

Syntactic Prediction through Reinforcement Learning

Jinman Zhao¹, Yining Wang^{*1}, Jiahe Liu³, Xueyan Zhang^{*2},
Linbo Cao², Jiakang Huang⁴, Yiren Zhao¹, Renyi Cai¹, Yitian Ding⁵, Gerald Penn¹

¹University of Toronto, ²University of Waterloo,

³Technical University of Denmark, ⁴University of British Columbia, ⁵McGill University
jzhao@cs.toronto.edu

Abstract

Syntactic prediction requires models to generate outputs that are globally consistent and satisfy hierarchical structural constraints. Although large language models have been applied to such tasks, existing approaches still rely mainly on prompting or supervised fine-tuning, and their performance generally remains below that of specialized neural syntactic models. Our experiments show that supervised fine-tuning reaches a plateau, and additional supervised training does not further improve prediction quality. We therefore propose a two-stage post-training framework for syntactic prediction. The first cold-start stage adapt the model to a unified syntax generation format, and then applies reinforcement learning on a disjoint data subset to directly optimize structure-level objectives. Experiments on constituency parsing and CCG supertagging show that Syntax-RL consistently outperforms supervised LLM baselines and achieves state-of-the-art results. Further analyses suggest that its gains come primarily from the reinforcement learning stage. These findings identify reinforcement learning as an effective post-training mechanism for improving hierarchical syntactic prediction in large language models.

1 Introduction

Syntactic prediction has long served as a fundamental testbed for structured language understanding. Tasks such as constituency parsing (Kitaev et al., 2019; Mrini et al., 2020; Tian et al., 2020b) and supertagging (Clark et al., 2018; Tian et al., 2020a; Yamaki et al., 2023) require models not only to make locally plausible decisions, but also to produce outputs that are globally consistent with hierarchical syntactic structure. Success in these tasks therefore depends on structure-level correctness, reflecting the broader importance of modeling dependencies within complex output spaces (Kou

et al., 2024): outputs must satisfy non-local constraints, maintain sentence-level consistency, and reflect coherent syntactic organization. With the rise of large language models (LLMs) (OpenAI, 2024), some studies (Bai et al., 2025; Tian et al., 2024) have attempted to tackle these tasks using LLMs via in-context learning or fine-tuning. However, their performance still typically falls short of that of traditional encoder-based models specifically designed for syntactic prediction. This raises a central question: *what kind of post-training signal is needed to unlock the syntactic competence of LLMs?*

A natural approach is supervised fine-tuning (SFT). SFT trains the model to imitate gold structured outputs under a token-level objective, although task-specific applications often require evaluating model outputs against higher-level domain objectives (Shi et al., 2026). SFT does substantially improve LLM performance on syntactic tasks, but our results suggest that these gains soon reach a clear plateau. In fact, prior works (Bai et al., 2025; Zhao and Penn, 2024) have already observed that even LLMs fine-tuned for constituency parsing still fail to surpass traditional task-specific syntactic models. For example, Bai et al. (2025) reported that for constituency parsing, even a 70B-scale model remained inferior to a much smaller XLNet-based parser. Our experiments reveal the same pattern on both constituency parsing and supertagging. Neither stronger supervised training on the full training set nor continued SFT on the exact same second-stage subset used for Reinforcement Learning (RL) can match the performance of models further optimized with second-stage RL. We refer to this phenomenon as a *supervised plateau*. These results suggest that additional supervised training alone is insufficient to fully optimize hierarchical syntactic prediction.

To overcome this limitation, we propose **Syntax-RL**, a syntax-oriented post-training framework that

^{*}Corresponding Author

combines cold-start SFT with a second-stage RL procedure. The key idea is that while SFT equips the model with basic task adaptation and output-formatting abilities, further improvements in hierarchical syntactic prediction require an optimization signal that directly targets task-level outcomes. Specifically, we first perform cold-start SFT on 80% of the training data to adapt the model to a unified syntax generation format. We then apply GRPO-based RL on the remaining disjoint 20% as a second-stage post-training step, directly optimizing the structure-level objectives required by syntactic tasks. To ensure that the observed gains come from the optimization objective rather than differences in data exposure or staged training alone, we compare Syntax-RL not only with full-data SFT but also with a continued-SFT baseline that uses exactly the same second-stage subset as RL. Under this controlled setup, only RL post-training consistently overcomes the supervised plateau, yields a clear second-stage gain on both constituency parsing and supertagging, and achieves state-of-the-art performance. In summary, our main contributions are as follows:

1. We propose **Syntax-RL**¹, a syntax-oriented post-training framework for syntactic prediction that combines cold-start SFT with a second-stage GRPO-based reinforcement learning procedure to directly optimize structure-level outcomes.
2. We demonstrate the effectiveness of Syntax-RL on two representative syntactic prediction tasks, constituency parsing and supertagging, achieving state-of-the-art performance on both tasks.
3. Through a series of controlled analyses, we show that the gains of Syntax-RL are driven by the second-stage RL objective itself, rather than by prolonged supervised training, repeated exposure to training data, or intermediate thought-style supervision.

