task simultaneously with other instruction-tuning datasets (Asada and Miwa, 2025); however, experimental verification remains a future challenge.

4.6 Unimplemented UD tasks

Tokenization. In the early stages of this work, we evaluated LLM-based word segmentation by inserting a word segmentation step at the beginning of step-by-step instructions. However, particularly for Japanese, the segmentation accuracy was significantly lower than that of commonly used morphological analyzers. To address this issue, full-parameter LLM training, including the word embedding layer, on large-scale training data would be necessary. However, the associated cost could be several orders of magnitude higher than that of LoRA-SFT, which is employed in this study. Thus, an efficient method for training word segmentation criteria tailored to LLMs is still required.

Lemmatization. Lemmatization has traditionally relied on dictionaries and heuristic rules; however, end-to-end approaches have recently gained traction (Qi et al., 2020). LLMs may also be capable of effectively selecting the appropriate normalized form from a range of synonymous expressions or character variants by leveraging the knowledge acquired through large-scale pre-training, although this remains to be empirically validated.

Morphological features. The Universal Features[19] inventories over 200 lexical and inflectional features designed to classify word properties. Decoder-based classifiers offer significant advantages for simultaneously classifying this large number of features, whereas using generative models such as LLMs is relatively inefficient.

5 Conclusions

We proposed a novel step-by-step prompting strategy for LLM-based dependency parsing using a simple tabular format, achieving improved output validity and parsing accuracy across 17 languages. Multilingual SFT often outperformed monolingual models and generalized well to unseen languages.

[19] https://universaldependencies.org/u/feat/

Acknowledgments

This work was conducted as part of a collaborative research project between Recruit Co., Ltd. and the National Institute for Japanese Language and Linguistics. We are grateful to all those involved in the management and support of this project. We would also like to express our sincere gratitude to Yuji Matsumoto of RIKEN AIP for his valuable advice from the early stages of this research. Finally, we thank the anonymous reviewers for their constructive and detailed comments.

References

Afra Amini, Tianyu Liu, and Ryan Cotterell. 2023. Hexatagging: Projective dependency parsing as tagging. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1453–1464, Toronto, Canada. Association for Computational Linguistics.

Masaki Asada and Makoto Miwa. 2025. Improving relation extraction by sequence-to-sequence-based dependency parsing pre-training. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7099–7105, Abu Dhabi, UAE. Association for Computational Linguistics.

Xuefeng Bai, Jialong Wu, Yulong Chen, Zhongqing Wang, and Yue Zhang. 2023. Constituency parsing using llms. *Preprint*, arXiv:2310.19462.

Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2025. Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57(6):1–39.

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.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324, Brussels, Belgium. Association for Computational Linguistics.

Claudiu Daniel Hromei, Danilo Croce, and Roberto Basili. 2024. U-deppllama: Universal dependency parsing via auto-regressive large language models. *Italian Journal of Computational Linguistics*, 10.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Chunpeng Ma, Lemao Liu, Akihiro Tamura, Tiejun Zhao, and Eiichiro Sumita. 2017. Deterministic attention for sequence-to-sequence constituent parsing. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3237–3243. AAAI Press.

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 Twelfth Language Resources and Evaluation Conference*, pages 4034–4043. European Language Resources Association.

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

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28, page 2773–2781. Curran Associates, Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22.

A Dataset Statistics

Statistics for the Universal Dependencies treebanks used in the experiments are shown in Table 6.

Language	Dataset	Train number of sentences	Train number of tokens	Dev number of sentences	Dev number of tokens	Test number of sentences	Test number of tokens
UD r2.2:							
English	en_ewt	12,543	204,585	2,002	25,148	2,077	25,096
UD r2.15:							
Arabic	ar_padt	6,075	223,881	909	30,239	680	28,264
Bulgarian	bg_btb	8,907	124,336	1,115	16,089	1,116	15,724
Catalan	ca_ancora	13,123	429,578	1,709	58,073	1,846	59,610
Czech	cs_pdt	68,491	1,173,285	9,270	159,283	10,146	173,918
German	de_gsd	13,814	263,791	799	12,480	977	16,498
English	en_ewt	12,544	204,579	2,001	25,149	2,077	25,094
Spanish	es_ancora	14,287	453,039	1,654	53,476	1,721	53,622
French	fr_gsd	14,450	354,652	1,476	35,721	416	10,018
Italian	it_isdt	13,121	276,014	564	11,907	482	10,417
Japanese	ja_gsd	7,050	168,333	507	12,287	543	13,034
Korean	ko_gsd	4,400	56,687	950	11,958	989	11,677
Dutch	nl_alpino	12,289	186,027	718	11,541	596	11,046
Norwegian	no_bokmaal	15,696	243,886	2,409	36,369	1,939	29,966
Romanian	ro_rrt	8,043	185,125	752	17,073	729	16,324
Russian	ru_syntagrus	69,630	1,204,640	8,906	153,325	8,800	157,718
Slovenian	sl_ssj	10,903	215,155	1,250	26,500	1,282	25,442
Simplified Chinese	zh_gsdsimp	3,997	98,616	500	12,663	500	12,012

Table 6: Statistics of Universal Dependencies treebanks used in SFT experiments.

B Contamination Verification

A major concern in LLM-based evaluation is the contamination of testing data (Shokri et al., 2017; Das et al., 2025). To address this, we employed two diagnostics: (1) observing learning curves on `UD_English-EWT r2.15` to detect unusually high initial performance, and (2) comparing fine-tuning results using training-only vs. training + test data. Evaluation results for contamination verification are presented below.

Learning curves. Prior to the analysis, the learning curves of token recall (Figure 3) show that `gpt-4o-mini` is able to generate outputs with correct formats in very early stage, while other models need to be trained, and the learning curves of token recall after recovery (Figure 4) indicates our simple recovery algorithm works effectively.

For the learning curves of UPOS recall (Figure 5), the similarity between Figure 4 and Figure 5 suggests that the UPOS tagging task is one of the abilities that has been acquired in advance in these LLMs, which is also indicated by the high initial accuracy of the precision-based learning curves of UPOS in Figure 8.

In contrast, the gradual learning curves for HEAD and DEPREL identification (Figures 6 and 7) indicate the necessity of SFT for learning the knowledge for dependency parsing.

Overall, we conclude that the tested LLMs do not exhibit potential contamination in syntactic head and relational label identification tasks, despite possible prior exposure to the UPOS tagging task.

Effect of additional training on test set. In Table 7, when the testing data was included in training, all models naturally achieved certain accuracy gains (>+0.7 for UPOS, >+2.3 for LAS). This indicates a low possibility of contamination for the test set of `UD_English-EWT r2.15`.

	UPOS train	UPOS train + test (± diff)	LAS train	LAS train + test (± diff)
`gpt-4o-mini-2024-07-18`	98.3	99.3 (+1.0)	93.2	97.1 (+3.9)
`Qwen2.5-7B`	97.1	98.5 (+1.4)	88.7	93.6 (+4.9)
`Llama-3.1-8B`	97.2	98.7 (+1.5)	88.9	94.1 (+5.2)
`gemma-2-9b`	98.0	98.7 (+0.7)	91.7	94.0 (+2.3)

Table 7: Difference of UPOS and LAS scores between SFT on training data only and training + test data for `UD_English-EWT r2.15`.

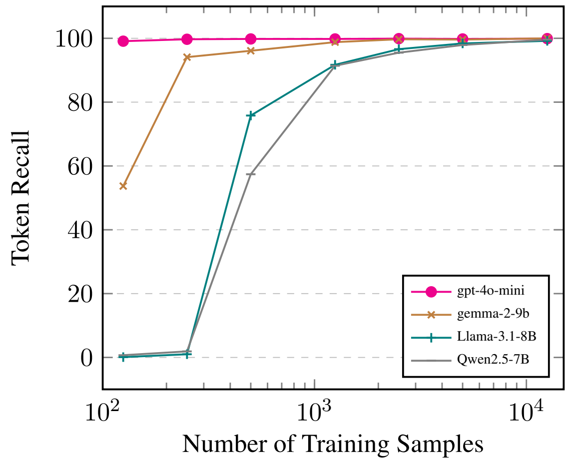


Figure 3: Learning curve - token recall.

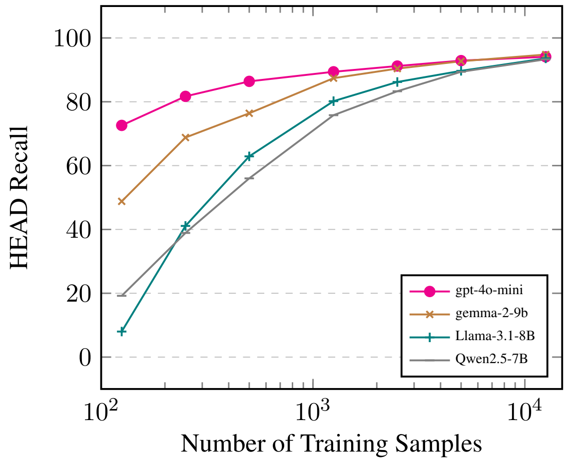


Figure 6: Learning curve - HEAD recall.

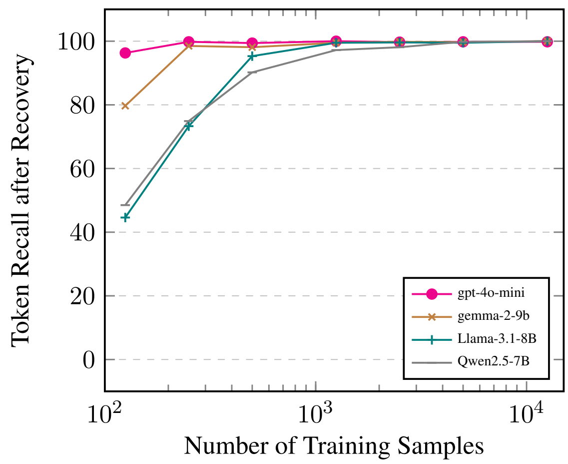


Figure 4: Learning curve - token recall after recovery.

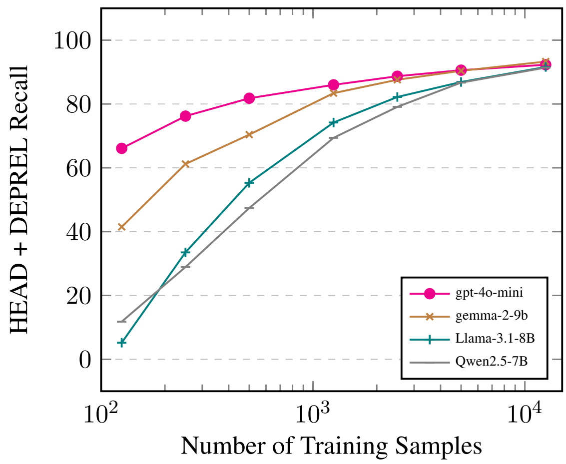


Figure 7: Learning curve - HEAD+DEPREL recall.

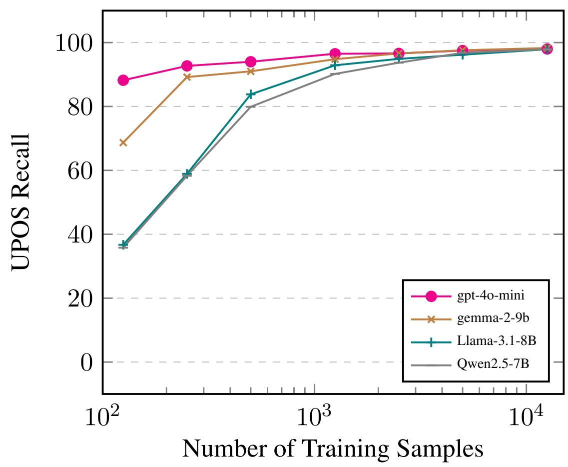


Figure 5: Learning curve - UPOS recall.

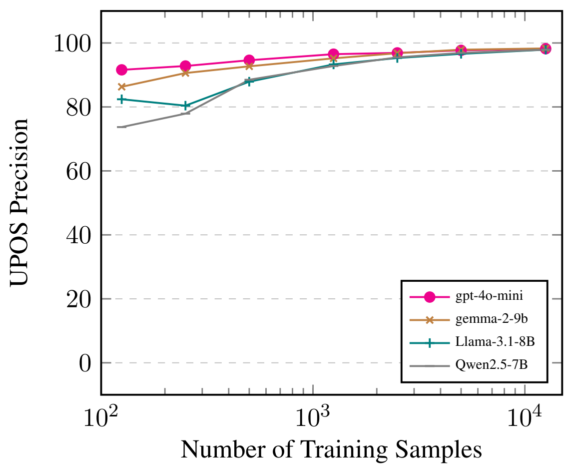


Figure 8: Learning curve - UPOS precision.

C Prompt Examples

The examples of Chain-of-Thought instruction prompts used in the experiments are shown in Figure 9 and Figure 10.

```
[system]
You are an English linguist and specialize in
English dependency analysis based on Universal
Dependencies.

[user]
We will now perform dependency parsing on English
sentence. After splitting the input sentence into
words as shown below, create a TSV with five fields:
word index from 1 to 6 + word +  the Universal
Dependencies part of speech + the dependent word
index + the Universal Dependencies relation.
However, for the word that is the main predicate of
the sentence, the dependent word index should be 0.

input sentence:
I'm just speculating now.

words:
I
'm
just
speculating
now
.

[assistant]
1 I  PRON 4 nsubj
2 'm  AUX 4 aux
3 just  ADV 4 advmod
4 speculating  VERB 0 root
5 now ADV 4 advmod
6 .  PUNCT 4 punct
```

Figure 9: Example for 1-step parsing instruction.

```
[system]
You are an English linguist and specialize in
English dependency analysis based on Universal
Dependencies.

[user]
We will now perform dependency parsing on English
sentence. After splitting the input sentence into
words as shown below, execute following three
tasks:

- Task 1
Create a TSV with three fields: word index from 1
to 6 + word + part of speech.

- Task 2
Add a field for the dependent word indexes to
each row to the output of Task 1. However, for
the word that is the main predicate of the
sentence, the dependent word index should be 0.

- Task 3
Add a field for the Universal Dependencies
relation labels to the output of Task 2.

input sentence:
I'm just speculating now.

words:
I
'm
just
speculating
now
.

[assistant]
- Task 1
1 I  PRON
2 'm  AUX
3 just  ADV
4 speculating  VERB
5 now ADV
6 .  PUNCT

- Task 2
1 I  PRON 4
2 'm  AUX 4
3 just  ADV 4
4 speculating  VERB 0
5 now ADV 4
6 .  PUNCT 4

- Task 3
1 I  PRON 4 nsubj
2 'm  AUX 4 aux
3 just  ADV 4 advmod
4 speculating  VERB 0 root
5 now ADV 4 advmod
6 .  PUNCT 4 punct
```

Figure 10: Example for 3-step parsing instruction.

CCG Revisited: A Multilingual Empirical Study of the Kuhlmann-Satta Algorithm

Paul He and **Gerald Penn**
Department of Computer Science
University of Toronto
{hepaul, gpenn}@cs.toronto.edu

Abstract

We revisit the polynomial-time CCG parsing algorithm introduced by Kuhlmann and Satta (2014), and provide a publicly available implementation of it. We evaluate its empirical performance against a naive CKY-style parser across the Parallel Meaning Bank (PMB) corpus. While the fast parser is slightly slower on average, relative to the size of the PMB, the trend improves as a function of sentence length, and the PMB is large enough to witness an inversion. Our analysis quantifies this crossover and highlights the importance of derivational context decomposition in practical parsing scenarios.

