Dynamic Head Selection for Neural Lexicalized Constituency Parsing

Yang Hou and **Zhenghua Li***
School of Computer Science and Technology,
Soochow University, China
yhou1@stu.suda.edu.cn zhli13@suda.edu.cn

Abstract

Lexicalized parsing, which associates constituent nodes with lexical heads, has historically played a crucial role in constituency parsing by bridging constituency and dependency structures. Nevertheless, with the advent of neural networks, lexicalized structures have generally been neglected in favor of unlexicalized, span-based methods. In this paper, we revisit lexicalized parsing and propose a novel latent lexicalization framework that dynamically infers lexical heads during training without relying on predefined head-finding rules. Our method enables the model to learn lexical dependencies directly from data, offering greater adaptability across languages and datasets. Experiments on multiple treebanks demonstrate state-of-the-art or comparable performance. We also analyze the learned dependency structures, headword preferences, and linguistic biases.

1 Introduction

Constituency parsing is a fundamental task in natural language processing (NLP), aiming to represent the syntactic structure of a sentence as a hierarchical tree. Parsing is beneficial for numerous downstream tasks, including machine translation (Wang et al., 2007; Currey and Heafield, 2019), semantic role labeling (Zhou et al., 2020), and named entity recognition (Li et al., 2017).

Historically, early approaches to constituency parsing emphasized **lexicalized parsing**, where headwords—key words within phrases—played a central role in guiding syntactic analysis (Magerman, 1995; Charniak, 1997; Collins, 1996, 1997; Charniak, 2000, 2001; Gildea, 2001; Collins, 2003). Headwords encapsulate critical syntactic and semantic information, enabling the resolution of complex phenomena such as agreement and long-distance dependencies (Hindle and Rooth, 1993).

* Corresponding author

However, the emergence of **unlexicalized parsing** models challenged the necessity of lexical information. Research demonstrated that unlexicalized models could achieve comparable or superior performance, often with reduced complexity and better generalization across datasets (Klein and Manning, 2003; Petrov and Klein, 2007).

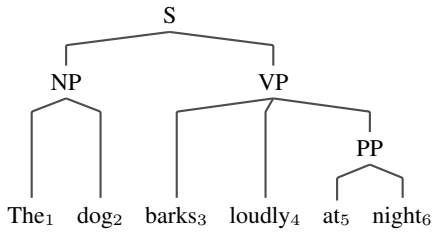
Despite their historical significance, lexicalized structures are largely ignored in modern neural approaches. State-of-the-art constituency parsers predominantly adopt unlexicalized, span-based methods, leveraging pre-trained language models (PLMs) such as BERT (Devlin et al., 2019). These PLMs inherently encode rich lexical and syntactic dependencies, prompting a reassessment of the role of lexicalization in parsing. Specifically, two key questions arise:

1. Does modeling bilexical dependencies still offer benefits for neural constituency parsing?
2. Can the structural dependency representations implicitly encoded in PLMs be effectively utilized to enhance parsing?

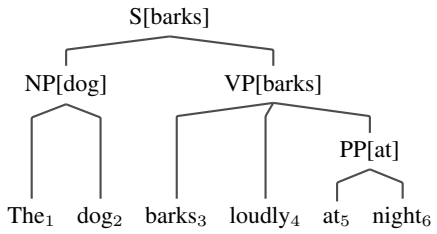
To address the first question, we reimplement Collins-formalism of lexicalized parsing with neural models. For the second question, we propose a novel latent lexicalization framework for neural constituency parsing. Unlike traditional methods that rely on fixed head-finding rules, our framework introduces **dynamic head selection** during training. By treating head selection as a latent variable, the implicit syntactic knowledge encoded in PLMs are effectively leveraged, eliminating the need for explicit dependency supervision and enabling greater adaptability across languages and datasets. Our contributions are as follows:

- We revisit lexicalized parsing in the context of neural models.
- We propose a latent lexicalization framework that dynamically determines lexical heads, making the process fully data-driven and flexible.
- We demonstrate through extensive experiments

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16141–16155



(a) An unlexicalized constituency tree



(b) A lexicalized constituency tree

Figure 1: Example trees for the sentence "The dog barks loudly at night." Part-of-speech tags are omitted.

on the Penn Treebank (PTB), Chinese Treebank (CTB), and SPMRL datasets that our method achieves state-of-the-art or comparable performance.

- We provide detailed analyses of the learned dependency structures, headword preferences, and linguistic biases.

Our code is publicly available at https://github.com/ironsword666/LatentLexConstParsing.

2 Background

In this section, we provide an overview of the foundational concepts in constituency parsing and lexicalization, and introduce common methods of lexicalization.

2.1 Constituency Parsing

Constituency parsing aims to represent the syntactic structure of a sentence as a hierarchical tree. Given a sentence $\boldsymbol{w} = w_1 w_2 \ldots w_n$, where w_i is the i-th word, the task involves generating a tree $\boldsymbol{t}$ where each node represents a linguistic unit (word or phrase) labeled with its syntactic category. For example, in the sentence "The dog barks loudly at night", the noun phrase (NP) "The dog" and the verb phrase (VP) "barks loudly at night" are constituents. Each constituent is defined by a span of words (i, j) and a syntactic category l, represented as the tuple (i, j, l), where i and j denote the start and end word positions. An example constituency tree is shown in Figure 1a.

2.2 Lexicalization in Parsing

Lexicalization enhances constituency parsing by associating each constituent with a lexical head—a specific word that carries the most syntactic and semantic importance. For instance:

- Unlexicalized rule: $\mathrm{S} \rightarrow \mathrm{NP\,VP}$
- Lexicalized rule: $\mathrm{S_{barks}} \rightarrow \mathrm{NP_{dog}\,VP_{barks}}$

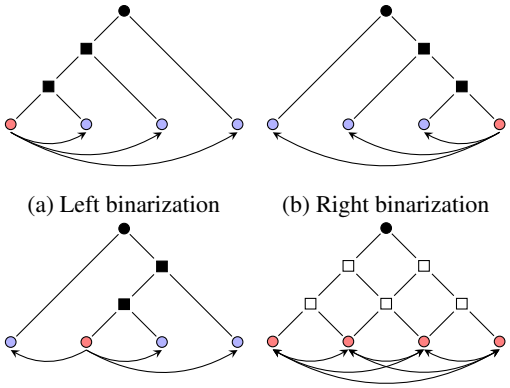
Here, "barks" serves as the head of the sentence (S), while "dog" is the head of the NP node. These lexical heads propagate from the leaves to the root of the tree, forming a lexicalized tree structure. Figure 1 contrasts unlexicalized and lexicalized trees. By enriching constituency parsing with lexical heads, lexicalization bridges the gap between constituency parsing and dependency parsing, providing a unified framework to represent both phrase structure and head-dependent relations. In the lexicalized tree $\boldsymbol{t}$ (the same symbol is reused), each constituent is represented as a headed span (i, j, h, l), where h denotes the position of the headword. Based on these headed spans, head-dependent relations $h \rightarrow m$ can be extracted, yielding a complete dependency tree as a by-product.