2 Related Work

2.1 Neural Syntactic Prediction

Supertagging. Supertagging is a long-standing syntactic prediction task with roots in lexicalized grammar formalisms such as LTAG (Bangalore and Joshi, 1999). For CCG, early approaches were

dominated by feature-based discriminative models (Clark, 2002; Clark and Curran, 2007), and later work shifted to neural sequence models that predict supertags directly from sentence context (Xu et al., 2015; Lewis et al., 2016; Bhargava and Penn, 2020; Clark et al., 2018). Recent progress has come from Transformer-based methods and architectures with richer structural bias, including graph-based and recursive-compositional approaches (Tian et al., 2020a; Yamaki et al., 2023; Kogkalidis and Moortgat, 2023; Prange et al., 2021).

Constituency Parsing. Modern neural constituency parsing has developed along several major modeling paradigms. One dominant line is chart-based parsing. These methods score candidate spans or labeled spans and recover a tree with chart-based decoding (Kitaev and Klein, 2018; Zhang et al., 2020; Xin et al., 2021). Another major line is transition-based parsing. These models construct parse trees through sequences of incremental parsing actions (Cross and Huang, 2016; Liu and Zhang, 2017). Beyond these two standard paradigms, some studies reformulate constituency parsing as linearized tree generation (Wei et al., 2020; Amini and Cotterell, 2022). Others represent tree construction with pointer sequences or related structured decoding processes (Nguyen et al., 2020; Yang and Tu, 2022). More recently, combinatory parsing has emerged as another promising direction. These methods build larger constituents by merging shorter spans in a layer-wise manner (Chen et al., 2021; Chen and Komachi, 2023; Tian and Song, 2022). Other work has explored alternative inductive biases, including head-driven formulations, stronger pretrained backbones, and ensemble-style improvements (Fried et al., 2019; Zhou and Zhao, 2019; Mrini et al., 2020; He and Choi, 2023; Shayegh et al., 2024).

2.2 LLMs for Structured and Syntactic Prediction

Large language models have been increasingly explored for structured prediction tasks, including named entity recognition (Wang et al., 2023; Xie et al., 2023; Chen et al., 2023) and structured information extraction (Li et al., 2024; Liu et al., 2026), discourse-level structured representation (Zhang et al., 2025), structured query generation (Mao et al., 2026) and multi-label prediction (Kou et al., 2025). Prior probing studies have also examined how syntactic knowledge is encoded within pre-

¹https://github.com/zhaojinm/syntax_RL

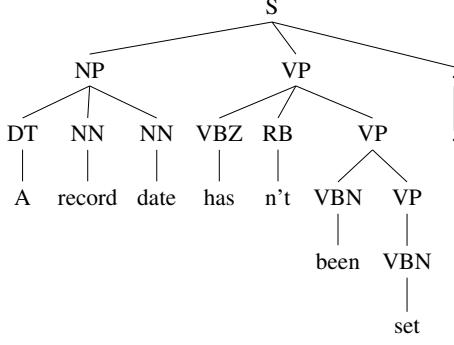


Figure 1: Example constituency tree for “A record date has n’t been set.”

trained language models (Niu et al., 2022). In syntactic prediction, prior work has studied constituency parsing and supertagging with LLMs under prompting, supervised fine-tuning, and data augmentation settings (Bai et al., 2025; Tian et al., 2024; Guo et al., 2025; Zhao and Penn, 2024). Recent studies have also extended LLM-based syntactic prediction to dependency parsing (Matsuda et al., 2025; Ginn and Palmer, 2025). These studies show that LLMs can serve as general structured predictors, but existing approaches mainly rely on prompting, supervised fine-tuning, knowledge distillation, or data augmentation (Song et al., 2026).

2.3 Reinforcement Learning

RL is an important tool for LLM post-training (Williams, 1992). PPO (Schulman et al., 2017) improved training stability and became a standard algorithm in RLHF. More recently, DPO (Rafailov et al., 2023) introduced a simpler alternative based on preference optimization. GRPO (Shao et al., 2024) later emerged as a practical variant of PPO for large-scale LLM post-training. Follow-up methods (Fan et al., 2026; Zhang et al., 2026; Kou et al., 2026; Zhou et al., 2026; Guo et al., 2026) further improved AI tasks performance via reinforcement learning.

3 Method

We study *syntactic prediction* in the form of two representative tasks including constituency parsing and supertagging. In both cases, the model takes a sentence as input and generates a structured syntactic output. Figure 2 gives an overview of the framework.