1 Introduction

Parsing with Combinatory Categorial Grammar (CCG) occupies a crucial space in natural language processing, balancing linguistic expressivity with computational tractability. CCG's position at the bottom of the mildly context-sensitive hierarchy enables the analysis of some long-range dependencies and cross-serial constructions (Kuhlmann et al., 2018; Steedman, 2000), but even an $\mathcal{O}(n^6)$ complexity places it out of reach of several large-scale applications.

Theoretically, CCG parsing was shown to be polynomial-time in sentence length by Vijay-Shanker and Weir (1993), with a worst-case complexity of $\mathcal{O}(n^6)$. Kuhlmann and Satta (2014) much later introduced a simplified algorithm in terms of *derivation contexts* to handle deep stacks of arguments. Through an operational lens, the existence of derivation contexts is the reason that CCG-parsing is polynomial-time.

Yet, CCG's complexity is more subtle than it appears. Kuhlmann et al. (2018) proved that CCG parsing is exponential in the combined size of the grammar and input, contrasting sharply with formalisms like Tree-Adjoining Grammar (TAG), in which parsing remains polynomial in both. This raises important questions about the practical utility of polynomial-time CCG algorithms: how often is an innovation like derivation contexts actually triggered in practice? Do real-world grammars ever blow up? Computational research on CCG, after all, has been driven for two decades by corpora with context-free backbones that use CCG-style notation.

To address this gap, we provide the first empirical evaluation of the algorithm introduced by Kuhlmann and Satta (2014). We compare it against a closely related CKY-style parser that lacks this innovation, widely used in existing CCG systems despite its exponential worst-case runtime. While it is true that traditional parsing algorithms have largely been supplanted by sequential supertagging methods in the research literature, in our experience, a stable and efficient reference algorithm is still important for experimental research in parsing, because it provides all possible parses (not just the most likely as scored by a language model), as well as a filter for derivability that conditions the evaluation statistics of supertagger outputs.

Our contributions here are threefold:

- We make a Python implementation of the Kuhlmann-Satta parser publicly available and evaluate it on the Parallel Meaning Bank (PMB) corpus (Abzianidze et al., 2017), spanning over 12,000 CCG-annotated sentences in English, German, Italian, and Dutch.
- We analyze runtime and rule activation across varying levels of derivational complexity, identifying conditions under which the polynomial parser provides real advantages.
- We show that while the naive parser is faster on average, the Kuhlmann-Satta parser is asymptotically safer, not just on a theoretical basis, but in measurable, practical terms, outperforming its baseline on structurally deep

Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), pages 20–25
August 26, 2025 ©2025 Association for Computational Linguistics

inputs sampled from the PMB and with consistent runtime stability.

This work thus bridges the gap between CCG parsing theory and empirical behavior, providing a grounded assessment of polynomial-time parsing in NLP pipelines.

2 Background

In this section, we introduce some background and terminology that we will refer to across this paper and our implementation.

A CCG Category Categories in Combinatory Categorial Grammar (CCG) are either **atomic elements** or **unary functions** that take a category as input and return another category. More formally, the set of categories $\mathcal{C}$ is defined inductively as follows:

- $A \subset \mathcal{C}$ where A is a set of **atomic categories** (e.g., S, NP, N)
- If $X, Y \in \mathcal{C}$, then $(X/Y) \in \mathcal{C}$ and $(X\backslash Y) \in \mathcal{C}$.

where the slash notation indicates the directionality of functional application. Categories are interpreted as unary functions applied from right to left; for example, S/NP/NP is parsed as (S/NP)/NP, indicating a function that takes two NP arguments in sequence.

Arity Bound As per Kuhlmann and Satta (2014), we define c_G as a constant representing the maximum arity permitted in a derivation. It is computed as:

$$c_G \geq \max\{\ell, a + d\} \tag{1}$$

where ℓ is the maximum arity of a lexical entry, a being the maximum arity of any argument type in the grammar, and d being the maximum composition degree allowed. This bound ensures that derivations exceeding a specified complexity are decomposed into reusable subcontexts, maintaining polynomial runtime.

Equivalence in the Limit Let c_G be the arity bound used by the Kuhlmann-Satta parser. Then for any input sentence s, the set of derivable categories produced by the Kuhlmann-Satta parser with $c_G \to \infty$ is equal to that produced by the naive CKY parser.

$$\lim_{c_G \to \infty} \text{Parse}_{\text{KS}}(s; c_G) = \text{Parse}_{\text{CKY}}(s) \tag{2}$$

This follows from the fact that context decomposition is only triggered when the arity of intermediate categories exceeds c_G. In the limit, such decomposition never occurs, and the two parsers are functionally identical.

Chart Items Let w be an input string of length $n \in \mathbb{N}$, and let $0 \leq i < j \leq n$. In CKY-style CCG parsing, chart items are of the form $\langle X, i, j \rangle$ where X is a category that spans the substring $w[i : j]$[1]. While standard application rules suffice for shallow derivation, deeper constructions involving nested argument stacks or crossing dependencies can cause exponential chart growth. To address this, Kuhlmann and Satta (2014) introduce derivation context items

$$\langle /Y, \beta, i, i', j', j \rangle \tag{3}$$

representing a partial derivation, meaning, if a constituent of type X/Y spans $w[i' : j']$, $i', j' < n$ then the entire expression of type $X\beta$ can be derived over $w[i : j]$. Here, β is the residual argument stack, and c_G is the arity bound as defined earlier.

3 Experiments

3.1 Setup and Dataset

Dataset We evaluate the empirical performance of the Kuhlmann-Satta parser on the Parallel Meaning Bank (PMB) corpus (Abzianidze et al., 2017), which provides aligned CCG derivations across English (11,950 sentences), German (3,128), Italian (1,928), and Dutch (1,494). Each sentence is parsed using both the naive CKY-style parser and the polynomial-time parser from Kuhlmann and Satta (2014). We compare runtime, chart size, rule usage, and correctness.

To avoid bias from non-standard derivation rules, we exclude sentences requiring the unary `lx` rule as illustrated in Table 1. This is a special case found in both PMB and CCGbank derivation files that introduces a phrasal NP directly from an N (e.g., N $\Rightarrow$ NP), without any overt determiner. This rule is not part of standard CCG combinatorics, and is essentially unsupportable within a purely CCG-based algorithm such as Kuhlmann-Satta.

Parser Architecture and Implementation Details Both the naive and Kuhlmann-Satta parsers are implemented in Python 3. We designed a unified chart data structure to facilitate fair comparison between

[1] $w[i : j]$ returns the substring of w from indices i to $j - 1$.

Language	LX Sentences	Non-LX Sentences	LX %
EN	6219	5731	52.05
DE	1535	1593	49.07
IT	959	969	49.74
NL	710	784	47.52

Table 1: Proportion of sentences using the non-standard lx rule per language. English and Dutch show the highest occurrence. These were excluded from runtime comparisons.

Language	Context Edges	Other Edges	Context %
EN	6510	87704	6.9
DE	1429	15734	8.3
IT	371	9967	3.6
NL	888	10159	8.0

Table 2: Proportion of **chart edges** using derivation context rules (Kuhlmann edges). While English produces significantly more edges, the relative context usage varies across languages.

Language	With Context	Without Context	Context %
EN	1133	4589	19.8
DE	300	1293	18.8
IT	97	872	10.0
NL	161	623	20.5

Table 3: Percentage of **sentences** using at least one derivation context rule (Kuhlmann item). Most sentences do not trigger context decomposition.

the two systems, ensuring that tokenization, category assignment, and lexical rule application are identical across runs.

The naive parser implements a standard bottom-up CKY strategy with forward, backward, and crossed composition rules. It constructs all derivable categories over input spans using unrestricted functional application, subject only to chart cell boundaries.

The Kuhlmann-Satta parser extends this by introducing derivation context items as described in Section 2. These items are constructed and recombined according to Rules (1) through (6) from the original paper (Kuhlmann and Satta, 2014), including nested context recombination and contextual substitution. Arity bounds c_G are computed dynamically per input, using the maximum arity observed in the lexicon and a fixed composition degree.

Runtime was measured using Python's built-in timing module, with warm-up excluded. The code for both parsers and all experimental scripts will be made available as open-source code.

3.2 How Often is Context Decomposition Needed?

Table 2 shows the number of chart edges generated using context-based rules ($\langle /Y, \beta, \cdots \rangle$) across languages. English produces an order of magnitude more chart items and context applications than other languages, likely reflecting greater average syntactic depth in this corpus slice.

At the sentence level (Table 3), we find that roughly 20% of English examples require at least one derivation context item. In German, Italian, and Dutch, fewer than 10% of sentences trigger decomposition. These results suggest that derivation context rules are sparsely activated overall but concentrated in a meaningful subset of complex examples.

3.3 Runtime Behavior and Inversion

Figure 2 presents raw runtimes over 5,000 sentences, sorted by sentence index. The fast parser (blue) maintains a steady runtime across the corpus, while the naive parser (orange) exhibits increasing variance and high-runtime spikes—suggesting exponential blowups on structurally deep derivations.

Figure 3 shows moving averages of runtime. We observe a "runtime inversion" trend: initially, the naive parser is faster, but as derivation complexity increases, its runtime begins to match or exceed the fast parser. The fast parser exhibits behavior consistent with a polynomial-time bound.

3.4 Speedup Distribution

Figure 4 shows the distribution of relative speedups $T_{\text{naive}}/T_{\text{fast}}$. The average speedup is $0.81\times$, suggesting the fast parser is slower overall—but the long tail on the right includes examples with speedups exceeding $10\times$.

This suggests that while derivation contexts are overhead on average, they yield significant efficiency gains in high-complexity cases, where redundant recursive expansion is avoided.

4 Analysis and Discussion

Further analysis of our results Let $T_{\text{naive}}(n)$ denote the runtime of the naive CKY-style parser for a sentence of length n, and let $T_{\text{fast}}(n)$ be the runtime of the Kuhlmann-Satta parser on the same input. Let $|C_n|$ represent the number of chart items produced during parsing.

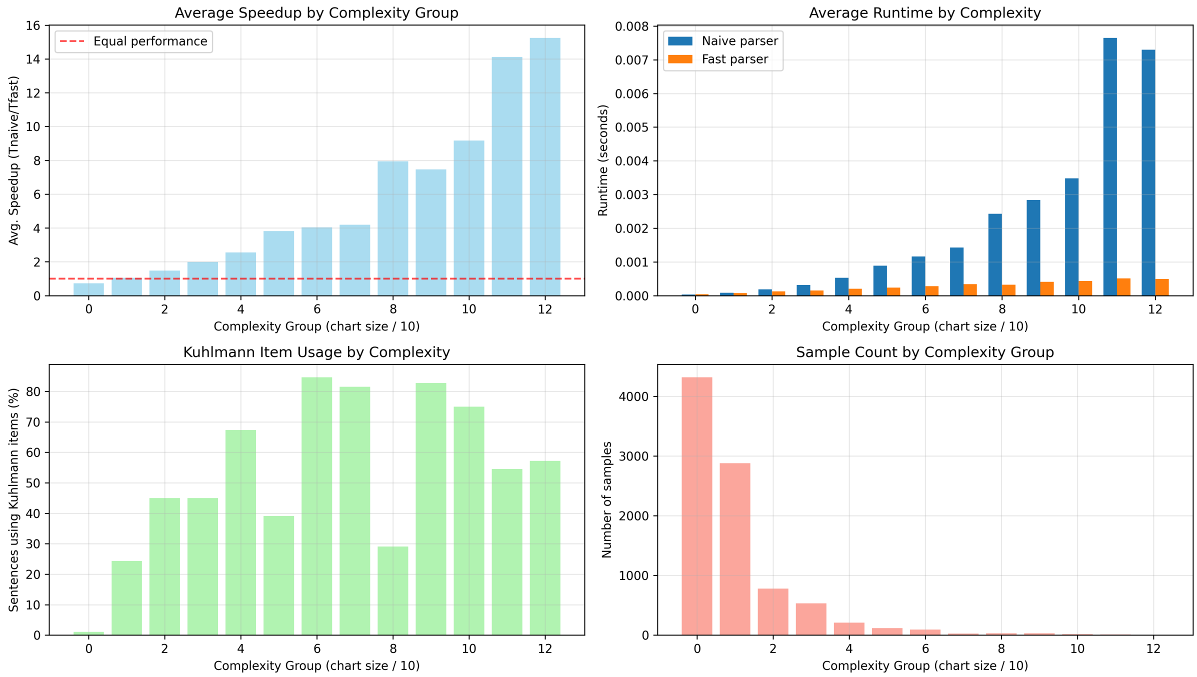


Figure 1: Analysis of parser performance as a function of sentence complexity (measured via chart size in bins of 10). Top-left: Average speedup ($T_{\text{naive}}/T_{\text{fast}}$); a crossover is visible between groups 0 and 1. Top-right: Absolute runtimes per parser. Bottom-left: Percentage of sentences per bin that used Kuhlmann-Satta derivation contexts. Bottom-right: Sample size per bin.

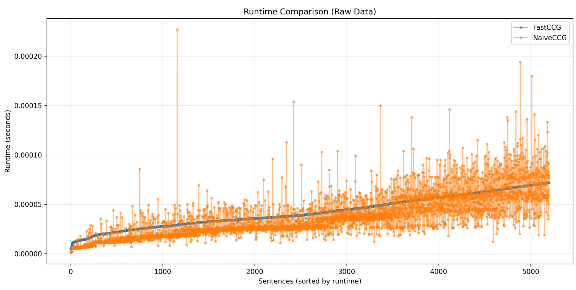


Figure 2: Raw runtime comparison across 5,000 examples. The naive parser is faster on average, but shows instability and many outliers.

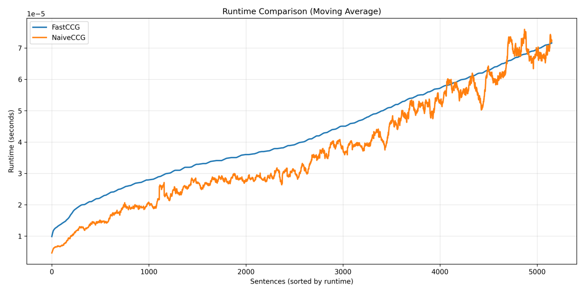


Figure 3: Smoothed runtime comparison. A trend toward inversion is visible: the naive parser is faster early, but the fast parser catches up as complexity rises.

In pathological cases, we empirically observe:

$$T_{\text{naive}}(n) \in \mathcal{O}(2^{|C_n|}), \quad T_{\text{fast}}(n) \in \mathcal{O}(n^6 \cdot g(c_G)) \tag{4}$$

where $g(c_G)$ accounts for grammar-dependent context handling.