2.3 Lexicalization Methods

Lexicalization in parsing relies on manual annotations or fixed heuristics to identify the lexical head of each constituent.

- **Human Annotation**: Dependency trees are annotated in parallel with constituency trees. While accurate, this method is labor-intensive, and consequently, such parallel treebanks are rare or virtually non-existent.
- **Naive Heuristics**: Simple methods such as always assigning the leftmost or rightmost child as the head are efficient but rarely align with linguistic intuition.
- **Linguistic Heuristics**: Handcrafted head-finding rules, such as those used by Collins (2003), determine lexical heads based on linguistic principles (e.g., the rightmost noun in a noun phrase is the head).

Among these methods, head-finding rules are most commonly used in research, as they obtain a balance between computational efficiency and linguistic precision.



(a) Left binarization (b) Right binarization

(c) Head binarization (d) Latent binarization

Figure 2: Different binarization and lexicalization methods for a parent node with four children (circles). Squares represent intermediate nodes. Solid shapes indicate determined states, while hollow shapes indicate undetermined states.

3 Latent Lexicalized Constituency Parsing

3.1 Proposed Latent Lexicalization

Linguistic heuristics, while effective, often fail to generalize across diverse datasets and syntactic variations. Treebanks are annotated using different guidelines, and languages exhibit inherent structural differences (Dixon, 2010). Moreover, multiple dependency representations, such as Stanford Dependencies (de Marneffe et al., 2006) and Universal Dependencies (de Marneffe et al., 2021), can exist for the same sentence, highlighting the need for a general and flexible lexicalization framework.

To address these challenges, we propose a latent lexicalization framework that treats the lexical head of a constituent as a latent variable. Key features of this approach include:

- **Dynamic Head Selection**: In our framework, the lexical head of a constituent is not predefined. For a constituent $A \rightarrow B_1 B_2 \ldots B_r$, any child B_i can serve as the head. This flexibility enables the model to dynamically select the most appropriate head, effectively utilizing implicit syntactic knowledge encoded in PLMs.
- **Representation as A Forest**: Instead of constructing a single lexicalized tree, we represent each sentence as a forest of possible lexicalized trees, where each tree corresponds to a valid head assignment. During training, the model optimizes over this forest, learning head assignments that maximize parsing performance.

This data-driven approach generalizes effectively across languages and datasets, overcoming the rigidity of traditional approaches.

3.2 Binarization of Lexicalized Trees

Efficient parsing algorithms, such as the Eisner-Satta algorithm (Eisner and Satta, 1999) used in this paper, require constituency trees to be binarized into Chomsky normal form (CNF).[1] The binarization process varies depending on how lexical heads are determined, as shown in Figure 2.

- **Left Binarization**: The leftmost child is selected as the head of the constituent. The tree is recursively binarized by recursively grouping the leftmost child with one sibling at a time, creating new intermediate nodes.[2]
- **Right Binarization**: Assign the rightmost child as the head. The tree is transformed by grouping the rightmost child with one sibling.
- **Head Binarization**: Head-finding rules are used to identify the head, and these heads guide how the nodes are binarized. The detailed process is described in Appendix A.1.

Latent binarization In our latent lexicalization framework, where head selection is dynamic, binarization must account for all possible head assignments. This leads to a **latent binarization**[3] approach, which considers all valid binary decompositions of an n-ary node during training. For instance, the constituent $A \rightarrow B_1 B_2 B_3$ can be binarized as:

- $A \rightarrow A_1 B_3$, $A_1 \rightarrow B_1 B_2$
- $A \rightarrow B_1 A_1$, $A_1 \rightarrow B_2 B_3$

This approach ensures consistency between the phrase structures and the learned dependency structures. In other words, for each head assignment, there exists a corresponding binary decomposition.

To clarify the notion of "latent" in this context: 1) In latent lexicalization, any two children of a parent node can form a potential dependency. 2) In latent binarization, for a span (i, j) in an n-ary tree, any span that crosses it is forbidden in the resulting binary tree. For example, the span $(1, 3)$ crosses with span $(2, 4)$, and such crossings are disallowed.

[1] For a discussion on different binarization strategies and their effects on parsing, see Chen et al. (2021).

[2] This process ensures that all children depend on the leftmost child.

[3] This differs from the **implicit binarization** (Stern et al., 2017), where an empty label $\varnothing$ represents all intermediate nodes in binary tree decomposition.

3.3 Incorporating Linguistic Knowledge

While the latent lexicalization framework is data-driven, incorporating linguistic Knowledge can further refine the head selection.

Punctuation Exclusion Punctuation marks are excluded as potential heads because they do not govern other words. This common linguistic constraint is applied across languages.

Content vs. Function Words Content words (e.g., nouns, verbs) are more likely to serve as heads than function words (e.g., determiners, conjunctions). Introducing a bias toward content words helps the model prioritize semantically significant words when selecting heads.

Phrase-Level Preferences Linguistic principles can guide head selection for specific phrase types. For example, in verb phrases VP $\rightarrow$ VP NP PP, objects (NP) or prepositions (PP) are less likely to serve as heads compared to the main verb (VP). Thus, NP and PP are forbidden as heads in this scenario. These preferences resemble traditional head-finding rules but are implemented as soft biases, allowing the model to adapt to diverse syntactic structures. For instance, the attachment of a PP to either a VP or an NP remains flexible, depending on the data.

For simplicity and cross-linguistic applicability, only punctuation exclusion is used in this work.

4 Model

In this section, we describe the scoring mechanism, model architecture, training, and decoding process.