To study the crossover empirically, we group sentences by chart size into discrete bins:

$$\mathcal{B}_k = \{n : |C_n| \in [10k, 10(k+1))\} \tag{5}$$

We define average runtimes over each group as:

$$\overline{T}_{\text{naive}}(k) = \frac{1}{|\mathcal{B}_k|} \sum_{n \in \mathcal{B}_k} T_{\text{naive}}(n) \tag{6}$$

$$\overline{T}_{\text{fast}}(k) = \frac{1}{|\mathcal{B}_k|} \sum_{n \in \mathcal{B}_k} T_{\text{fast}}(n) \tag{7}$$

and compute the relative speedup as:

$$S_k = \frac{\overline{T}_{\text{naive}}(k)}{\overline{T}_{\text{fast}}(k)} \tag{8}$$

We observe that $S_k < 1$ for $k = 0$, indicating that the naive parser is faster on simple sentences. However, $S_k > 1$ for all $k \geq 1$, with S_k increasing

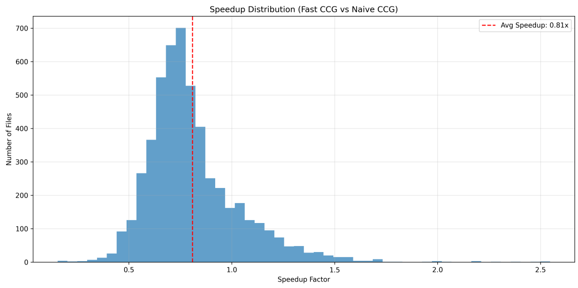


Figure 4: Speedup distribution (fast vs. naive parser). While the fast parser is slower on average, many outliers show large relative gains.

sharply for higher k. This identifies a crossover point near chart size $|C_n| \approx 10$, after which the fast parser becomes consistently preferable.

Moreover, the proportion of sentences that utilize Kuhlmann-Satta derivation contexts (items of the form $\langle /Y, \beta, i, i', j', j \rangle$) increases with k, from 1.0% in group 0 to over 80% by group 6. This strongly supports the theoretical intuition that derivation contexts are unnecessary for simple inputs, but crucial for more complex constructions.

Figure 1 summarizes these trends: the top-left panel reveals a clear crossover in speedup at $k = 1$, the top-right panel shows that naive parsing time grows rapidly with complexity, while the fast parser remains stable, the bottom-left panel shows the increasing use of derivation contexts with complexity, and the bottom-right panel reflects sample distribution, affirming the statistical strength of early bins.

These results validate the theoretical runtime guarantees of Kuhlmann and Satta (2014) and demonstrate that their parser provides real-world benefits for high-complexity sentences.

Context Use and Syntactic Depth Our results suggest a strong relationship between derivational complexity and the activation of derivation contexts. Let $D(s)$ denote the syntactic depth of a sentence s, and let $\Pr[\text{context} \mid s]$ denote the probability that at least one derivation context (i.e., Kuhlmann item) is used in parsing s. Then we conjecture:

$$\Pr[\text{context} \mid s] \propto D(s) \tag{9}$$

While we do not measure $D(s)$ directly, we use chart size $|C_s|$ as a proxy for derivational complexity. Empirically, Figure 1 (bottom-left) shows that the proportion of sentences using derivation contexts increases with $|C_s|$ on average, though not strictly monotonically. We attribute deviations at high complexity to data sparsity (cf. bottom-right), and to the fact that high chart complexity may also arise from breadth (many constituents) rather than depth (nested arguments).

5 Conclusion

We have presented the first empirical evaluation of the Kuhlmann-Satta polynomial-time CCG parser, comparing it to a naive CKY-style baseline across multiple languages in the Parallel Meaning Bank corpus. While the naive parser is faster on average, we find that the Kuhlmann-Satta algorithm offers clear advantages on structurally complex inputs, where it avoids exponential blowup within a reasonable overhead and maintains stable runtime behavior.

Our results validate the theoretical strengths of context decomposition in CCG parsing, and clarify when polynomial-time methods are practically beneficial. This work bridges the gap between parsing theory and real-world efficiency, demonstrating that asymptotic safety is achievable without sacrificing empirical utility.

Looking ahead, the Kuhlmann-Satta parser could play a key role in scaling symbolic parsers for tasks like grammar induction, real-time parsing, and multilingual NLP. Further research might explore adaptive strategies for setting arity bounds, tighter memory constraints, and hybrid approaches that combine polynomial safety with CKY-style heuristics in simpler cases.

6 Limitations and Future Work

While our results validate the theoretical runtime bounds, future work can explore: 1. grammar-aware heuristics for setting c_G dynamically, 2. hybrid parsers that fall back to CKY-style rules when contexts are overkill, 3. optimizing the memory footprint of context items. 4. extending this parser to freer-word-order languages.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Pa-*

pers, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Marco Kuhlmann and Giorgio Satta. 2014. A new parsing algorithm for Combinatory Categorial Grammar. *Transactions of the Association for Computational Linguistics*, 2:405–418.

Marco Kuhlmann, Giorgio Satta, and Peter Jonsson. 2018. On the complexity of CCG parsing. *Computational Linguistics*, 44(3):447–482.

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

K Vijay-Shanker and David J. Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

High-Accuracy Transition-Based Constituency Parsing

John Bauer
HAI
Stanford University
horatio@cs.stanford.edu

Christopher D. Manning
Linguistics & Computer Science
Stanford University
manning@stanford.edu

Abstract

Constituency parsers have improved markedly in recent years, with the F1 accuracy on the venerable Penn Treebank reaching 96.47, half of the error rate of the first transformer model in 2017. However, while dependency parsing frequently uses transition-based parsers, it is unclear whether transition-based parsing can still provide state-of-the-art results for constituency parsing. Despite promising work by Liu and Zhang in 2017 using an in-order transition-based parser, recent work uses other methods, mainly CKY charts built over LLM encoders. Starting from previous work, we implement self-training and a dynamic oracle to make a transition-based constituency parser, and test it on seven languages. Using Electra embeddings as the input layer on Penn Treebank, with a self-training dataset built from Wikipedia, our parser achieves a new SOTA F1 of 96.61.

1 Introduction

This work examines whether it is still possible to build a state of the art constituency parser using transition-based parsing.

Recent years have seen strong progress in both dependency and constituency parsers. Models for both of these tasks have progressed from using categorical features in PCFG models or shift-reduce parsers, to word embeddings, to transformers. Upgrading to a transformer can have a dramatic effect on the quality of the results (Nguyen et al., 2021; Vaswani et al., 2017).

While transition-based parsing is a widely used technique for dependencies, in recent years it has been less common for constituency parsing. There are still models built on transitions which are quite successful, such as Yang and Deng (2020), which uses an attention mechanism to predict how and where to attach the next word of a sentence. However, most state of the art models use a derivative of the CKY algorithm (Younger, 1967). For example, the current state of the art uses bidirectional attention as part of the encoder for a chart parser (Kim et al., 2023).

In this work, we revisit older transition schemes and show that they can be used to build a state-of-the-art model. We improve on existing dynamic oracle methods (Fernández-González and Gómez-Rodríguez, 2018), allowing the model to better learn the state space after an error. Furthermore, we explore self-training using a method inspired by active learning (Swayamdipta et al., 2020; Karamcheti et al., 2021), and present a simple way to ensemble transition models with the same structure. We present state of the art results on 6 languages. Using Electra embeddings with the Penn Treebank, our parser achieves a new SOTA F1 of 96.61.

All models and code described are publicly released in the (anonymous) software package.

2 Experimental Setup

We experiment on Penn Treebank 3 for English (Marcus et al., 1999) and the Chinese Treebank 5.1 for Simplified Chinese (Xue et al., 2005). We also report results for German, Indonesian, Italian, Japanese, and Vietnamese (Brants et al., 2001; Suan Lim et al., 2023; Delmonte et al., 2007; Thu et al., 2016; Ha et al., 2022). See table 1 for details.

See Appendix B for the hyperparameters used when training the model, and Appendix C for a more complete description of the datasets used.

Scores are reported as averages of 5 models.

Lang	Dataset	|Train|	|Dev|	|Test|
EN	PTB 3	39832	1700	2416
ZH	CTB 5.1	45893	2040	2725
DE	Tiger	40472	5000	5000
ID	ICON	8000	1000	1000
IT	VIT	7875	683	1042
JA	ALT	17195	934	931
VI	VLSP 22	8160	n/a	1204

Table 1: Datasets used in this work

Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), pages 26–39
August 26, 2025

3 Model Improvements

Improvements to Base Model. The parser builds from a base of the LSTM in-order transition-based parser of Liu and Zhang (2017), which itself builds from Dyer et al. (2015). Several specific improvements to this model improved scores slightly.

The first is that the original LSTM between the word embeddings and the classification layers was a unidirectional LSTM but using a bi-LSTM instead slightly improves performance. As we use self-training to boost performance, rather than reranking, we do not need an autoregressive parser.

Another minor improvement is to follow Bauer et al. (2023) in building the encoding of a new constituent by using max over the embeddings of the children. This is simpler and more effective than the Bi-LSTM labeled with the subtree used in Dyer et al. (2016) and Liu and Zhang (2017).

Pretrained Embeddings. In (Liu and Zhang, 2017), the standard at the time was to use pretrained word embeddings such as those from word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), or fasttext (Bojanowski et al., 2017). Later work incorporated character models (Akbik et al., 2018). The well-known Berkeley Neural Parser uses XLNet (Yang et al., 2019). We find that for English, Electra-Large (Clark et al., 2020) works best of the currently available pretrained word embeddings. Following best practices, we finetuned the embeddings and transformers for the parsing task. For release as FOSS, we found that PEFT, from Mangrulkar et al. (2022), reduces download size for a single model and VRAM when used in an ensemble without sacrificing model quality.

Attention Layer. Multiple past works on CKY constituency parsing add a specialized attention layer after the transformer, such as Partitioned Attention (Kitaev and Klein, 2018), Label Attention (Mrini et al., 2020), and Bi-Directional Attention (Kim et al., 2023).

Here, we combine ideas from Bi-Directional Attention with Streaming LLM (Xiao et al., 2024). Bi-Directional attention computes attention layers twice, alternatively masking forwards and backwards directions. Streaming LLM uses a windowed attention layer with attention sinks to improve NLG. We find that a single layer of windowed attention in both the forwards and backwards directions improves scores on lower resource constituency datasets, with insignificant effects on larger datasets (table 2).

Lang	No Attn	Attn
EN	96.40	96.37
ZH	95.16	95.07
DE	95.68	95.77
ID	89.22	89.28
IT	83.68	83.87
JA	93.06	93.28
VI	82.80	83.30

Table 2: Dev set scores w/ and w/o attn layer

Lang	No Attn	Attn
EN	95.46	95.51
ZH	94.19	94.49
DE	94.09	94.27
ID	88.76	89.03
IT	83.41	83.59
JA	92.06	92.31
VI	82.33	82.86

Table 3: Dev scores w/ and w/o attn, 5000 sentences

To explore how the resource size affects these scores, we reran the experiment with a randomly selected 5000 sentences for each language (table 3). In this case, the attention layer has a much more pronounced effect. We hypothesize that more data enables the LSTM layer of the encoder to capture the same information extracted by the attention layer in lower data settings.

Ensemble. We investigated the gains that are possible by ensembling multiple parsing models (at the cost of more computation). For chart parsing, a common technique is to use a high-accuracy language model to rerank many outputs of a single parser model, as in (Choe and Charniak, 2016). For transition-based parsing, there is a simpler mechanism for combining the results of multiple models. Instead of choosing a transition by taking the maximum scoring legal transition from one model, we first add the logits from several models, then use the transition with the maximum sum.

It is necessary to build models with different results in order to successfully ensemble them. We explore three methods of achieving the necessary differentiation: (i) initializing each model's layers with a different random seed (*varied seed*), (ii) using different numbers of layers from the transformer for the input embedding (*mixed layers*), or (iii) randomly reweighting the training trees while keeping the structure the same, ensuring training proceeds differently (*random reweighting*). Each method produced useful gains versus a single model, without much or consistent differentiation between them; see table 4.

Model	ID Icon	DE Tiger
single model	89.12	95.76
mixed layers ensemble	89.86	95.93
varied seed ensemble	89.74	95.95
random reweighting ensemble	89.82	95.89

Table 4: When combining models to make an ensemble, different methods for building base models in the ensemble have roughly equivalent dev scores

4 Training Improvements

Oracle. The standard oracle training for transition parsers is teacher forcing: backpropagate the errors from the prediction of a transition, apply the correct transition, and repeat. The limitation of this approach is, at runtime, the model will naturally make errors, resulting in a state which does not directly correspond to any training state.

An improvement is to update the remaining transitions to build the best result still possible after an error. This technique is a *dynamic oracle*, introduced in Goldberg and Nivre (2012) for dependency parsers. This technique works for constituency parsing as well, as shown in the bottom-up parser of Fernández-González and Gómez-Rodríguez (2019) or the discontinuous parser of Coavoux and Cohen (2019).

Fernández-González and Gómez-Rodríguez (2018) further explored this concept for top-down and in-order parsers. After each training error, the oracle used the first possible transition which minimized subsequent errors in the tree. However, this is a potentially ambiguous solution, as some transition sequences have multiple fixes which result in the same number of errors.

An example of an ambiguous transition is the subtree in figure 1 from Marcus et al. (1993). In both the in-order and top-down schemes, the correct transition is to close the ADJP after the word "rolled". If the Close is missed, the dynamic oracle can correct the sequence equally well by closing after "sheet" or after "steel".

We ran an experiment using multiple different types of ambiguous top-down dynamic oracle repairs (see Appendix E). When run over multiple languages, we found that not using a dynamic oracle when the optimal repair is ambiguous works better than choosing either randomly or deterministically; see table 5. Attempting to continue using the dynamic oracle by using the first valid repair, the last valid repair, or using the model itself to predict the repair were each less effective than using teacher forcing specifically for the ambiguous errors.

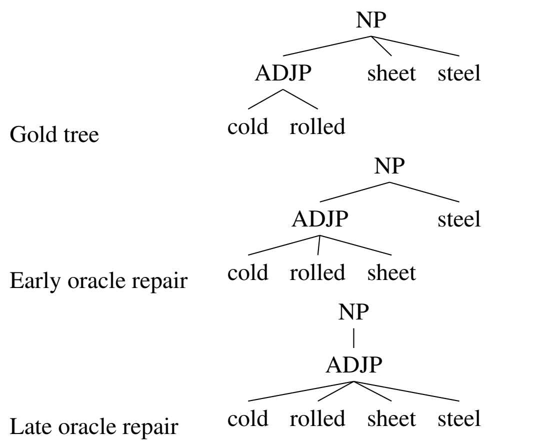


Figure 1: Example of an ambiguous repair

Lang	no oracle	w/o amb	w/ amb
EN (no trans)	92.24	92.68	92.68
ZH	90.96	91.41	91.37
DE	95.76	95.81	95.79
ID	88.75	89.13	88.89
IT	83.62	84.00	84.04
VI	82.60	82.92	82.90
avg Δ	−0.34		−0.05

Table 5: Dev set scores for the top-down model show that leaving out ambiguous repairs is slightly helpful

Self-Training. We generalized the methods of McClosky et al. (2006) and Choe and Charniak (2016) to build a corpus of silver trees for self-training for each of the languages studied. Those works used multiple types of parsers to find higher quality parses of a large corpus of newswire. In this work, we use active learning to select higher quality silver trees.