4.1 Scoring factorization

We adopt a two-stage chartb-based framework for lexicalized parsing, following Dozat and Manning (2017) and Zhang et al. (2020). In this framework, constituent brackets are predicted first, followed by the labeling of these constituents. The overall score for a (unlabeled) lexicalized tree $\boldsymbol{t}$ is decomposed into constituency and dependency scores:

$$\begin{aligned} s(\boldsymbol{t}) &= s^{\text{con}}(\boldsymbol{t}) + s^{\text{dep}}(\boldsymbol{t}) \\ &= \sum_{(i,j)\in\boldsymbol{t}} s(i,j) + \sum_{(h\rightarrow m)\in\boldsymbol{t}} s(h\rightarrow m) \end{aligned} \tag{1}$$

where $s(i,j)$ is the score of constituent span (i,j), and $s(h \rightarrow m)$ denotes the score of a dependency arc from a head h to its dependent m.

This decomposition enables the use of Eisner-Satta algorithm (Appendix A.2), a chart-based dynamic programming approach with a time complexity of $O(n^4)$ for both training and decoding.[4]

4.2 Model Architecture

For an input sentence $\boldsymbol{w} = w_0 w_1 \ldots w_n$, where w_0 is an artificial `ROOT` token, we first obtain contextualized representation of each word w_i using a pre-trained language model (BERT in this work).

$$\mathbf{x}_i = \text{BERT}(w_i) \tag{2}$$

The representations[5] are then used to compute scores for candidate constituents and dependencies using biaffine attention mechanisms (Dozat and Manning, 2017).

$$\begin{aligned} \mathbf{x}_i^{\text{con-left}} &= \text{MLP}^{\text{con-left}}(\mathbf{x}_i) \\ \mathbf{x}_j^{\text{con-right}} &= \text{MLP}^{\text{con-right}}(\mathbf{x}_j) \\ \mathbf{x}_h^{\text{dep-head}} &= \text{MLP}^{\text{dep-head}}(\mathbf{x}_h) \\ \mathbf{x}_m^{\text{dep-mod}} &= \text{MLP}^{\text{dep-mod}}(\mathbf{x}_m) \end{aligned} \tag{3}$$

Scores are computed for constituency spans (i,j) and dependency arcs $h \rightarrow m$:

$$\begin{aligned} s(i,j) &= \text{BiAffine}^{\text{con}}(\mathbf{x}_i^{\text{con-left}}, \mathbf{x}_j^{\text{con-right}}) \\ s(h{\rightarrow}m) &= \text{BiAffine}^{\text{dep}}(\mathbf{x}_h^{\text{dep-head}}, \mathbf{x}_m^{\text{dep-mod}}) \end{aligned} \tag{4}$$

Additionally, constituent label scores $s(i,j,l)$ are computed using another biaffine mechanism.

4.3 Training

We train the model using conditional random fields (CRF) to maximize the probability of the gold-standard tree. The probability of a lexicalized tree $\boldsymbol{t}$ is defined as:

$$p(\boldsymbol{t}|\boldsymbol{w}) = \frac{\exp(\text{s}(\boldsymbol{t}))}{\boldsymbol{Z}(\boldsymbol{w}) \equiv \sum_{\boldsymbol{t}'} \exp(\text{s}(\boldsymbol{t}'))} \tag{5}$$

where $\boldsymbol{Z}(\boldsymbol{w})$ is the partition function summing over all possible trees. When headwords are explicitly specified, the tree probability directly supervises the model. When headwords are latent, we optimize over a forest $\mathcal{T}$ of valid lexicalized trees:

$$\begin{aligned} p(\mathcal{T}|\boldsymbol{w}) &= \frac{\sum_{\boldsymbol{t}^*\in\mathcal{T}} \exp(\text{s}(\boldsymbol{t}^*))}{\boldsymbol{Z}(\boldsymbol{w})} \\ \mathcal{L}_{\text{bracket}}(\theta) &= -\log p(\mathcal{T}|\boldsymbol{w}) \end{aligned} \tag{6}$$

[4] Alternatively, scores can be factored over headed spans (i,j,h), but this would require a CKY-style algorithm (Collins, 1997), leading to a higher computational complexity ($O(n^5)$).

[5] We use fencepost representaions for constituents, as described in Stern et al. (2017).

The summation process is performed using a Inside-version of Eisner-Satta algorithm. See Algorithm 1 in Appendix A.2 for details.

For constituent labels, we use cross-entropy loss:

$$\mathcal{L}_{\text{label}}(\theta) = -\sum_{(i,j,l)\in \boldsymbol{t}} \log p(l|i,j) \tag{7}$$

The final loss combines both objectives:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{bracket}}(\theta) + \mathcal{L}_{\text{label}}(\theta) \tag{8}$$

4.4 Inference

During inference, we use Eisner-Satta algorithm to decode the best lexicalized tree:

$$\hat{\boldsymbol{t}} = \arg\max_{\boldsymbol{t}} s(\boldsymbol{t}) \tag{9}$$

Then, the optimal label for each constituent (i, j) is predicted as:

$$\hat{l}_{i,j} = \arg\max_{l} s(i,j,l) \tag{10}$$

In addition, head-dependent relations can be directly extracted from the predicted lexicalized tree.

5 Experiments

5.1 Setup

Data. We conduct experiments on benchmarks for multiple languages. For English, we use Penn Treebank 3.0 (PTB) (Marcus et al., 1993), following the standard splits (sections 2–21 for training, section 22 for development, and section 23 for testing). For Chinese, the Chinese Treebank 5.1 (CTB5) (Xue et al., 2005), split according to the common practice. We also evaluate our model on six morphologically rich languages (French, German, Korean, Hungarian, Basque, and Polish) from the SPMRL dataset (Seddah et al., 2013), using the provided splits. The statistics of the datasets are shown in the Appendix B.1.

Evaluation Metrics Parsing performance is evaluated using the standard precision, recall and F1 score metrics with the `EVALB` tool[6] for PTB and CTB, and the `EVALB_SPMRL` script[7] for SPMRL.

Implementation Details For English and Chinese, we extract dependencies with Stanford parser v3.3.0[8]. And for SPMRL, we use the provided gold-standard dependencies. Additional implementation details, including hyperparameter settings, optimization strategies, and computational resources, are provided in Appendix B.1.