The inspiration for our method comes from Swayamdipta et al. (2020) and Karamcheti et al. (2021), who explore the idea that a high quality dataset has two general traits, high accuracy and high difficulty.

To produce silver parses for a given language, we use two ensembles (see 3), one of top-down parsers and the other of in-order parsers, and parse Wikipedia for that language. By only keeping the trees where the two ensembles agree, although not guaranteed to filter only correct parses, we build a dataset of silver trees with higher accuracy than either ensemble can produce by itself.

Such a silver dataset does not use the idea of difficulty from the active learning work, though. We can extend this by noting that the individual parsers in those ensembles do not always agree, with less agreement occurring on "harder" parses. For each silver parse, we tally how many of those individual parsers return that exact parse. Using only the parses with the *least* agreement in an ensemble produces a silver dataset which is both accurate and difficult, improving the overall results. As shown in figure 2, using trees with full agreement produced little effect, and the dataset with zero or one agreeing parsers were too small to use by themselves.[1][2] Table 6 shows the fraction of trees built from Wikipedia which had all 10 models agree, along with the improvement after discarding the full agreement trees.

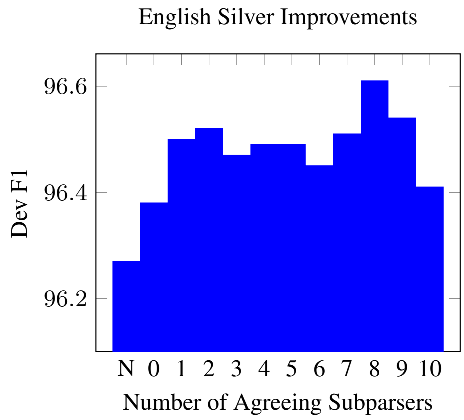


Figure 2: All silver datasets showed an improvement compared to the baseline N, but the smallest one was too small and the full agreement portion was too easy

Lang	Agreement	W/O	With
EN	0.688	96.40	96.61
ZH	0.208	93.25	93.50
DE	0.626	93.06	93.34
ID	0.419	88.68	89.01
IT	0.305	84.18	84.80
JA	0.257	93.28	93.43
VI	0.325	78.91	79.66

Table 6: In-order test scores improve for each dataset with added silver data

5 Results

In table 7, we test these techniques on 7 languages to explore a mixture of word orders and dataset sizes. For English, we investigate multiple transformers to make a fair comparison to past work, and find that Electra gives the best result for this model. We find that ensembling gives an average gain of 0.56 F1, and self training gives an average gain of 0.37 F1. Using an approximate randomization test[3], each of leaving out ambiguous oracle repairs and the windowed attention layer give statistically significant gains with $p < 0.05$.

	en_ptb3	
Kim et al. (2023)	Bert	95.86
Yang and Tu (2022)	Bert	96.01
Ours	Bert	95.95
Ours - ensemble	Bert	96.13
Kim et al. (2023)	XLNet	96.47
Ours	Electra	96.40
Ours - ensemble	Electra	96.61
Ours - self	Electra	96.61
Ours - self ensemble	Electra	96.70
	zh_ctb51	
Kim et al. (2023)	Bert	94.15
Ours	Electra	93.25
Ours - ensemble		93.71
Ours - self		93.50
Ours - self ensemble		93.66
	de_tiger	
Kitaev et al. (2019)	XLM-R	92.10
Ours	Electra	93.06
Ours - ensemble		93.47
Ours - self		93.34
Ours - self ensemble		93.66
	id_icon	
Suan Lim et al. (2023)	IndoSpanBERT	88.85
Ours	Indobert	88.68
Ours - ensemble		88.99
Ours - self		89.01
Ours - self ensemble		89.45
	it_vit	
Ours	Electra	84.18
Ours - ensemble		84.97
Ours - self		84.80
Ours - self ensemble		85.35
	ja_alt	
Ours	Roberta	93.28
Ours - ensemble		94.06
Ours - self		93.43
Ours - self ensemble		94.12
	vi_vlsp22	
Bauer et al. (2023)	PhoBert	78.73
Ours	PhoBert	78.91
Ours - ensemble		79.90
Ours - self		79.66
Ours - self ensemble		80.18

Table 7: Bracket F1 for the in-order model compared to previous results. Individual scores are the average of 5 models, and ensembles are those 5 models ensembled together. EN with Bert is included to provide a fair comparison to previous work with bert-large. ZH does not show an improvement compared to previous work. DE, ID, and VI all demonstrate large improvements. For notes on the datasets and the transformers used, see Appendix C

[1]It is possible for zero parsers to agree if each sub-model disagrees with the other nine on different transitions.

[2]This experiment in other languages showed the same effect.

[3]https://github.com/Sleemanmunk/approximate-randomization

This work sets a new SOTA score for a single model parser for PTB when using self-training, while performing close to SOTA for CTB. It also outperforms previously established best results for German, Indonesian, and Vietnamese.

6 Conclusion

In this paper, we have shown the continuing viability of neural transition-based constituency parsing, once the basic technique is carefully combined with state-of-the-art neural text encoders, other successful parser techniques, and active learning.

Limitations

Our parser only supports continuous (projective, tree-structured) constituency parses. There are transition schemes which address discontinuous trees. For example, Coavoux and Cohen (2019) wrote a parser in 2019 which achieved (at the time) SOTA results on discontinuous English and German treebanks. This limitation affects the German results in particular, where we use the SPMRL continuous version of discontinuous constituencies rather than the original Tiger trees.

The running time advantage of a transition parser is that it theoretically runs in $O(N)$ time, as opposed to a chart parser which needs $O(N^3)$ time. Once a transformer with full sentence context is used as the input embedding, the time complexity of the model becomes $O(N^2)$ at best. However, in practice, this model is actually slower than a modern chart parser. There is a large constant factor cost in the usage of Python control code to determine which operations to perform next. Whether this is because of a limitation inherent to Pytorch, an unavoidable limitation of determining at runtime which operations to perform rather than batching every operation at once, or a deficiency in the implementation is an open question.

We compare empirical results to several existing models, showing competitive or SOTA results for multiple languages. However, there are several continuous datasets not tested here, including but not limited to other treebanks in SPMRL, multiple Spanish resources from LDC, and the Icelandic Parsed Historic Corpus (Rögnvaldsson et al., 2012). Expanding the language suite used for testing would provide further evidence of the usefulness of the techniques described here and would improve the usability of this parser. Models for additional tasks will be provided upon request.

Ethics Statement

The model described above has all of the ethical limitations of the underlying datasets. It will parse any amount of offensive, misleading, or otherwise inappropriate text without any hesitation.

Constituency parsing is a highly specialized field, and as such, lower resourced languages are less studied. We deliberately chose two examples of lower resourced languages (Indonesian and Vietnamese) to highlight recent work by those communities.

While training a single parser is not expensive, the experiments described in this paper involved training many parsers to verify average results instead of using a single result, costing many days of GPU time. The data center used for the training is powered by renewable energy, mitigating the environmental impact of this work.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback. Hung Bui and Vy Thai contributed valuable experiments to past versions of this work (Bauer et al., 2023). We also thank Sidd Karamcheti and Drew Hudson for valuable conversations about the parser in its early stages. Finally, we thank the licensees of CoreNLP for their continued support of StanfordNLP, which provided funding for this work.

References

Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. 2022. Towards a Cleaner Document-Oriented Multilingual Crawled Corpus. *arXiv e-prints*, arXiv:2201.06642.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Linda Alfieri and Fabio Tamburini. 2016. (almost) automatic conversion of the venice italian treebank into the merged italian dependency treebank format. In *Proceedings of the Third Italian Conference on Computational Linguistics CLiC-it 2016*, pages 19–23.

John Bauer, Hung Bui, Vy Thai, and Christopher Manning. 2023. In-order transition-based parsing for vietnamese. *Journal of Computer Science and Cybernetics*, 39(3):207–221.

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.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2001. The tiger treebank. In *Third Workshop on Linguistically Interpreted Corpora LINC-2001*, Leuven, Belgium.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *ICLR*. arXiv.

Maximin Coavoux and Shay B. Cohen. 2019. Discontinuous constituency parsing with a stack-free transition system and a dynamic oracle. 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 204–217, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 657–668, Online. Association for Computational Linguistics.

Rodolfo Delmonte, Antonella Bristot, and Sara Tonelli. 2007. VIT – Venice Italian Treebank: Syntactic and quantitative features. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistic Theories*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018. Dynamic oracles for top-down and in-order shift-reduce constituent parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1303–1313, Brussels, Belgium. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Faster shift-reduce constituent parsing with a non-binary, bottom-up strategy. *Artificial Intelligence*, 275:559–574.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task -

automatically annotated raw texts and word embeddings. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING*.

My Linh Ha, Thi Minh Huyen Nguyen, The Quyen Ngo, Tuan Thanh Le, Tran Thai Dang, Viet Hoang Ngo, Xuan Dung Doan, Thi Luong Nguyen, Van Cuong Le, Thi Hue Phan, and Xuan Luong Vu. 2022. VLSP 2022 Challenge: Vietnamese Constituency Parsing. In *Journal of Computer Science and Cybernetics*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher Manning. 2021. Mind your outliers! Investigating the negative impact of outliers on active learning for visual question answering. 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 7265–7281, Online. Association for Computational Linguistics.

Soohyeong Kim, Whanhee Cho, Minji Kim, and Yong Choi. 2023. Bidirectional masked self-attention and n-gram span attention for constituency parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 326–338, Singapore. Association for Computational Linguistics.

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

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

Fajri Koto, Afshin Rahimi, Jey Han Lau, and Timothy Baldwin. 2020. IndoLEM and IndoBERT: A benchmark dataset and pre-trained language model for Indonesian NLP. In *Proceedings of the 28th COLING*.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

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

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3 ldc99t42. Philadelphia: Linguistic Data Consortium.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 337–344, Sydney, Australia. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics.

Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. PhoBERT: Pre-trained language models for Vietnamese. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1037–1042.

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

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

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.

Eiríkur Rögnvaldsson, Anton Karl Ingason, Einar Freyr Sigurðsson, and Joel Wallenberg. 2012. The Icelandic parsed historical corpus (IcePaHC). In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1977–1984, Istanbul, Turkey. European Language Resources Association (ELRA).

Kei Sawada, Tianyu Zhao, Makoto Shing, Kentaro Mitsui, Akio Kaga, Yukiya Hono, Toshiaki Wakatsuki, and Koh Mitsuda. 2024. Release of pre-trained models for the Japanese language. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13898–13905. `https://arxiv.org/abs/2404.01657`.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, and 4 others. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA. Association for Computational Linguistics.

Ee Suan Lim, Wei Qi Leong, Ngan Thanh Nguyen, Dea Adhista, Wei Ming Kng, William Chandra Tjh, and Ayu Purwarianti. 2023. ICON: Building a large-scale benchmark constituency treebank for the Indonesian language. In *Proceedings of the 21st International Workshop on Treebanks and Linguistic Theories (TLT, GURT/SyntaxFest 2023)*, pages 37–53, Washington, D.C. Association for Computational Linguistics.

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. *Preprint*, arXiv:2009.10795.

Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. 2012. Stylebook for the Tübingen treebank of written german (TüBa-D/Z). Technical report, University of Tübingen, Seminar für Sprachwissenschaft.

Ye Kyaw Thu, Win Pa Pa, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Introducing the Asian language treebank (ALT). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1574–1578, Portorož, Slovenia. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Preprint*, arXiv:1706.03762.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks. In *Neural Information Processing Systems (NeurIPS)*.

Songlin Yang and Kewei Tu. 2022. Bottom-up constituency parsing and nested named entity recognition with pointer networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2403–2416, Dublin, Ireland. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: generalized autoregressive pretraining for language understanding. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA. Curran Associates Inc.

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *Preprint*, arXiv:1212.5701.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

A Transition Parsing and Related Work

The underlying mechanism of transition-based parsing is similar to that of a shift/reduce compiler.

To implement a shift/reduce compiler without relying on the call stack, the compiler maintains a state with two data structures: a stack of components it has already built, which will have zero or more pieces on it, and a queue of tokens remaining to be processed. The operations allowed are to shift a new item from the queue onto the stack and to reduce some number of items from the stack into a larger, combined item.

This same general idea applies to transition parsing for NLP models. However, rather than deterministic rules to choose the next action, the parser models the current state to choose the next action. The training mechanism is to turn the gold parse tree into a gold sequence of actions, then train the model to correctly predict those actions.

The first transition models to use this mechanism focused on dependencies, becoming a well-studied method for dependency parsing (Nivre, 2003).

In one early work using this type of model for constituencies, the transitions chosen either combine the most recent pair of subtrees into a larger subtree, or create a subtree of one word out of the next node. Described as a 'bottom up' model, this constructs a binarized parse tree out of the entire sentence. Adding labels to the reduce actions makes a labeled tree, and further adding 'complete' or 'incomplete' state to the reduce actions allows the model to construct a tree with arbitrary branching, not just binarized (Clevert et al., 2016).

Building trees in a top-down manner allows for a more comprehensive understanding of the sentence or local phrase when making a determination of the next action. Adding small recurrent networks, in particular an LSTM over the subtrees, facilitated recusively constructing embeddings for phrases as well as words. Adding these mechanisms was, at the time, very close to state of the art (Dyer et al., 2016).

The in-order transition sequence improves on this mechanism by delaying when the labeling of the subtree is chosen. First, the left side of the subtree is built, and only after that is built does the model label the subtree it is currently building (Liu and Zhang, 2017). Predicting the label of the constituent with the additional information of the first child further improves accuracy.

Aside from these models, there are even more complicated options available. For example, some formalisms allow for discontinuous parse trees to represent long distance dependencies, and specialized action sets exist to account for those (Coavoux and Cohen, 2019). Furthermore, recent work explored using attention to attach the currently built subtree at any depth of the previously built subtrees, not just the top, a mechanism titled attach-juxtapose (Yang and Deng, 2020).

In this work, we continue with the top-down and in-order sequences.

B Hyperparameters and Training Setup

Encoding	
Embedding Dim	100
Forward & Backward CharLM	1024 + 1024
Electra	768
POS Tag Embedding	20
LSTM Layers	2
LSTM Input Dropout	0.2
Prediction Head	
MLP Layers	3
Nonlinearity	ReLU
Optimization	
Batch Size	50 trees
Eval Frequency	5000 trees
AdaDelta optimization	200 evals
AdaDelta LR	1.0
AdaDelta WD	0.02
AdamW optimization	200 evals
AdamW LR	0.0002
AdamW WD	0.05
Plateau LR Decay	0.6
Plateau Patience	5
Plateau Cooldown	10

Table 8: Hyperparameters used when training the model. The publicly released software includes flags for each of these settings.

After 200 training and evaluation cycles with AdaDelta (Zeiler, 2012), the model with the best dev set evaluation is trained for another 200 cycles with AdamW (Loshchilov and Hutter, 2019). The model with the best dev set evaluation among all of the models trained is kept as the final model.

The final model size is dependent on whether the model uses a transformer and whether that transformer is fully finetuned or finetuned with LoRA (Hu et al., 2021). A fully finetuned English model using ELECTRA has 59,645,169 parameters, of which 25,165,824 were finetuned Electra parameters.