5.2 Main Results

Effectiveness of Lexicalization We evaluate the impact of different lexicalization methods by comparing the following models, all trained with a CRF loss. The results of the max-margin training are presented in the Appendix B.2 for details.

- **Con**: A baseline unlexicalized parser, following the approach of Zhang et al. (2020).
- **Con (Latent)**: An extension of Con, incorporating latent binarization (see Section 3.2).
- **Lex (Head)**: A lexicalized parser that uses prespecified lexical heads, extracted or annotated.
- **Lex (Left)**: Assigns the leftmost child as the head for each constituent.
- **Lex (Right)**: Assigns the rightmost child as the head for each constituent.
- **Lex (Latent)**: Uses latent binarization and latent lexicalization to infer heads during training.

Table 1 presents the results on PTB, CTB5 and SPMRL datasets. Surprisingly, using predetermined lexical heads (Lex (Head)) degrades performance compared to the unlexicalized baseline (Con), suggesting that rigid heuristic rules may not align well with the implicit syntactic knowledge encoded in PLMs. Assigning the leftmost (Lex (Left)) or rightmost (Lex (Right)) child as the head yields slight improvements on some languages, despite being linguistically simplistic. Our latent lexicalization approach (Lex (Latent)) consistently outperforms other methods across all languages, except for two low-resource languages, Basque and Polish. This demonstrates the advantages of dynamically inferred heads over fixed head assignments. These findings highlight the limitations of traditional head-finding heuristics in the context of PLMs. While there may be room for improvement in leveraging rule-extracted headwords, the results emphasize the promise of latent lexicalization as a flexible and effective strategy for improving constituent lexicalization in neural parsing models.

Compared with previous work Tables 2, 3 and 4 compare our model with state-of-the-art parsers

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

[7] https://www.spmrl.org/spmrl2013-sharedtask.html

[8] https://nlp.stanford.edu/software/lex-parser.shtml

Model	English	Chinese	French	German	Korean	Hungarian	Basque	Polish
Con	95.81	91.96	87.25	90.36	89.38	94.54	91.20	95.94
Con (latent)	95.86	91.68	87.88	90.41	89.42	94.79	91.19	**95.98**
Lex (head)	95.77	91.67	87.30	90.33	89.28	94.32	91.16	95.62
Lex (left)	95.79	92.00	87.79	90.34	89.31	94.69	**91.34**	95.68
Lex (right)	95.82	92.07	87.29	90.52	89.43	94.82	91.13	95.76
Lex (latent)	**96.02**	**92.34**	**87.91**	**90.53**	**89.51**	**94.83**	91.28	95.79
Δ vs. Con (latent)	+0.16	+0.66	+0.03	+0.17	+0.09	+0.04	+0.09	-0.19

Table 1: Results of different lexicalization methods on PTB (English), CTB5 (Chinese) and SPMRL (covering French, German, Korean, Hungarian, Basque and Polish). Only the F1 scores are reported.

Model	P	R	F1
Kitaev et al. (2019)	95.46	95.73	95.59
Zhou and Zhao (2019)	95.70	95.98	95.84
Zhang et al. (2020)	95.85	95.53	95.69
Xin et al. (2021)	**96.29**	95.55	95.92
Cui et al. (2022)	95.70	**96.14**	95.92
Yang and Tu (2022)	96.19	95.83	96.01
Yang and Tu (2023)	96.21	95.87	**96.04**
Ours	96.19	95.85	96.02

Table 2: Results on PTB test set.

Model	P	R	F1
Kitaev et al. (2019)	91.96	91.55	91.75
Zhang et al. (2020)	92.51	92.04	92.27
Yang and Deng (2020)	**93.80**	**93.40**	**93.59**
Xin et al. (2021)	92.94	92.06	92.50
Cui et al. (2022)	92.45	92.17	92.31
Yang and Tu (2023)	92.83	91.97	92.41
Ours	92.60	91.87	92.34

Table 3: Results on CTB5 test set.

on PTB, CTB, and SPMRL datasets. On PTB, our model achieves a F1 score of 96.02, comparable to the best-performing parser (Yang and Tu, 2023). For CTB5, our method achieves competitive results, demonstrating its effectiveness across languages. On the multilingual SPMRL dataset, our parser outperforms previous methods on French and Korean, and achieves strong results on other languages, showcasing its robustness in morphologically rich settings.

6 Analysis

In this section, we analyze the prediction behavior of our model to better understand the effects of latent lexicalization. Specifically, we investigate the head selection patterns and its consistency with Stanford Dependencies (SD) on PTB test set.

6.1 Lexcial Head Selelction

We analyze the predicted head selection patterns from two perspectives: (1) the headword choices for different phrase types, and (2) the dependency relationships inferred between POS tags. The gold-standard references can be found in the Appendix B.4.

Phrase Head Preferences We investigate head selection for various types of constituents.[9] The results, shown in Table 5, reveal several patterns:

- Noun Phrase (NP): The model predominantly selects the noun as the head, aligning with linguistic expectation.
- Verb Phrase (VP): The main verb is usually chosen as the head, but auxiliary verbs (e.g., is, has) are also frequently identified as heads.
- Prepositional Phrase (PP): The model frequently selects the noun phrase (NP) within the PP as the head. This behavior is consistent with the Universal Dependencies (UD) annotation scheme.

These results suggest that the model captures certain linguistic norms, though it does not always strictly adhere to them.

Dependency Relationship To further investigate head selection, we examine the predicted dependencies between POS tags. Table 6 outlines frequent dependency relationships. Some dependencies align with linguistic conventions, such as:

- The dependency between verbs and nouns (VB $\curvearrowright$ NN).

[9] Inherited heads from unary chains (e.g., NP $\rightarrow$ NNS) have minimal impact on the results.

Model	Rich Resource			Low Resource			Avg
	French	German	Korean	Hungarian	Basque	Polish	
Kitaev et al. (2019)	87.42	90.20	88.80	94.90	91.63	**96.36**	91.55
Nguyen et al. (2020)	86.69	90.28	88.71	94.24	**92.02**	96.14	91.34
Cui et al. (2022)	87.51	90.43	89.07	94.95	91.73	96.33	91.67
Yang and Tu (2023)	87.89	**91.07**	89.31	**95.06**	91.72	96.18	**91.87**
Ours	**87.91**	90.53	**89.51**	94.83	91.28	95.79	91.64

Table 4: Results on SPMRL test set. Only the F1 scores are reported.