This work used torch version 2.3.0 and peft version 0.10.0 (Paszke et al., 2019; Mangrulkar et al., 2022). It is likely earlier or later versions of both packages would produce similar results.

Fully training a parser with these hyperparameters takes 1–2 days on a consumer GPU, such as an Nvidia RTX 3090, depending on the dataset used. German, for example, trains faster than Chinese, as the German treebank has a much shorter average sentence length.

When building a silver dataset, we randomly chose 200,000 trees from the set of trees which did not have full agreement between the 10 submodels.

Aside from the silver dataset improvement graphs, results reported in this paper use an average of 5 models with different random seeds to report single model results. Ensemble results are based on a single ensemble built from those 5 models.

C Datasets

C.1 English

For English, we evaluate on the standard Penn Treebank (Marcus et al., 1993) We use the standard Evalb evaluation,[4] with PRT/ADVP collapsing and punctutation removal, to measure the performance of the parser. In particular, we use the "nk.prm" settings from (Kitaev and Klein, 2018) with the standard EvalB metric. For the input embedding, we use Electra-Large (Clark et al., 2020), along with a charlm (Akbik et al., 2019) and the word vectors from the CoNLL 2017 shared task (Ginter et al., 2017).

For some of the experiments, we built the model without the pretrained transformer or charlm in order to better emphasize the difference in model quality the proposed changes made. For example, the silver experiment in section 4 has results with and without the more powerful representations.

Previous work has used Devlin et al. (2018) and Yang et al. (2019) as the embedding for SOTA results. We compare with Bert for a fair comparison, while finding that XLNet and other autoregressive models are less compatible with transition constituency parsing, perhaps because the bidirectional encoders are necessary to have proper knowledge of future words.

C.2 Chinese

The Chinese Treebank version 5.1 is another standard measurement for the quality of a model (Xue et al., 2005).

One caveat with CTB is there are two standard test splits of CTB 5.1 in the literature. One is a split which includes new trees from CTB 5.1, used as recently as (Zhou and Zhao, 2019), and the other is to inherit the smaller test section from previous versions of CTB, such as used in (Kim et al., 2023). Experimentally, the smaller test set is entirely newswire and models achieve higher scores on the more structured language. Unfortunately, this dichotomy has previously gone unreported, and CTB constituency leaderboards tend to mix scores from the two test sets.[5]

In this work, we use the split of 301–325 for dev, 271–300 for test, and ignore 400–439 to provide a fair comparison with the most recent SOTA, Kim et al. (2023). The chinese-electra-180g-large-discriminator transformer (Cui et al., 2020) produced the best results for us, as opposed to Kim et al. (2023), which used Chinese BERT.

C.3 German

Three commonly used treebanks exist for German: Negra, Tiger, and Tübingen (Brants et al., 2001; Telljohann et al., 2012). Licensing reasons prevented us from using Tübingen, as we intend to publicly release our models, and Negra was included in the Tiger treebank. Accordingly, we used Tiger as the best option available to us.

For this paper, we use the SPMRL version of the dataset and evaluate it with the "spmrl.prm" settings for EvalB (Seddah et al., 2013). In order to compare the accuracy of the parser without concern for the tokenization, we use the gold tokenization provided in the SPMRL task. Furthermore, while the original Tiger treebank uses discontinuous trees, this parser only handles continuous trees. The SPMRL version of the treebank allows for such processing.

We use the Electra model from german-nlp-group[6] to build the final version of the parser. The previous SOTA on this dataset, the Kitaev et al. (2019) parser, used XLM-R (Conneau et al., 2020).

SPMRL includes several other treebanks. Models for those tasks will be made available for com-

[4] https://nlp.cs.nyu.edu/evalb/

[5] https://chinesenlp.xyz/docs/constituency_parsing.html

[6] https://huggingface.co/german-nlp-group/electra-base-german-uncased

parison or annotation purposes on request.

C.4 Indonesian

At GURT 2023, Suan Lim et al introduced a newly written Indonesian constituency treebank (Suan Lim et al., 2023). They reported a score of 88.85 using Benepar and a custom fine tuned transformer, using the standard nk.prm parameters from evalb. We built a character model from Wikipedia and the Oscar Common Crawl (Abadji et al., 2022), then tested on the various publicly available Indonesian transformers on HuggingFace.

Our best models use the Indolem Indobert model (Koto et al., 2020); other Indonesian or multilingual transformers were less accurate in our experiments.

C.5 Italian

We use the Venice Treebank (Delmonte et al., 2007) to build an Italian model. As the original treebank does not have defined train/dev/test splits, we align the sentences with the edited sentences of the UD conversion of VIT (Alfieri and Tamburini, 2016), which does have train/dev/test splits. Where no alignment is possible, such as for sentences which are split in the UD dataset, we drop the sentence. This leaves 7875 train, 683 dev, and 1042 test trees. Code to reproduce this split is included in the software release.

Using the Electra model from DBMDZ[7] produced the most accurate model for this task.

We evaluated this model using the standard evalb evaluation, adding $punto$ to the list of ignored constituencies as that represents punctuation in this treebank.

C.6 Japanese

We use the Japanese ALT (Thu et al., 2016) to build a Japanese model. This is a parallel treebank, intended to eventually have many languages parsed, but currently only Japanese is finished enough to use for constituency parsing. The treebank advertises 20,106 trees, but some number are missing from the Japanese portion of the corpus. We further eliminate 9 trees for having entire words composed of nothing but spaces. This leaves 17195 train, 934 dev, and 931 test trees.

After some brief exploration, we found that the Rinna Roberta model (Sawada et al., 2024) was a good combination of ease of use and performance.

[7] https://huggingface.co/dbmdz/electra-base-italian-xxl-cased-discriminator

C.7 Vietnamese

In 2022, VLSP produced a constituency parsing dataset, along with a bakeoff (Ha et al., 2022). We compare our results against the best performing model from the bakeoff, from (Bauer et al., 2023). Note that the publicly reported contest scores include both POS and bracket scores, whereas this score is reported on only brackets, leading to our scores differing from the publicly reported scores. We use evalb to evaluate, adding $punct$ to nk.prm.

As there is no defined dev set for VLSP22, we use a random sample of 1/10th of the training dataset.

We found the best transformer for building the parser was Phobert-Large (Nguyen and Nguyen, 2020).

C.8 Availability

The previous artifacts are available for research purposes. PTB and CTB are both available from LDC, whereas VIT is available at ELRA. SPMRL and Vietnamese were provided by request. ICON and ALT are freely available as part of the published work. Whenever possible, we confirmed with the authors that models derived from the work can be publicly released. This limitation informed our choice of German treebank, in particular.

The models derived from these datasets will be available at (anonymous) under the Apache License, Version 2.0.

D Example Oracle Sequence

Both the in-order and top-down transition scheme are capable of constructing any tree, doing so with an unambiguous transition sequence. Included in Table 9 is an end to end parsing example for "Transition parsing is fun" using a top-down transition sequence.

E Ambiguous Dynamic Oracle

An unambiguous dynamic oracle repair is when the dynamic oracle has only one minimum error option for how to rewrite the transition sequence after the parser makes an error at training time. An example of this is when the top-down gold sequence is to Shift, but the model chooses to Close. The incorrectly closed bracket is both a recall and a precision error, but causes no further errors provided the remainder of the sequence is properly followed, so the best, unambiguous repair is to remove the correct Close from later in the sequence.

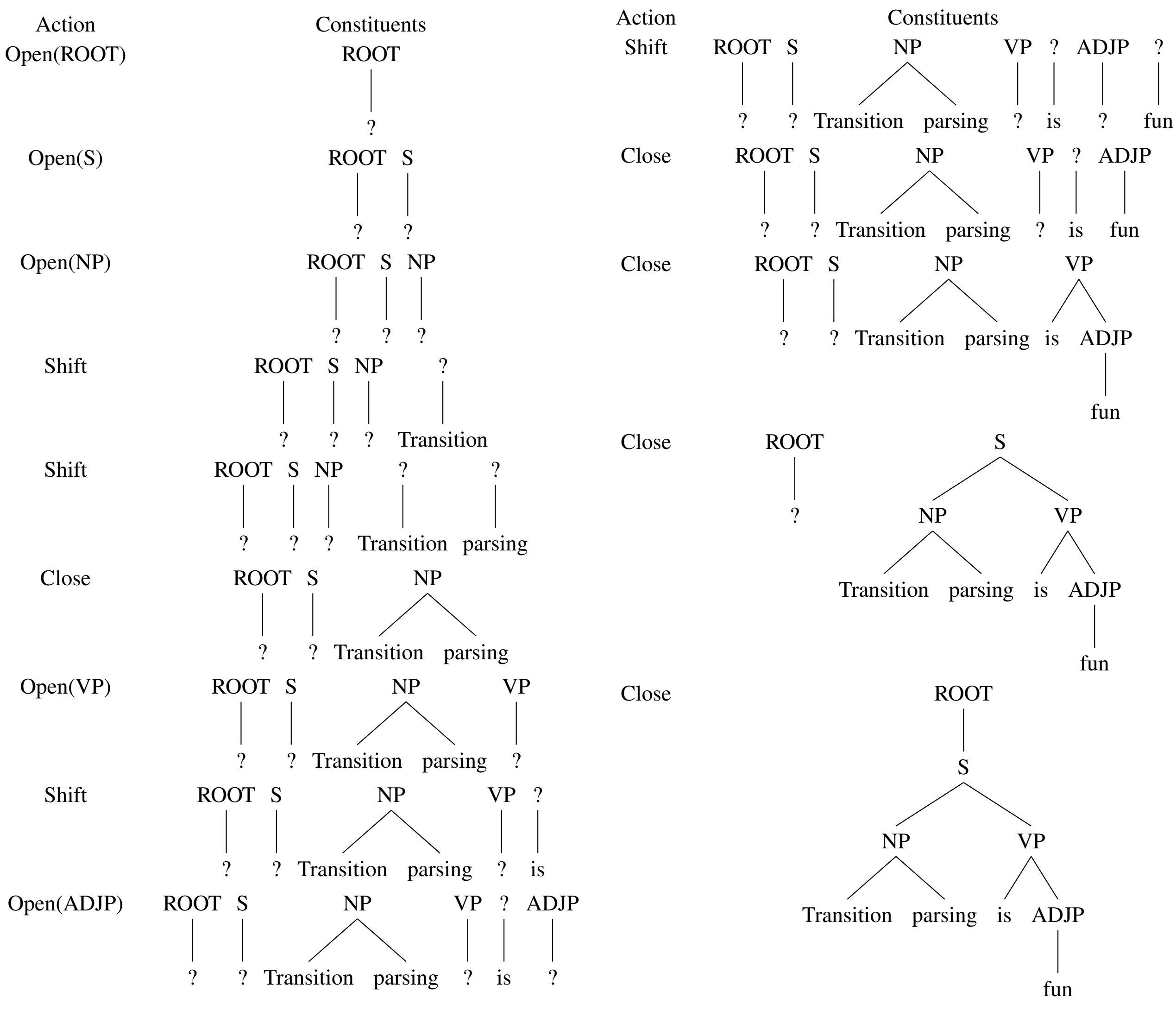
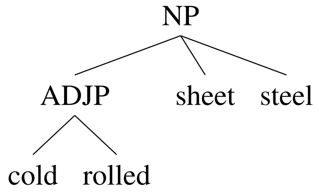
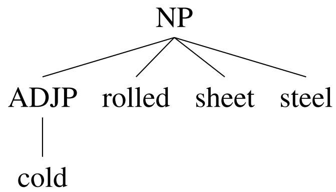


Table 9: Complete top-down transition sequence for “Transition parsing is fun”



Correct



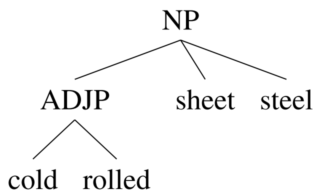
After an incorrect Close, this is the best tree the model can construct

An ambiguous error has multiple possible resolutions which produce the same number of errors.

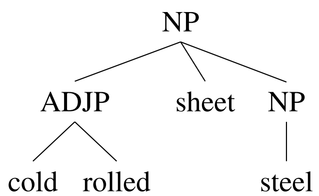
The most common ambiguous transition error for the in-order model is that of a shift replaced with an open, empirically representing 1/6th of the errors made by a fully trained model. Such a transition introduces a single precision error, the incorrectly opened bracket. There are multiple ways of "repairing" the transition sequence after such an error. For example, an immediate close of the new bracket represents a unary transition around the previous item, whereas a close at the end of the bracket could contain several subtrees.

In this section, we enumerate the ambiguous errors possible in the top-down scheme. The in-order scheme is similar, but with more complicated exceptions.

If the correct transition is Shift, but the model predicts Open, this creates a new bracket where there is none, a single precision error. The new constituent must be closed at some point. If the Shift was the final word of the current bracket, this is not ambiguous:

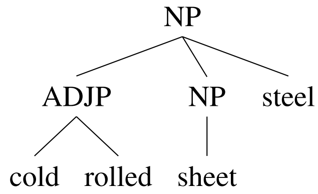


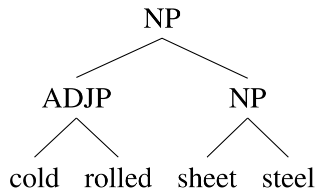
Correct



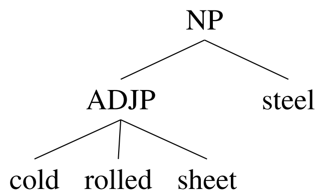
Incorrect NP with an unambiguous close

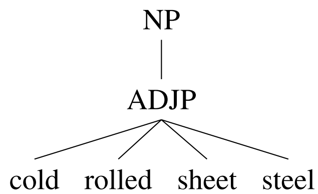
If the incorrect NP in this example opens before the word 'sheet', though, the new constituent can close either after 'sheet' or 'steel':





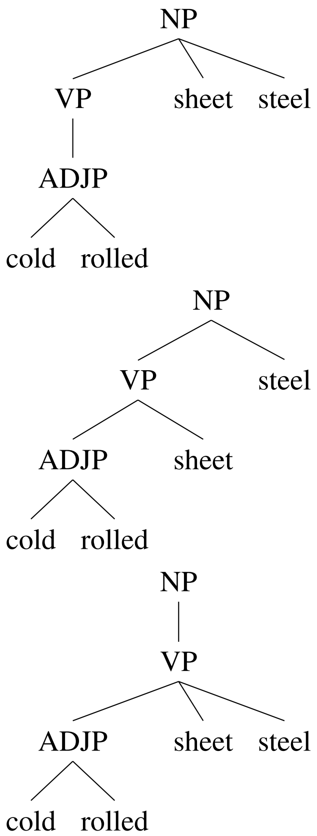
If the correct transition was to Close, but the model predicts Shift, this causes both a precision error and a recall error. The newly created bracket is a precision error, and the correct bracket cannot be recovered, a recall error. The dynamic oracle repair is ambiguous in the case of more than one piece after which the bracket could close. Continuing with the 'steel' example, if the ADJP is not correctly closed:





The same effect happens with a Close to Open error. Both the new constituent and the unclosed constituent from earlier need to be closed, and the time at which to do that is ambiguous unless the constituent has only one possible resolution. Note that the incorrect Open does not necessarily cause a recall error in the case of building a correct constituent at the wrong time, such as in the phrase 'eat (NP spaghetti) (PP with a fork)' becoming 'eat (NP spaghetti (PP with a fork))'

In the case of an Open constituent incorrectly labeled, in most cases this is a precision error which the dynamic oracle can repair by building the correct constituent anyway, then closing the incorrect constituent later. When to close may be ambiguous, such as if the model added an incorrect VP in the 'cold rolled sheet steel' example:



Another ambiguity in the mislabeled Open case can arise when the incorrectly predicted Open transition is the next correct transition, such as if the model had predicted ADJP instead of NP. In that case, discarding the Open(NP) and the corresponding Close would cause a single recall error instead of a precision error.