Phrase	Head	Count	Ratio (%)	Headword
NP	NN	5382	29.0	it, "%",
	NNS	3123	16.9	the, company,
	NNP	2795	15.1	he, market,
	NP	2511	13.6	share, million
VP	VP	2453	28.0	said, is,
	VBD	1220	13.9	be, are,
	VB	973	11.1	have, was,
	VBZ	690	7.9	says, for
S	VP	5265	93.4	said, is,
	S	221	3.9	are, was
PP	NP	4570	83.3	in, of,
	IN	552	10.1	market, for
ADVP	RB	1027	83.5	also, ago,
	RBR	43	3.5	now, still
ADJP	JJ	420	49.3	"$", million,
	S	51	6.0	more, billion
WHNP	WDT	272	63.0	that, which,
	WP	106	24.5	who, what

Table 5: Most frequent heads for each constituent label, along with high-frequency headwords (listed in left-to-right, top-down order).

Head ↷ Dependent	Count	Ratio (%)
NN ↷ DT	3250	5.7
NNP ↷ NNP	2837	5.0
NN ↷ IN	1871	3.3
NN ↷ NN	1506	2.7
NNS ↷ IN	1231	2.2
NNS ↷ NN	973	1.7
NN ↷ JJ	959	1.7
VBD ↷ .	901	1.6
VB ↷ NN	885	1.6
ROOT ↷ VBD	880	1.5

Table 6: Dependency relationships between POS tags.

Model	UAS	UUAS
Ours	37.0	51.3
for reference		
Wu et al. (2020)	41.7	52.1

Table 7: Dependency parsing performance. Wu et al. (2020) use a syntax probing method on the English Parallel Universal Dependencies (PUD) treebank.

- The tendency for verbs to serve as the head of sentences (ROOT ↷ VBD).

However, certain dependencies contradict commonly used Stanford Dependencies. For instance:

- Dependencies between nouns and prepositions (NN ↷ IN) contradict the expected head direction, where prepositions typically govern nouns.

These inconsistencies do not necessarily indicate a failure to learn dependency relationships. Intriguingly, the prediction preference for nouns as heads of prepositional phrases aligns with Universal Dependencies conventions. This alignment offers practical advantages for resolving classic PP attachment ambiguities. For example, in the sentence *"They saw a cat with a telescope"*, to facilitate disambiguation, the model requires prioritizing semantically meaningful associations (*saw* ↔ *telescope*, *cat* ↔ *telescope*) over syntactically functional connections (*saw* ↔ *with*, *cat* ↔ *with*).

6.2 Consistency with Stanford Dependencies

Parsing Performance We compare the predicted dependencies against Stanford Dependencies using unlabeled attachment score (UAS) and undirected UAS (UUAS) (Klein and Manning, 2004). The results, presented in Table 7, show that the UAS value is relatively low, indicating that the predicted dependencies differ substantially from those in the SD. However, UUAS, which eliminates sensitive to annotation variation, gains a higher score.

Arc Direction The directionality of dependency arcs offers valuable insight into language-specific syntactic patterns (note that arc direction is distinct from head-dependent direction). Dependency arc

Model	Leftward Arc	Rightward Arc
SD	51.8	48.2
Ours	67.0	33.0

Table 8: Ratio of leftward arcs and rightward arcs.

Language	Source	UAS	UUAS
English	C	37.01	51.32
Chinese	C	63.02	70.34
French	C	34.83	52.76
German	C	46.22	58.22
Korean	C	29.70	54.45
Hungarian	C	36.34	49.22
Polish	C	13.10	33.21
Basque	D	44.07	58.92

Table 9: Dependency parsing performance on PTB, CTB5, and SPMRL datasets. C and D denote converted and manually annotated dependency trees, respectively.

direction varies across languages. For example, English is a head-initial language, whereas Chinese is head-final.[10]

To evaluate arc directionality, we calculate the ratio of leftward arcs and rightward arcs. As shown in Table 8, the predicted dependencies predominantly exhibit head-final patterns, aligning with the results in Table 6. Specifically, the model correctly predicts head-initial arcs for verbs governing objects (e.g., eat $\curvearrowright$ apple), However, a significant proportion of noun phrases in the dataset follow a head-final structure (e.g., the $\curvearrowleft$ dog), which influences the overall prediction patterns.

6.3 Dependency Evaluation

We expand the evaluation of dependency parsing to more datasets, including CTB5 and SPMRL. We distinguish between dependency trees that are manually annotated (D) and those converted from constituency trees (C). The results are shown in Table 9. We observe that for CTB5, the dependency performance is quite strong, while the model performs worst in Polish. Basque, with manually annotated dependency trees, shows a fair result.

[10]The annotation formalism also influences arc direction. For instance, Universal Dependencies (UD) often prioritize right-to-left dependencies.

7 Related Work

Lexicalized Parsing Lexicalized parsing has been a cornerstone in syntactic analysis, underpinning many early parsing frameworks (Magerman, 1995; Charniak, 1997; Collins, 1996, 1997). Among these, the Collins formalism stands out as particularly influential, utilizing lexical heads to facilitate syntactic derivations (Collins, 2003). Although initially designed for English and chart-based parsing, these models have been successfully adapted to other languages and transition-based parsing paradigm (Crabbé, 2014, 2015).

Joint Parsing The inherent ability of lexicalized parsing to encode dependency relations has motivated efforts to jointly predict constituency and dependency structures within a unified framework. Some approaches maintain separate representations for dependency and constituency structures while enforcing consistency constraints between them (Strzyz et al., 2019; Fernández-González and Gómez-Rodríguez, 2022). Others adopt a single-tree representation to jointly predict both structures (Zhou and Zhao, 2019; Gu et al., 2024).

Modern Constituency Parsing Building on traditional methods, modern constituency parsers predominantly follow either chart-based (Kitaev and Klein, 2018; Zhang et al., 2020; Xin et al., 2021) or transition-based (Cross and Huang, 2016; Liu and Zhang, 2017) approaches. Transition-based parsers, in particular, have seen various innovations. Some models transform input sentences into linearized tree representations (Wei et al., 2020; Amini and Cotterell, 2022) or pointer sequences (Nguyen et al., 2020; Yang and Tu, 2022). Another promising direction is combinatory parsing (Chen et al., 2021; Chen and Komachi, 2023), where shorter spans are merged into longer spans in a layer-wise manner. Notably, their multi-branching architecture—similar to our work—can be adapted to select the head of each constituent.