F Dynamic Oracle Repair

We ran an extensive test on the in-order transition error described in section 4, when the gold sequence has a Close-Shift, but the prediction is Shift. In this case, the correct bracket is lost, meaning a recall error, and a new, wider bracket is created, meaning a precision error. To prevent further errors, this new bracket must be closed before the bracket enclosing it is also closed. In the case of a bracket wider than one more node, this close transition could occur after any number of additional nodes without further affecting the score, meaning it is an ambiguous oracle repair.

The signal is not strong, but as shown in table 10, the overall trend for this one repair is to be slightly more accurate to not choose any ambiguous transition. Further tests using all of the dynamic oracle repairs found that not using any ambiguous repairs was an improvement.

As observed in section 4 for the top-down oracle,

Lang	Unamb	Early	Late
EN	92.53	92.59	92.52
ZH	91.55	91.45	91.38
ID	89.21	89.24	89.21
DE	95.66	95.64	95.72
IT	83.76	83.63	83.65
VI	82.85	82.76	82.78

Table 10: Dev scores for the close/shift error described above show this ambiguous error cannot be deterministically resolved in a positive manner.

when testing multiple such ambiguous repairs at once, the trend is that using teacher forcing for the ambiguous errors is a slight improvement.

Crosslingual Dependency Parsing of Hawaiian and Cook Islands Māori using Universal Dependencies

Gabriel H. Gilbert[1], Rolando Coto-Solano[2], Sally Akevai Nicholas[3], Lauren Houchens[2], Sabrina Barton[2], Trinity Pryor[2]

[1] University of Chicago, [2] Dartmouth College
[3] The University of Auckland (Waipapa Taumata Rau)
Correspondence: ghgilbert@uchicago.edu, rolando.a.coto.solano@dartmouth.edu, ake.nicholas@auckland.ac.nz

Abstract

This paper presents the first Universal Dependency (UD) treebank for ʻŌlelo Hawaiʻi (Hawaiian). We discuss some of the difficulties in describing Hawaiian grammar using UD, and train models for automatic parsing. We also combined this data with UD parses from another Eastern Polynesian language, Cook Islands Māori, to train a crosslingual Polynesian parser using UDPipe2. The crosslingual parser produced a statistically significant improvement of 2.4% in the labeled attachment score (LAS) when parsing Hawaiian, and this improvement didn't produce a negative impact in the LAS of Cook Islands Māori. We will use this parser to accelerate the linguistic documentation of Hawaiian.

Hōʻuluʻulu Manaʻo

I kēia pepa, hōʻike mākou i waihona pepeke mua loa no ka ʻŌlelo Hawaiʻi i ke ʻano Universal Dependency (UD). Wehewehe aku mākou i nā pilikia me ka hoʻohana ʻana iā UD, a waiho mai i kumu mīkini hou no ke kuhikuhi ʻano huaʻōlelo ʻana i ke ʻano hana nona iho. Hoʻohui i ia ʻike me nā palapala ʻōlelo o ke ʻano UD mai iā ʻŌlelo Kuke ʻAilani, a hoʻomaʻamaʻa i mīkini kuhikuhi ʻano huaʻōlelo me UDPipe2. He kōkua maoli nō, me ka maikaʻi aʻe o 2.4% i ka "labeled attachment score" (LAS) me ka ʻŌlelo Hawaiʻi, ʻaʻole naʻe i hoʻopilikia i ka LAS o ʻŌlelo Kuke ʻAilani. Makemake mākou e hoʻohana aku i ia no ka pono o ka hoʻōla ʻōlelo ʻia ʻana o ka ʻŌlelo Hawaiʻi.

1 Introduction

This paper presents the first attempt to construct a Universal Dependencies (Nivre et al., 2020) treebank for ʻŌlelo Hawaiʻi (hereafter: Hawaiian). Hawaiian is an Indigenous Polynesian language spoken in Hawaiʻi as a community language by Kānaka Maoli (Native Hawaiians) and non-Hawaiian residents (Kimura, 1983). Hawaiian has been the subject of intense revitalization efforts since the Hawaiian Cultural Renaissance of the 1970s (Kamanā and Wilson, 2001). Given the need for increased grammatical analysis to further revitalization goals, NLP tools like parsing can potentially facilitate the creation of annotated corpora.

Here we describe the structure of the treebank and use the treebank of a related Polynesian language (Cook Islands Māori) to build a crosslingual Polynesian UD parser.

1.1 NLP for Polynesian Languages

There are two motivations for this project. The first one is to create a parser for Hawaiian in order to conduct syntactic analysis of sentences gathered in the process of linguistic documentation and description. The second, larger goal, is to foster networks of collaboration across linguists from Polynesia, and to join efforts in accelerating documentation of their languages, with the ultimate purpose of language revitalization and normalization. In the case of this project, the Hawaiian author (Gilbert) is collaborating with the Cook Islands author (Nicholas) because Cook Islands Māori is the only other Polynesian language that has a treebank available (Karnes et al., 2023). These two languages are Eastern Polynesian and they are closely related. Their syntax shows numerous commonalities: VSO order, a verbal complex with tense-aspect-mood markers and directionals, a very similar system of articles, demonstratives and numerals, and numerous cognates with relatively transparent changes between the proto-language and the two languages (e.g. *taŋata "person" > CIM *tangata*, Haw. *kanaka*) (Elbert and Pukui, 1979; Nicholas, 2017). These two languages also share the status of being under-resourced in terms of NLP data. We hope to leverage their linguistic commonalities to improve the parsing models and to work towards the common goal of describing their syntax to facilitate language teaching and transmission.

Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), pages 40–50
August 26, 2025

There is previous work on NLP for Eastern Polynesian languages, including work on automatic speech recognition (ASR) for te Reo Māori from Aotearoa New Zealand (Jones et al., 2023) and Cook Islands Māori (Coto-Solano et al., 2022a), development of text-to-speech for both languages (Keith, 2024; James et al., 2024), part-of-speech tagging for both languages (Finn et al., 2022; Coto-Solano et al., 2018), and forced alignment for Cook Islands Māori (Nicholas and Coto-Solano, 2019; Coto-Solano et al., 2022b).

NLP work on Hawaiian has involved forays into speech recognition (Chaparala et al., 2024), morphological analysis (Hosoda, 2019) and orthographic reconstruction (Shillingford and Parker Jones, 2018). Additionally, some large language models like ChatGPT (OpenAI, 2022) and Gemini (GeminiTeam et al., 2024) support the generation and translation of Hawaiian, while speech recognition models like Whisper (Radford et al., 2023) can provide support for Hawaiian ASR. We hope to expand the availability of Hawaiian NLP applications and develop tools to further empower language documentation work.

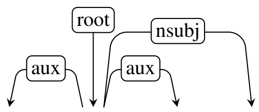
This project is similar to work on other treebanks for under-resourced languages (Rodríguez et al., 2022; Tyers and Henderson, 2021; Coto-Solano et al., 2021; Ramsurrun et al., 2024). Our main goal is to create treebanks that can help with linguistic documentation. A secondary goal is to use the data collected along the way to create temporary models that can be used for bootstrapping, so that the annotation process can switch from completely manual annotation to computer-aided annotation. By doing this, the model can provide a first-pass of the parsing and a human expert can correct this. This accelerates the process and leads to faster annotation.

2 Methodology

We will first describe the structure of the treebanks, and then describe the experiments to parse the Hawaiian and CIM data using zero-shot, monolingual and crosslingual methods. In these experiments we attempt to leverage the similarities between Hawaiian and CIM to improve the performance of the Hawaiian model: the model with the least amount of data. We also attempt to leverage a high-resource language, English, to investigate whether its models can aid in the initial parsing of these two low-resource languages.

2.1 Data Sources and Annotation

The first step was to create a dependency treebank for Hawaiian. We collected a total of 145 sentences, containing a total of 1015 tokens. Sentences were 7 ± 3 tokens on average, and came from past documentary linguistic work (12 from Elbert and Pukui 1979; 2 from Pukui and Elbert 1986; 10 from Gilbert 2023), published interviews (24 from Kanahele 1970), and from fieldwork with Hawaiian speakers (98). Sentences were manually annotated using Universal Dependencies 2 (UD2) (Nivre et al., 2020). Example (a) shows a typical parse; the Tense-Aspect-Modality (TAM) particles surrounding the verb root are tagged as auxiliaries, and the subject follows the verbal complex.



(a)	E	hiamoe	ana	lākou
	IPFV	sleep	IPFV	they
	AUX	VERB	AUX	PRON

"They were / are / will be sleeping"

We manually tagged the corpus using Universal Parts of Speech (Nivre et al., 2020). Table 1 shows the distribution of POS tags for Hawaiian. The most frequent part of speech is VERB (15%), followed by PUNCT (15%) and NOUN (12%).

VERB	157 (15%)	PRON	96 (9%)
PUNCT	147 (15%)	DET	85 (8%)
NOUN	119 (12%)	PROPN	30 (3%)
ADP	115 (11%)	ADJ	12 (1%)
AUX	112 (11%)	CCONJ	10 (1%)
ADV	107 (11%)	Others	25 (2%)

Table 1: Frequency of UPOS tags in the Hawaiian Treebank (145 sentences, 1015 tokens).

We also annotated the corpus for relations. Table 2 shows that the most common relations in the Hawaiian corpus are root (14%), punct (14%) and case (11%).

Next, we expanded the pre-existing treebank for Cook Islands Māori (CIM) (Karnes et al., 2023). We grew the previous treebank, which contained 126 sentences (1035 tokens), by adding more sentences from a grammar of the language (Nicholas, 2017) and an L2 learning manual (Turepu Carpenter and Beaumont, 1995), manually annotating them using UD2. This new corpus has 663 sentences, with a total of 7658 tokens and an average

root	145 (14%)	advmod	46 (5%)
punct	145 (14%)	compound	39 (4%)
case	115 (11%)	obj	33 (3%)
nsubj	106 (10%)	obl	21 (2%)
aux	96 (9%)	cc	20 (2%)
det	74 (7%)	Others	175 (17%)

Table 2: Frequency of relations in the Hawaiian treebank (145 sentences, 1015 tokens).

sentence length is 12±7 tokens. Table 3 shows the most common parts of speech. The three most frequent ones are NOUN (18%), ADP for adpositions (15%) and DET determiners (14%). All of these occur at higher proportions than in the Hawaiian corpus.

NOUN	1418 (18%)	ADV	513 (7%)
ADP	1122 (15%)	PUNCT	472 (6%)
DET	1094 (14%)	PROPN	239 (3%)
VERB	894 (12%)	PART	155 (2%)
AUX	861 (11%)	ADJ	140 (2%)
PRON	587 (8%)	Others	163 (2%)

Table 3: Frequency of UPOS tags in the CIM treebank (663 sentences, 7658 tokens).

The CIM dataset was also tagged for relations; the summary of these is shown on Table 4. The proportion of nsubj, aux and obj tags in CIM is similar to those in Hawaiian, but the CIM data has more instances of case (15%) and det (12%).

case	1156 (15%)	advmod	446 (6%)
det	953 (12%)	obl	409 (5%)
aux	850 (11%)	nmod	354 (5%)
nsubj	694 (9%)	obj	340 (4%)
root	663 (9%)	amod	161 (2%)
punct	471 (6%)	Others	1161 (15%)

Table 4: Frequency of relations in the CIM treebank (`nmod` includes the possessive `nmod:poss`) (663 sentences, 7658 tokens).

2.2 Zero-Shot and Monolingual Experiments

The first step in our experiment was to train monolingual parsing models for each language. The total number of sentences for each language were randomly split into 80%-10%-10% train/dev/test sets. The test sets belong to the same domain as the training sets: linguistic examples and language learning textbook examples (see section 2.1). We repeated this process 30 times, obtaining 30 unique test sets for each language. The Hawaiian sets had 115/15/15 sentences, while the CIM sets had 531/66/66 sentences. We trained 30 separate models with these sets for each language using the UDPipe2 parser (Straka, 2018). For each model, we calculated the F1 of the Universal Parts of Speech (UPOS), unlabeled attachment score (UAS), and labeled attachment score (LAS). We report the median score of these 30 measures.

Next, we used the monolingual models to test zero-shot parsing with a closely related language. We parsed the original 30 test sets for Hawaiian with the CIM monolingual model. We also parsed the original 30 test sets of CIM using the Hawaiian monolingual model. We evaluated these parses with respect to median UPOS, UAS, and LAS.

One of our ultimate goals in this project is to study the parsing of extremely low-resource Indigenous languages, for which entirely new datasets might need to be built from scratch at great expense to community members, language practitioners, and researchers. If existing models can facilitate this work, we could obtain a considerable head start in new projects. To test this, our next experiment was to parse the Hawaiian and CIM sentences using a zero-shot method, with a model from a completely unrelated language. We chose the `en_core_web_sm` English model from spaCY (Honnibal et al., 2020) because of its easy usability by other researchers. We used this model to parse the same 30 test sets for Hawaiian and 30 test sets for CIM, and report UPOS, UAS, and LAS.

2.3 Crosslingual Experiments

In the second stage of our experiments we trained models where we combined the Hawaiian and CIM data during the training stage. We trained UDPipe2 models, combining the 30 training/dev sets for both languages, and parsed 30 test sets for each language. We also report UPOS, UAS, and LAS for these models.

We then performed an additional experiment where we modified one language to resemble the other. Hawaiian and CIM are closely related, and their cognates show well-attested regular sound correspondences that go all the way back to Proto-Polynesian. Table 5 shows five sound correspondences that are stable enough that they can very transform a CIM word into a Hawaiian word. For example, the CIM word *rātou* 'they' may be changed into its Hawaiian cognate `lākou` by chang-

CIM	Hawaiian
k	‘ (‘okina)
t	k
v	w
' (saltillo)	h
ng / n	n

Table 5: Examples of regular sound alternations between CIM and Hawaiian (Otsuka, 2005).

ing the ‘r’ for an ‘l’ and the ‘t‘ for a ‘k’. These transformations, based on well-documented diachronic processes (Otsuka, 2005), were applied to the CIM data so that it would bear an even closer resemblance to the Hawaiian data. We performed these changes to the 30 train/dev sets of CIM, combined them with the original train/dev Hawaiian sets, and then tested on the Hawaiian test sets.

Finally, we replicated the modified condition, this time modifying the Hawaiian text to more closely resemble the CIM text. For example, the orthography of the Hawaiian word *kākou* ‘everyone’ was transformed into *tātou*, again using the historical sound correspondences in table 5. We applied the first four changes but were unable to do so for the fifth change (n>ng), because the ⟨n⟩ in Hawaiian can be related to either an ⟨n⟩ in CIM (e.g. Haw: *manu*, CIM: *manu* ‘bird’) or to an /ŋ/ (e.g. Haw: *mauna*, CIM: *maunga* ‘mountain’). We applied the one-to-one changes to the Hawaiian sentences and combined them with the original CIM sets. We ran the 30 training rounds and evaluated on the 30 CIM test sets.

In summary, the experiments with the parsing models have five conditions: (i) zero-shot evaluation on an English model, (ii) zero-shot evaluation on a closely-related Polynesian language, (iii) monolingual training, (iv) crosslingual training with data from both Hawaiian and CIM, and (v) crosslingual training where one of the Polynesian languages was modified to more closely resemble the other.

3 Results

Table 6 shows the medians for each language, condition, and metric. Figure 1 summarizes the results for zero-shot parsing versus parsing with a monolingual model trained specifically for each language. Figure 2 summarizes the results for the crosslingual training compared to using monolingual models.

3.1 Hawaiian Models

First, we study the relationship between the zero-shot parses by using an ANOVA model with the zero-shot and the monolingual conditions as independent variables. When zero-shot parsing is performed with a model from a closely related language, it provides significantly better results than when the model is trained on a genetically unrelated language. The zero-shot parses for Hawaiian using an English model have a median of LAS=3%; this is much lower than the parses using a CIM-only model, which have LAS=42% ($F(2,87)=1308$, $p<0.0001$; Bonferroni-corrected $p<0.0001$). These improvements also hold for the other metrics: zero-shot UAS is significantly higher for the CIM-only model than for the English model (66% versus 24%, $F(2,87)=691$, $p<0.0001$; Bonferroni-corrected $p<0.0001$), and zero-shot UPOS follows this pattern (56% versus 19%, $F(2,87)=1947$, $p<0.0001$; Bonferroni-corrected $p<0.0001$).

Using the same ANOVA models, we will study the relationship between the zero-shot parses and the parses generated with the monolingual Hawaiian model. For the three metrics (UPOS, UAS, LAS), the model trained on monolingual Hawaiian data has a significantly higher F1 than the best performing zero-shot approach. When parsing Hawaiian sentences, the monolingual Hawaiian model has a median LAS of 69.5% compared to 42% for zero-shot using a CIM model. UAS has a median of 80.5% compared to 66% for zero-shot with CIM, and UPOS has a median of 85.8% for the monolingual Hawaiian model, compared to 56% when Hawaiian is parsed with the zero-shot using CIM. All of these differences are significant (Bonferroni-corrected $p<0.0001$).

Finally, we will study the effects of building a crosslingual model by training on both Hawaiian and CIM data. A repeated measures t-test showed that training on both the Hawaiian and CIM data produced a median significant improvement of 1.94% in LAS ($t(29)=2.4$, $p<0.05$), compared to parsing with a model trained only on Hawaiian data. When the data is not paired, the difference is larger: LAS_{Cross}: 71.9%, LAS_{Mono}: 69.5%; ΔLAS=2.4%. When the model was trained on a combination of the Hawaiian and modified CIM data (see section 2.3), this produced a smaller but still significant improvement of 1.64% ($t(29)=2.3$, $p<0.05$) in LAS compared to the parses generated by the monolingual Hawaiian model. The non-

	Hawaiian			CIM		
	UPOS	UAS	LAS	UPOS	UAS	LAS
(1) Zero-Shot (English model)	19.1	24.3	2.8	16.2	20.9	1.8
(2) Zero-shot (with Polynesian model)	56.1	66.2	41.6	47.4	48.2	28.3
(3) Monolingual	85.8	80.5	69.5	**90.9**	**87.0**	**78.0**
(4) Crosslingual (Hawaiian + CIM)	85.5	81.5	**71.9**	90.7	86.9	77.8
(5) Crosslingual (with modified Polynesian lang)	**86.9**	**81.9**	71.7	90.7	86.9	77.5

Table 6: Median F1 for UD parsing. (In condition 2, the Hawaiian data was parsed using a model trained on CIM, and the CIM data was parsed using a model trained on Hawaiian. In condition 5, we modified the Cook Islands data to match Hawaiian orthography and tested on Hawaiian, and viceversa for CIM).

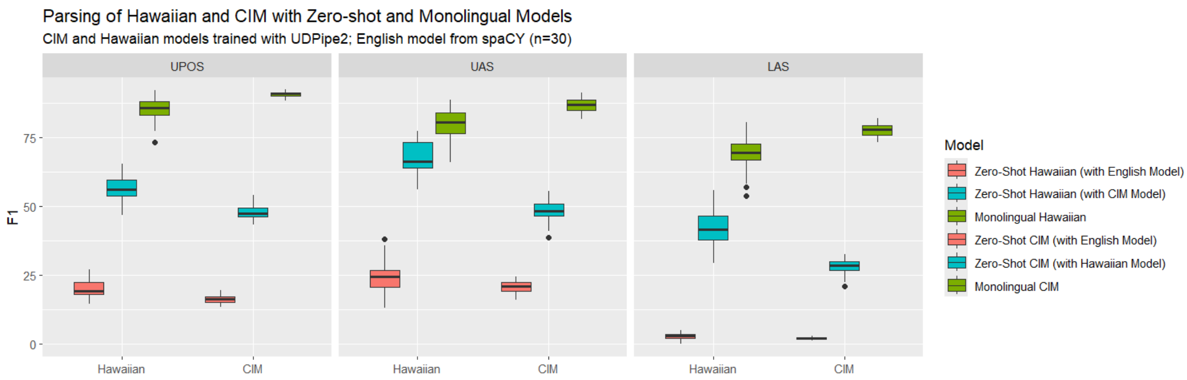


Figure 1: Zero-shot and monolingual parsing for Hawaiian and Cook Islands Māori.

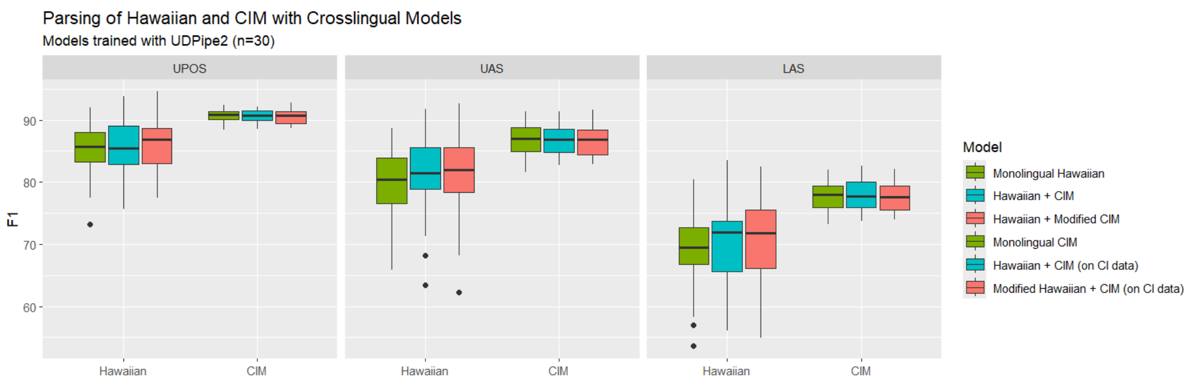


Figure 2: Monolingual and Crosslingual parsing for Hawaiian and Cook Islands Māori.

paired difference is ΔLAS=2.2%.

Both of these patterns also hold for UAS. The crosslingual model had a median paired improvement of 1.7% (t(29)=2.1, p<0.05), and the crosslingual model with the modified CIM data had a median paired improvement of 1.6% (t(29)=1.8, p<0.05). However, this pattern does not hold for UPOS. Training on both the Hawaiian and the CIM data does not provide statistical improvements for UPOS, regardless of whether the CIM data is modified to more closely resemble Hawaiian (ΔLAS_{Modif}=1.1%, p=0.10) or not (ΔLAS_{Modif}=-0.3%, p=0.28).

In summary, the crosslingual training provided a small but significant boost to the parsing of Hawaiian data. Modifying the CIM data did not provide further improvement. Zero-shot parsing is possible, but is improved by using a model from a closely-related language.

3.2 CIM Models

We also studied the performance of zero-shot parsing of CIM (using both English and Hawaiian trained models), monolingual parsing, and crosslingual parsing with the added Hawaiian data. The F1 is lowest when parsing with an English model (median LAS: 2%), compared to parsing with a model trained on the Hawaiian data from section 2.1 (median LAS: 28%, $F(2,87)=8411$, $p<0.0001$, Bonferroni corrected: $p<0.0001$). This holds true for the other metrics: UAS is 21% for the zero-shot English and 48% for zero-shot using the Hawaiian monolingual model (Bonferroni corrected: $p<0.0001$). This is also the case for UPOS: F1 is 16% for zero-shot English, and 47% for parsing with Hawaiian (Bonferroni corrected: $p<0.0001$).

When we compare the monolingual versus the crosslingual models, the patterns for CIM are different from those in Hawaiian. There is no significant difference in the parsing results when using the crosslingual model, compared to the monolingual model (LAS_{Cross}=77.8, LAS_{Mono}=78.0, paired t-test p=0.09). Likewise, there are no significant differences between the crosslingual model with modified Hawaiian and the monolingual model (LAS_{Modif}=77.5, LAS_{Mono}=78.0, paired t-test p=0.42). This is also true for the other metrics: there are no significant differences when calculating the UAS ($p_{Cross/Mono}$=0.14, $p_{Modif/Mono}$=0.58) or the UPOS ($p_{Cross/Mono}$=0.60, $p_{Modif/Mono}$=0.81).

In summary, using a crosslingual model to parse the CIM data does not significantly improve or affect the results, compared to using a monolingual CIM model. Zero-shot parsing of CIM is also better when using a model from a closely-related language (i.e. Hawaiian), but the improvement is much less (ΔLAS=26.5%) than what was found when parsing Hawaiian using the CIM model (ΔLAS=39%), probably because there was much less Hawaiian data to contribute to the learning of CIM.

4 Discussions

In this section, we discuss the performance of the monolingual and crosslingual models, the kinds of errors they make when parsing, and consider some issues encountered while constructing the Hawaiian treebank.

4.1 Crosslingual Parsing

Figure 3 shows the change in LAS between the monolingual and crosslingual (without orthographic modification) conditions. There is a significant performance gain when using the crosslingual model on Hawaiian: the median gain was 2.4% (LAS_{Mono}=69.5% versus LAS_{Cross}=71.9%). But these gains were not uniform. As is depicted in Figures 1 and 2, there is considerable variation for the crosslingual conditions. In fact, for some of the 30 test sets, we actually observed a loss in F1. Gains can be as high as 9.6% (LAS_{Mono}=69.1% versus LAS_{Cross}=78.7%), while losses may be as high as 8.6% (LAS_{Mono}=69.8% versus LAS_{Cross}=61.2%). This pattern should be kept in mind when working with such small datasets, especially where the exact sentences used in each of the train/dev/test sets might have major effects on performance down the line.

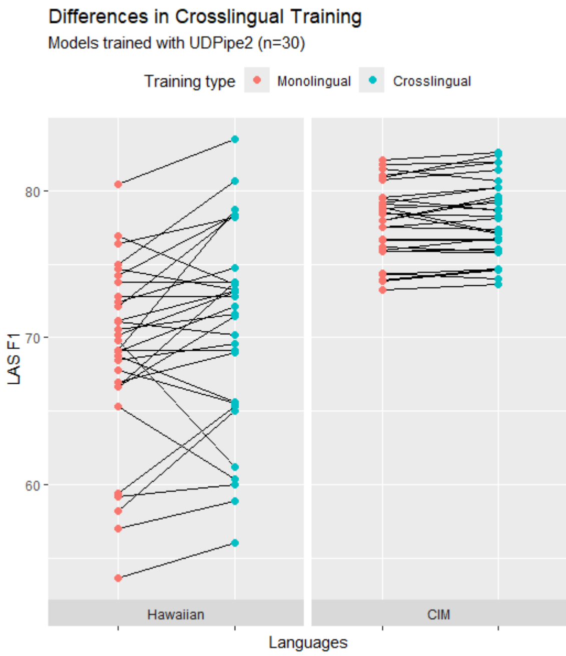


Figure 3: LAS of parses in the crosslingual and monolingual conditions, for specific test sets.

As observed in the results section, crosslingual training did not significantly impact the performance of the CIM LAS. The unpaired difference between the crosslingual and monolingual conditions is 0.2 in favor of the monolingual models, but the paired difference between them is 0.26 in favor of the crosslingual model. Figure 3 shows the values for the 30 test sets, and again we see variation: gains as high as 1.7% in some sets, losses as low as 1.7% in others.

Modifying the orthography of one language to be closer to the other did not provide gains in performance important enough to justify its usage. In the case of Hawaiian, even if the modified dataset did have the highest scores for UPOS and UAS, these were not significantly higher than those of the simple crosslingual model (p=0.36).

In general, the Hawaiian model's performance is higher than that of other similarly-sized models, e.g. Yoruba (140 sentences, UPOS 59, UAS 45, LAS 29, (Dione, 2021)). This is potentially due to Hawaiian's lower number of inflectional suffixes. As for CIM, its performance is comparable to that of models of similar size, for example Ottoman Turkish (9000 sentences, UPOS 88, UAS 62, LAS 52), Tamil (12000 sentences, UPOS 89, UAS 78, LAS 69) and Telugu (6000 sentences, UPOS 94, UAS 91, LAS 84) (Straka, 2025).

4.2 Common Parsing Problems

The total number of errors across all 30 Hawaiian test sets show fewer errors for the crosslingual model than the monolingual model, with respect to both UPOS tagging (421 vs. 414 errors) and LAS (686 vs. 672 errors). The most common problems when tagging relations were mislabeling a coordinating conjunction (cc) as an auxiliary (27 times for the monolingual model vs. 26 times for the crosslingual), labeling a adverbial modifier (advmod) as the root (21 vs. 17), and mislabeling an oblique argument as the direct object (18 vs. 14). As for the parts-of-speech, the most common errors relating to parts-of-speech involved mislabeling adverbs as verbs (34 vs. 39), adverbs as nouns (22 vs. 19), and adverbs as auxiliaries (20 vs. 10).

As for the CIM data, the most common relations errors were mislabeling oblique arguments as objects (141 errors in the 30 test sets parsed using monolingual models), objects as obliques (101 errors), and auxiliaries as case markers (95 errors). These errors possibly stem from the fact that a given word with the form *i* can be either a TAM marker, direct object marker, or locative/temporal oblique marker; the system may still be learning to correctly identify each homophone. As for parts-of-speech, the most common errors were mislabeling pronouns as determiners (123), verbs as nouns (122), and auxiliaries as adpositions (86).

4.3 Challenging Structures in Hawaiian

Here we will discuss three challenges during the construction of the Hawaiian treebank: (i) issues with orthography, (ii) morphology and tokenization, and (iii) dependent clauses.

4.3.1 Orthography

Orthographic representations of Hawaiian offer challenges to straightforward data processing. Hawaiian has a relatively small phonemic inventory: eight consonants (/p/, /m/, /w/, /n/, /l/, /k/, /ʔ/, /h/) and five vowels (/i/, /e/, /a/, /o/, /u/) with contrastive vowel length (e.g. /pipi/ 'cow' vs. /pīpī/ 'stingy'). Hawaiian's original orthography did not mark glottal stops or vowel length; a standardized orthography developed in the 1970s introduced (1) the *ʻokina* for the glottal stop /ʔ/, e.g. */paʔina/* as ⟨paʻina⟩, and (2) the *kahakō* (macron) for vowel length, e.g. /laːkou/ as ⟨lākou⟩ (Wilson, 1981).