Latent Parsing The use of latent representations has proven effective in various NLP tasks. Fu et al. (2020) model nested named entities using a constituency parsing framework. Lou et al. (2022) extend their work by introducing a latent lexicalized tree structure. Zhang et al. (2022) treat semantic arguments as latent single-root dependency trees. These approaches demonstrate the potential of latent parsing frameworks to handle complex linguistic structures without manual annotations.

Compared with these work, our work is the first to apply latent parsing to constituency parsing.

Syntax Probing Syntax probing has emerged as a key methodology for analyzing the linguistic knowledge encoded in PLMs (Tenney et al., 2019; Belinkov, 2022). Probing tasks often rely on linear transformations to map model representations to syntactic parse trees (Hewitt and Manning, 2019; Manning et al., 2020). Utilizing parameter-free methods, researchers have shown that PLMs implicitly learn rich syntactic structures, such as constituency and dependency parses, even without explicit supervision (Wu et al., 2020; Niu et al., 2022; Buder-Gröndahl, 2024). Our work complements syntax probing by explicitly incorporating lexicalized parsing into a neural framework.

8 Conclusion

In this work, we proposed a latent lexicalization framework for constituency parsing, designed to dynamically infer head words during training. By treating head selection as a latent variable, the model can learn lexical dependencies directly from data, eliminating the need for predefined head-finding rules and enhancing adaptability across languages and datasets.

Our experiments on PTB and CTB demonstrate that the latent lexicalization framework consistently outperform traditional lexicalization approaches, such as heuristic-based. Key findings include: 1) The heads inferred by the model successfully align with linguistic conventions for some syntactic structures, such as verbs governing their arguments. However, for preposition attachment, the head selection conflicts with Stanford Dependencies. 2) The model often predicts head-final arcs in English, which is common for nouns governing their modifiers. These results underscore the potential of latent lexicalization to utilize lexical dependency representations encoded in PLMs.

Limitations

Projectivity and Continuous Spans The parsing algorithm (Eisner-Satta) used in this work inherently restricts the model to projective dependency structures. Furthermore, our framework focuses exclusively on continuous spans, meaning it does not account for discontinuous constituents. Discontinuous spans are often associated with non-projective dependencies, which frequently occur in linguistic phenomena such as extraposition or topicalization. As a result, our model may not fully capture certain syntactic configurations that require non-projective or discontinuous representations. To address this limitation, future work could explore alternative parsing algorithms that support non-projective structures and discontinuous spans.

Computational Complexity While the proposed method supports efficient batch processing on GPUs, its theoretical complexity of $O(n^4)$ may limit scalability in scenarios involving long sentences. Although parsing speeds are comparable to existing methods in practice (Appendix B.3), the increased computational overhead could pose challenges for resource-constrained environments.

Headword Consistency The discrepancies observed between the predicted heads and gold-standard heads suggest that the model does not always capture the correct headword information. This inconsistency may propagate errors in downstream tasks, which rely on accurate dependency structures. While incorporating linguistic constraints could improve consistency, doing so may introduce additional complexity and reduce the cross-linguistic generalizability.

Acknowledgements

We thank all the anonymous reviewers for their valuable comments. This work was supported by National Natural Science Foundation of China (Grant No. 62176173 and 62336006), and a Project Funded by the Priority Academic Program Development (PAPD) of Jiangsu Higher Education Institutions.

References

Afra Amini and Ryan Cotterell. 2022. On parsing as tagging. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8884–8900, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Tommi Buder-Gröndahl. 2024. What does parameter-free probing really uncover? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 327–336, Bangkok, Thailand. Association for Computational Linguistics.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island, USA*, pages 598–603. AAAI Press / The MIT Press.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 124–131, Toulouse, France. Association for Computational Linguistics.

Zhousi Chen and Mamoru Komachi. 2023. Discontinuous combinatory constituency parsing. *Transactions of the Association for Computational Linguistics*, 11:267–283.

Zhousi Chen, Longtu Zhang, Aizhan Imankulova, and Mamoru Komachi. 2021. Neural combinatory constituency parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2199–2213, Online. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

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

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, Santa Cruz, California, USA. Association for Computational Linguistics.

Benoit Crabbé. 2014. An LR-inspired generalized lexicalized phrase structure parser. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 541–552, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Benoit Crabbé. 2015. Multilingual discriminative lexicalized phrase structure parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1847–1856, Lisbon, Portugal. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Leyang Cui, Sen Yang, and Yue Zhang. 2022. Investigating non-local features for neural constituency parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2065–2075, Dublin, Ireland. Association for Computational Linguistics.

Anna Currey and Kenneth Heafield. 2019. Incorporating source syntax into transformer-based neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 24–33, Florence, Italy. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

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

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

Robert MW Dixon. 2010. *Basic Linguistic Theory: Volume 2 Grammatical Topics*. Oxford University Press.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2022. Multitask pointer network for multi-representational parsing. *Knowl. Based Syst.*, 236:107760.

Yao Fu, Chuanqi Tan, Mosha Chen, Songfang Huang, and Fei Huang. 2020. Nested named entity recognition with partially-observed treecrfs. In *AAAI Conference on Artificial Intelligence*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*.

Yanggan Gu, Yang Hou, Zhefeng Wang, Xinyu Duan, and Zhenghua Li. 2024. High-order joint constituency and dependency parsing. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8144–8154, Torino, Italia. ELRA and ICCL.

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

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

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.

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

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

Peng-Hsuan Li, Ruo-Ping Dong, Yu-Siang Wang, Ju-Chieh Chou, and Wei-Yun Ma. 2017. Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2664–2669, Copenhagen, Denmark. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017. Encoder-decoder shift-reduce syntactic parsing. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 105–114, Pisa, Italy. Association for Computational Linguistics.

Chao Lou, Songlin Yang, and Kewei Tu. 2022. Nested named entity recognition as latent lexicalized constituency parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6183–6198, Dublin, Ireland. Association for Computational Linguistics.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proc. Natl. Acad. Sci. USA*, 117(48):30046–30054.

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.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq Joty, and Xiaoli Li. 2020. Efficient constituency parsing by pointing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3284–3294, Online. Association for Computational Linguistics.

Jingcheng Niu, Wenjie Lu, Eric Corlett, and Gerald Penn. 2022. Using roark-hollingshead distance to probe BERT's syntactic competence. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 325–334, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

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, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 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.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser.

In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Sequence labeling parsing by learning across representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5350–5357, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 746–754, Prague, Czech Republic. Association for Computational Linguistics.

Yang Wei, Yuanbin Wu, and Man Lan. 2020. A span-based linearization for constituent trees. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3267–3277, Online. Association for Computational Linguistics.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics.

Xin Xin, Jinlong Li, and Zeqi Tan. 2021. N-ary constituent tree parsing with recursive semi-Markov model. 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 2631–2642, Online. Association for Computational Linguistics.

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*, pages 207–238.

Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

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.

Songlin Yang and Kewei Tu. 2023. Don't parse, choose spans! continuous and discontinuous constituency parsing via autoregressive span selection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8420–8433, Toronto, Canada. Association for Computational Linguistics.

Yu Zhang, Qingrong Xia, Shilin Zhou, Yong Jiang, Guohong Fu, and Min Zhang. 2022. Semantic role labeling as dependency parsing: Exploring latent tree structures inside arguments. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4212–4227, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and accurate neural CRF constituency parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4046–4053. ijcai.org.

Junru Zhou, Zuchao Li, and Hai Zhao. 2020. Parsing all: Syntax and semantics, dependencies and spans. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4438–4449, Online. Association for Computational Linguistics.

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.

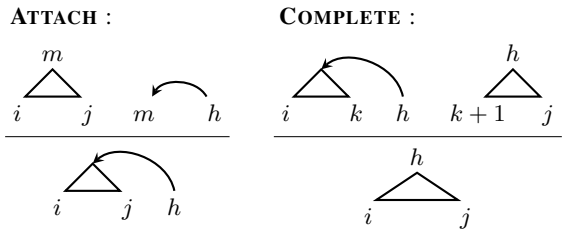


Figure 3: Deduction rules for Eisner-Satta algorithm (**ATTACH** and **COMPLETE**). We show only Left-rules, omitting the symmetric Right-rules as well as initial conditions for brevity.

A Method

A.1 Head Binarization

If the headwords are extracted with head-finding rules, the left binarization and right binarization strategies are no longer applicable. That is because the headword is not always the leftmost or rightmost child. In this case, we can use the head binarization to binarize the constituency tree.

The basic idea is to split the constituent into two parts with the headword in the middle. However, it is not enough to simply split the constituent into two parts. We need to ensure that the dependency structures in both parts are valid. Specifically, if node A is binarized as $A \rightarrow A_1 A_2$: (1) there exists a dependency relation between the headword of A_1 and A_2, (2) dependency arcs within both A_1 and A_2 can form single-rooted dependency subtrees[11].

A.2 Eisner-Satta Algorithm

The Eisner-Satta algorithm is a dynamic programming algorithm for finding the highest-scoring lexicalized tree. It iteratively combines smaller spans into larger ones to construct a complete lexicalized tree. It distinguishes between two types of spans: headed spans and hooked spans. A headed span (i, j, h) is a span (i, j) with a headword h, while a hooked span (i, j, h) is a span (i, j) with an unknown headword, which is governed by a word h outside the span. The deduction process is shown in Figure 3.

During training, we use an Inside-version of the Eisner-Satta algorithm to compute the partition function $\boldsymbol{Z}(\boldsymbol{w})$, by replacing the max operator with the log-sum-exp operator. For the proposed latent lexicalization method, we use a masked version of the algorithm to enumerate all valid lexicalized trees. As shown in Algorithm 1.

[11]This requires dependency trees to be projective.

Dataset	Language	#Train	#Dev	#Test
PTB	English	39,832	1,700	2,416
CTB5	Chinese	17,544	352	348
SPMRL	French	14,759	1,235	2,541
SPMRL	German	40,472	5,000	5,000
SPMRL	Korean	23,010	2,066	2,287
SPMRL	Hungarian	8,146	1,051	1,009
SPMRL	Basque	7,577	948	946
SPMRL	Polish	6,578	821	822

Table 10: Data statistics. We present the number of sentences in the training, development, and test sets.

Params	**BERT**
MLP Input Size	768/1024
MLP Layers	1
Con/Dep BiAffine Size	500
Label BiAffine Size	100
BERT Dropout	0.1
Other Dropout	0.33
Optimizer	AdamW
BERT Learning Rate	5×10^{-5}
Other Learning Rate	1×10^{-3}
Batch Size (tokens)	1000

Table 11: Parameters settings.

B Experiments

B.1 Implementation Details

The data statistics for our experiments are summarized in Table 10. We adopt the majority of hyperparameters from Zhang et al. (2020). A key modification in our approach is the removal of the cascaded LSTM layers traditionally placed above BERT. Instead, we directly fine-tune the BERT model. Specifically, we employ `bert-large-cased` for English, `bert-base-chinese` for Chinese, and `bert-base-multilingual-cased` for the SPMRL dataset. For PTB and CTB, we fine-tune BERT for 10 epochs, and for SPMRL, we fine-tune for 15 epochs for high-resource languages and 100 epochs for low-resource languages. The detailed hyperparameters are listed in Table 11. Experiments are conducted on a single NVIDIA V100 GPU. Results are averaged over three runs with different random seeds.