Most Hawaiian texts predate this modern orthography, and involve both homographs (e.g. *paʻina* 'crack' and *paina* 'pine' are both written ⟨paina⟩) and true homophones (e.g. directional adverb *mai* 'towards speaker' vs. preposition *mai* 'since, from'). In this dataset, we included a small number of sentences (three) which were written in both the traditional and the modern orthography, in an attempt to familiarize the system with this variation.

Furthermore, historical and modern orthographic representations often differ in how they write high frequency collocations (e.g. either as a single or as multiple words). Some of these are given below:

Traditional	Modern	Gloss
akula	*aku lā / akula*	'away'
apau	*ā pau / āpau*	'all'
oia	*ʻo ia / ʻoia*	'3SG'

Table 7: Orthographic comparisons of collocations.

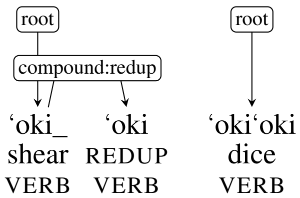
While decomposition seems possible, these collocations suggest a degree of lexicalization beyond orthographic choice. With the 3SG pronoun, for example, both morphemes may be found elsewhere—*ʻo* is a focus marker, *ia* is a demonstrative element meaning 'that (one)'—but inflection of the 3SG requires both. Several parses (three) were created for different variations of the same sentence to enrich the treebank with orthographic variation.

For Hawaiian varieties of slightly different phonological inventories whose speakers may not follow standardized spelling conventions (e.g. the Niʻihau community, see NeSmith (2019)), we chose to keep their words as represented by the community, in order to familiarize the treebank with intra-linguistic variation.

4.3.2 Morphology and Tokenization

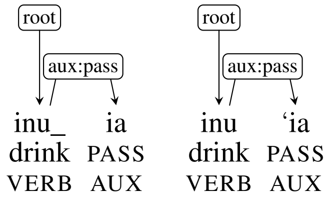
Hawaiian's particular morphology raised several questions as to the best route for tokenization. Whereas morphological inflection's marginality and non-productivity motivated a simple analysis of morphological processes, the presence of non-concatenative morphology (e.g. vowel lengthening, reduplication) presented a more complex situation that merits comment.

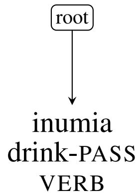
Inflectional morphology surfaces as vowel lengthening (e.g. *wahine* 'woman' vs. *wāhine* 'women') and reduplication, both partial and total (e.g. *mele* 'sing' vs. *memele* 'sing (pl.)'; *'oki* 'shear, cut once' vs. *'oki'oki* 'dice, cut repeatedly'). Vowel lengthening is reserved for a closed class of roots, and only represented orthographically; items were represented with the plural feature only when rendered with modern orthography. Because reduplication is no longer a productive process, and because certain idiosyncratic meanings were specific to certain roots, reduplicated forms were kept as single morphemes for parsing purposes. Example (b) shows two possible parses for this; the second option was ultimately chosen.

(b) 

Two possible parses for:
"to dice, to cut repeatedly"

All prefixes are universally represented as root-attached in texts; we treated prefixed verbs as single items accordingly. There was more variation for suffixes, exemplified particularly by the case of Hawaiian's passive morphemes. There are two types of passives: suffixes {-'ia} and {-Cia}. In modern texts, {-'ia} is usually written as a separate word (e.g. *inu 'ia* '[for something] to be drunk'), but in older texts it may appear as a bound suffix on the verb (e.g. *inuia*). We chose to represent -*'ia* as a separate unit linked by aux:pass to the verb, choosing to align older documents with contemporary norms as in (c). Conversely, {-Cia} is a fossilized passive no longer productive in Hawaiian, limited to just a few specific roots (e.g. *inumia* '[for something] to be drunk'). Since the morpheme itself is fossilized and its phonological shape depends on a given root, we chose to keep it root-attached as shown in (d).

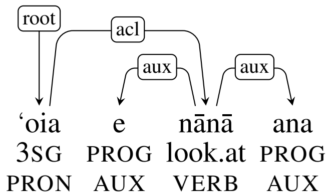
(c) 

(d) 

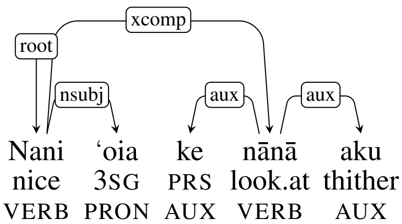
Parses for "to be drunk" using (c) separate *'ia* for both modern and historical representations and (d) fossilized passive suffixes (e.g. -*mia*).

4.3.3 Dependent Clauses

There are numerous issues involved in tagging dependent clauses. For example, Hawaiian has sentences similar to English *tough*-constructions like *Linguists are tough to please* where "an apparently "missing" object" of an embedded infinitival clause is "obligatorily interpreted as coreferential with the matrix subject" (Hicks, 2009, 535). For these, it is unclear whether a dependent should be connected to either a nominal or verbal element. In (e) below, the relative clause *e nānā ana* "(that) t_i is looking" describes the 3rd-person singular pronoun *'oia*.

(e) 

"S/he, who is looking"

(f) 

"S/he is nice to look at"

The dependent clause in (e) syntactically parallels that in (f), but with a different semantic interpretation. In (f), the dependent clause describes *'oia*, similar to the meaning of the English *tough*-construction: "to look at" is a modifier of how "nice" to look at the person is, here expressed by the matrix stative verb *nani*.

5 Conclusions and Future Work

In this paper, we presented a treebank for the Eastern Polynesian language Hawaiian, and used this dataset, along with a treebank in Cook Islands Māori, to construct a crosslingual model to parse both languages. The crosslingual model produced a statistically significant increase in performance for Hawaiian in comparison to the monolingual model. These gains in Hawaiian neither helped nor hurt the performance on CIM.

The zero-shot approach using an unrelated language (English) did not result in any remarkable increase to model performance. A zero-shot model of a related language might still be required to get initial parses when constructing new treebanks from scratch.

Much future work remains for this project. As for the models, we need to perform fine-tuning tests on LLMs (e.g. BERT) and test if they will provide improvements in performance over UDPipe2. Labeling using LLM prompting is more complex, as it touches upon issues of data sovereignty, and the transmission and potential use of this information by the companies that host the LLMs. As for the treebanks themselves, both of them need to be tagged for their morphological features (UFeats), and expanded so that they can dependably label larger collections of data. The Hawaiian language has thousands of pages of text, especially in historical newspaper collections (Shillingford and Parker Jones, 2018), available for parsing; similarly, there is a wealth of legacy information available for Cook Islands Māori that could be analyzed with these parsers.

Evidently the Hawaiian treebank is still extremely small, and the models here will be used for bootstrapping and expanding the treebank. In the work with CIM, we used the Karnes et al. (2023) model to obtain preliminary parses, correct them, and include these newly parsed sentences in the treebank. This approach has been very fruitful in expanding the CIM dataset, from 126 sentences in Karnes et al. (2023) to the present 663. We intend to use this approach for Hawaiian as well, and use these new crosslingual models to accelerate the construction of the Hawaiian treebank even further. Going forth, we intend to coordinate the efforts of the Hawaiian and CIM teams during the annotation, so that any corrections due to improved understanding of linguistic structures can find their way into both sets.

We also intend to expand the domains from which sentences come from to include more varieties of data (e.g. spoken sentences typical of transcribed interviews), so that we can ultimately release these models to the interested stakeholders in their respective communities.

At present, our priority for the Hawaiian parser is to facilitate its access and use by community scholars and organizations involved in language revitalization and pedagogical work. Heeding present intra-community concerns about data stewardship (Alegado et al., 2023), we hope to arrive at a wider consensus among stakeholders before releasing the Hawaiian model for wider distribution and use. As for the CIM, sharing the treebank and the model publicly presents similar concerns, and more consensus is needed before its release. This work is part of a larger project to train linguists and NLP specialists in the Cook Islands, who can collaborate with other scholars from Polynesia in the documentation of their languages.

We hope that this work will be used not only to tag collections in Hawaiian and CIM, but also to foster work in NLP in other Polynesian and Indigenous languages, accelerating documentation work to contribute to language revitalization, normalization, and reclamation efforts in the Pacific and worldwide.

Limitations

The treebanks are largely restricted to written data. While some sentences come from oral interviews, the parsers may still face issues parsing unmodified depictions of spoken language. This limits their applications to naturalistic speech data.

Replicating this project might be difficult in some communities given computational resource demands. To calculate our results, we required 207 hours of GPU time (NVIDIA Tesla K80).

Acknowledgments

We would like to express our heartfelt thanks to all those involved in Hawaiian revitalization, without whose labor this work would not be possible. Mahalo nui loa.

References

Rosie Alegado, Katy Hintzen, Sara Kahanamoku, and Kaleo Hurley. 2023. Kūlana noiʻi: Indigenous data stewardship in hawaiʻi.

Kaavya Chaparala, Guido Zarrella, Bruce Torres Fischer, Larry Kimura, and ʻŌiwi Parker Jones. 2024. Mai Hoʻomāuna i ka ʻAi: Language Models Improve Automatic Speech Recognition in Hawaiian. *arXiv preprint arXiv:2404.03073*.

Rolando Coto-Solano, Sharid Loáiciga, and Sofía Flores-Solórzano. 2021. Towards universal dependencies for bribri. In *Proceedings of the Fifth Workshop on Universal Dependencies (UDW, SyntaxFest 2021)*, pages 16–29.

Rolando Coto-Solano, Sally Akevai Nicholas, Samiha Datta, Victoria Quint, Piripi Wills, Emma Ngakuravaru Powell, Liam Kokaʻua, Syed Tanveer, and Isaac Feldman. 2022a. Development of automatic speech recognition for the documentation of Cook Islands Māori.

Rolando Coto-Solano, Sally Akevai Nicholas, Brittany Hoback, and Gregorio Tiburcio Cano. 2022b. Managing data workflows for untrained forced alignment: examples from costa rica, mexico, the cook islands, and vanuatu. *The Open Handbook of Linguistic Data Management*, 35.

Rolando Coto-Solano, Sally Akevai Nicholas, and Samantha Wray. 2018. Development of natural language processing tools for Cook Islands Māori. In *Proceedings of the Australasian Language Technology Association Workshop 2018*, pages 26–33, Dunedin, New Zealand.

Cheikh M Bamba Dione. 2021. Multilingual Dependency Parsing for Low-Resource African Languages: Case Studies on Bambara, Wolof, and Yoruba. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 84–92.

Samuel H. Elbert and Mary Kawena Pukui. 1979. *Hawaiian Grammar*. Univ. of Hawaii Press.

Aoife Finn, Peter-Lucas Jones, Keoni Mahelona, Suzanne Duncan, and Gianna Leoni. 2022. Developing a part-of-speech tagger for te reo māori. In *Proceedings of the Fifth Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 93–98.

GeminiTeam, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1331 others. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Gabriel H. Gilbert. 2023. Directionals in Spoken Hawaiian: A Corpus Analysis. B.A. thesis, Dartmouth College.

Glyn Hicks. 2009. Tough-constructions and their derivation. *Linguistic Inquiry*, 40(4):535–566.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Kelsea Kanohokuahiwi Hosoda. 2019. *Hawaiian morphemes: Identification, usage, and application in information retrieval*. Ph.D. thesis, University of Hawaiʻi at Manoa.

Jesin James, Rolando Coto-Solano, Sally Akevai Nicholas, Joshua Zhu, Bovey Yu, Fuki Babasaki, Jenny Tyler Wang, and Nicholas Derby. 2024. Development of community-oriented text-to-speech models for māori ʻavaiki nui (cook islands māori). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4820–4831.

Peter-Lucas Jones, Keoni Mahelona, Suzanne Duncan, and Gianna Leoni. 2023. Kia tangata whenua: Artificial intelligence that grows from the land and people.

Kauanoe Kamanā and William H. Wilson. 2001. "Mai Loko Mai O Ka ʻIʻini: Proceeding from a Dream": The ʻAha Pūnana Leo Connection in Hawaiian Language Revitalization. In *The Green Book of Language Revitalization in Practice*, pages 147–176. Brill.

Clinton Kanahele. 1970. Clinton Kanahele Collection. University Archives, Joseph F. Smith Library.

Sarah Karnes, Rolando Coto-Solano, and Sally Akevai Nicholas. 2023. Towards universal dependencies in cook islands māori. In *Proceedings of the Sixth Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 124–129.

Tūreiti Keith. 2024. Work in progress: Text-to-speech on edge devices for te reo māori and ʻōlelo hawaiʻi. In *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages@ LREC-COLING 2024*, pages 421–426.

Larry L. Kimura. 1983. Native Hawaiian Culture. In *Report on the Culture, Needs and Concerns of Native Hawaiians*, volume 1 of *Native Hawaiians Study Commission*. Department of the Interior.

R. Keao NeSmith. 2019. Take My Word: Mahalo No i Toʻu Matua Tane. *Linguapax Review*, 7:93–111.

Sally Akevai Nicholas. 2017. *Ko Te Karāma o Te Reo Māori o Te Pae Tonga o Te Kuki Airani: A Grammar of Southern Cook Islands Måori*. Ph.D. thesis, University of Auckland.

Sally Akevai Nicholas and Rolando Coto-Solano. 2019. Glottal variation, teacher training and language revitalization in the cook islands. In *Proceedings of the 19th International Congress of Phonetic Sciences, University of Melbourne, Australia*, pages 3602–3606.

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 International Conference on Language Resources and Evaluation (LREC 2020)*, pages 4034–4043.

OpenAI. 2022. Introducing ChatGPT.

Yuko Otsuka. 2005. History of polynesian languages. *Linguistics*, 345:267–296.

Mary Kawena Pukui and Samuel H. Elbert. 1986. *Hawaiian Dictionary: Hawaiian-English, English-Hawaiian*. University of Hawaii Press.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.

Neha Ramsurrun, Rolando Coto-Solano, and Michael Gonzalez. 2024. Parsing for mauritian creole using universal dependencies. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 12622–12632.

Lorena Martín Rodríguez, Tatiana Merzhevich, Wellington Silva, Tiago Tresoldi, Carolina Aragon, and Fabrício F Gerardi. 2022. Tupían language ressources: Data, tools, analyses. In *Proceedings of the 1st Annual Meeting of the ELRA/ISCA Special Interest Group on Under-Resourced Languages*, pages 48–58.

Brendan Shillingford and ʻŌiwi Parker Jones. 2018. Recovering missing characters in old Hawaiian writing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4929–4934.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Milan Straka. 2025. Universal Dependencies 2.15 Models.

Tai Tepuaotera Turepu Carpenter and Clive Beaumont. 1995. *Kai Kōrero*.

Francis Tyers and Robert Henderson. 2021. A corpus of k'iche'annotated for morphosyntactic structure. In *Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas*, pages 10–20.

William H Wilson. 1981. Developing a standardized Hawaiian orthography. *Pacific Studies*, 4:19–19.

Author Index