Algorithm 1 Eisner-Satta Algorithm

Input: span scores $s(i, j) \in \mathbb{R}^{n \times n}$
▷ *scores of crossing spans are masked to* $-\infty$ *(Section 3.2)* ◁
Input: arc scores $s(h \rightarrow m) \in \mathbb{R}^{(n+1) \times n}$
▷ *some arc scores are masked to* $-\infty$ *based on linguistic constraints (Section 3.3)* ◁
Define: $\alpha, \beta \in \mathbb{R}^{n \times n \times (n+1)}$
Initialize: $\alpha_{:,:,:} = 0, \beta_{:,:,:} = 0$
for $w = 1, \ldots, n$ **do**
  for $i = 1, \ldots, n - w$ **do**
    $j = i + w$
    for $h = i, \ldots, j$ **do**
      $\alpha_{i,j,h} = s(i, j) + \max_{i \leq k \leq j}(\alpha_{i,k,h} + \beta_{k+1,j,h}; \beta_{i,k,h} + \alpha_{k+1,j,h})$
      ▷ *Inside version:* $\log \sum_{i \leq k \leq j}[\exp(\alpha_{i,k,h} + \beta_{k+1,j,h}) + \exp(\beta_{i,k,h} + \alpha_{k+1,j,h})]$ ◁
    end for
    for $h = 0, \ldots, n$ **do**
      $\beta_{i,j,h} = \max_{i \leq m \leq j}(\alpha_{i,j,m} + s(h \rightarrow m))$
      ▷ *Inside version:* $\log \sum_{i \leq m \leq j} \exp(\alpha_{i,j,m} + s(h \rightarrow m))$ ◁
    end for
  end for
end for
return $\beta_{1,n,0}$

Model	English	Chinese	French	German	Korean	Hungarian	Basque	Polish
Con	95.79	91.93	87.50	90.45	89.45	94.71	91.26	95.97
Lex (head)	95.76	91.86	87.63	90.46	89.48	94.38	91.12	95.68
Lex (left)	95.88	91.94	87.52	90.42	89.36	94.48	91.34	96.00
Lex (right)	95.74	91.96	87.39	**90.57**	89.49	**94.80**	**91.52**	95.87
Lex (latent) (n=4)	95.87	92.16	**87.75**	90.03	89.43	94.78	91.07	95.89
Lex (latent) (n=8)	**95.97**	**92.20**	87.55	90.08	**89.50**	94.44	91.34	**96.17**

Table 12: Max-margin training results on PTB (English), CTB5 (Chinese) and SPMRL (covering French, German, Korean, Hungarian, Basque and Polish). For latent lexicalization, we report the results with different numbers of sampled trees (n).

Model	Sent/Sec.
Zhou and Zhao (2019)	158.7
Chen et al. (2021)	**411.2**
Gu et al. (2024)	305.8
Con	254.4
Lex (latent)	249.7

Table 13: Decoding speed comparison on PTB test set.

B.2 Max-Margin Training

In table 12, we adopt a max-margin training objective to optimize the benchmark models.

$$\mathcal{L} = \max(0, \Delta(\hat{\boldsymbol{t}}, \boldsymbol{t}^*) + s(\hat{\boldsymbol{t}}) - s(\boldsymbol{t}^*))$$

For latent lexicalization, a possible approach to apply max-margin loss is to enumerate all valid lexicalized trees and compute the margin loss for each tree.

$$\mathcal{L} = \max(0, s(\hat{\boldsymbol{t}}) - \frac{\sum_{\boldsymbol{t}^* \in \mathcal{T}} s(\boldsymbol{t}^*)}{|\mathcal{T}|} + \frac{\Delta(\hat{\boldsymbol{t}}, \mathcal{T})}{|\mathcal{T}|})$$

However, these exists two key challenges:

1. The number of trees in the forest (i.e. $|\mathcal{T}|$) grows exponentially, making it difficult to determine the exact count.

Phrase	Head	Count	Ratio (%)	Headword
NP	NN	5823	31.3	"$", "%",
	NP	4443	23.9	it, company,
	NNS	3273	17.6	he, market,
	NNP	2939	15.8	year, Corp.
VP	VP	2745	31.4	said, have,
	VBD	1416	16.2	expected, says,
	VB	1318	15.1	is, buy,
	VBN	900	10.3	take, had
S	VP	5359	94.6	said, is,
	S	195	3.4	have, says
PP	IN	4971	90.5	of, in,
	TO	394	7.2	for, to,
	VBG	64	1.2	on, by,
	PP	29	0.5	with, from
SBAR	S	1758	97.8	have, is,
	SBAR	27	1.5	had, expects
ADVP	RB	1065	87.8	also, down,
	RBR	50	4.1	ago, still,
	IN	47	3.9	up, however,
	ADVP	21	1.7	back, well
ADJP	JJ	598	67.0	"$", million,
	ADJP	61	6.8	likely, higher
QP	$	249	50.8	million, "%"
	CD	196	40.0	1/2, billion
WHNP	WDT	270	62.9	that, which,
	WP	106	24.7	who, what

Table 14: Most frequent heads for each constituent label on SD.

2. Summing over all tree scores requires a dynamic programming algorithm (Eisner-Satta in our case). However, for numerical stability, it computes $\log \sum_{\boldsymbol{t}^* \in \mathcal{T}} \exp(s(\boldsymbol{t}^*))$ rather than $\sum_{\boldsymbol{t}^* \in \mathcal{T}} s(\boldsymbol{t}^*)$, which is necessary for max-margin loss.

A potential solution involves sampling a subset of trees from the forest, and we present the results using 4 and 8 sampled trees.

B.3 Parsing Speed Comparison

Parsing efficiency is critical for practical applications. Table 13 compares the parsing speed of our model with **Con** on PTB. Although our model has a higher theoretical complexity ($O(n^4)$ vs. $O(n^3)$), both achieve comparable speeds in practice due to the GPU-optimized batch processing. This demonstrates that the added computational cost of latent lexicalization is negligible in real-world scenarios.

Head ↷ Dependent	Count	Ratio (%)
NN ↷ DT	3554	6.27
NNP ↷ NNP	2489	4.39
IN ↷ NN	2148	3.79
NN ↷ JJ	1706	3.01
NN ↷ IN	1667	2.94
NN ↷ NN	1304	2.30
IN ↷ NNS	1300	2.29
NNS ↷ JJ	1211	2.14
IN ↷ NNP	905	1.60
ROOT ↷ VBD	879	1.55

Table 15: Top ten dependency relationships on SD.

Model	Arc Distance	Root Distance
SD	3.4	8.9
Ours	4.1	9.6

Table 16: Average dependency arc distance, with the root arc listed separately.

B.4 Head Selection from Stanford Dependencies

In this section, we provide the gold-standard head selection by the Stanford Dependencies (SD). The pharse head selection is shown in Table 14, and the dependency relationships are shown in Table 15.

B.5 Arc Distance

We analyze the average distance between heads and their dependents, as well as the distance of the root to the main verb. Table 16 shows that predicted dependencies tend to span longer distances than gold-standard ones. While the dependency structures are constrained by the constituency structures, this results indicate that the inferred dependencies are not limited to local relationships.