3.1 Task Formulation

Let $x = (w_1, \dots, w_n)$ denote an input sentence of length n and each w_i is a token.

Constituency Parsing. In constituency parsing, the target output y^{parse} is a phrase-structure tree over x . Since the model is implemented as a language model, we linearize the tree into a bracketed string and treat parsing as sequence generation:

$$y^{\text{parse}} = (t_1, \dots, t_m), \quad (1)$$

Following Bai et al. (2025), we use bracketed-based tree representations as target outputs. For example, given the sentence “A record date has n’t been set.”, the target output shown in Figure 1 is:

```

(S (NP (DT A) (NN record) (NN date)) (VP
  (VBZ has) (RB n't) (VP (VBN been) (VP
    (VBN set))))) (. .)

```

Categorial Grammar Supertagging. In categorial grammar supertagging, the target output assigns each token w_i a lexical category c_i from a predefined category inventory $\mathcal{C}$. The desired output can therefore be written as

$$y^{\text{stag}} = (c_1, \dots, c_n), \quad c_i \in \mathcal{C}. \quad (2)$$

To align with the text-generation interface of large language models, we serialize the category assignment as a textual sequence

$$\tilde{y}^{\text{stag}} = \text{Serialize}((w_1, c_1), \dots, (w_n, c_n)), \quad (3)$$

Following Zhao and Penn (2024), we represent each token together with its categorial grammar supertag in a unified textual format. For example, the target output is:

```

{NP[nb]/N A} {N/N record} {N date}
{(S[dc1]\NP)/(S[pt]\NP) has} ... { . .}

```

Under this formulation, both tasks can be unified as structured sequence generation problems.

3.2 Training Pipeline

3.2.1 Stage 1: Supervised Fine-Tuning for Cold Start

The first stage of Syntax-RL is cold-start SFT. Its goal is to adapt the base LLM to the unified syntactic generation tasks described above, so that the model can reliably produce structured syntactic outputs conditioned on an input sentence. Rather than directly applying RL to an unadapted base model, this stage first establishes basic task competence,

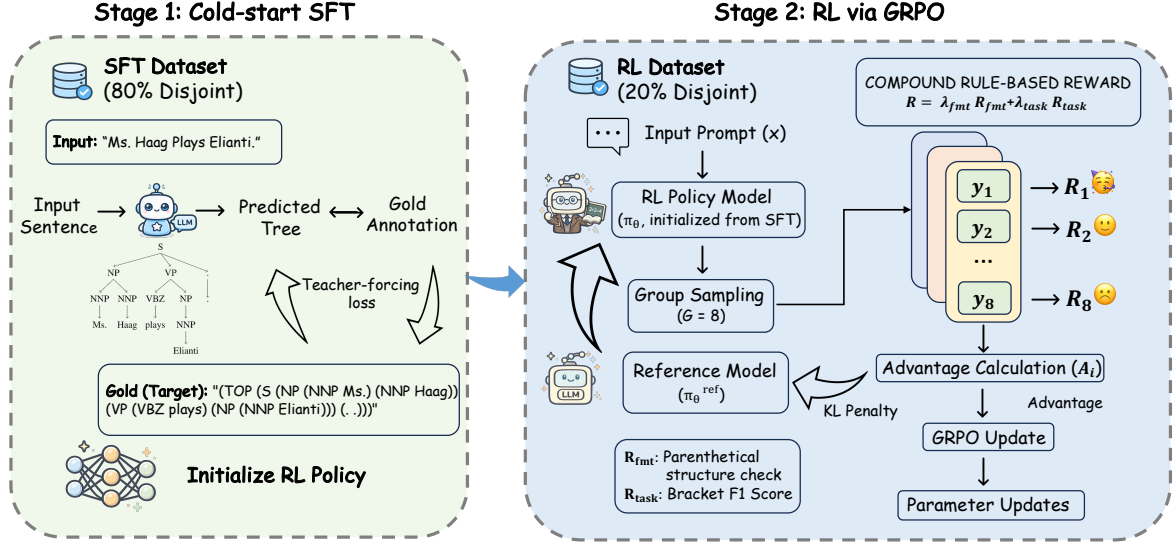


Figure 2: Overview of our training pipeline.

including following task instructions, adhering to the required output format, and making initial local syntactic decisions.

We use $\mathcal{D}_{\text{SFT}}$ to denote SFT data. The model is optimized with the standard teacher-forcing negative log-likelihood objective:

$$\mathcal{L}_{\text{SFT}} = - \sum_{(x,y) \in \mathcal{D}_{\text{SFT}}} \sum_{t=1}^{|y|} \log p_{\theta}(y_t | x, y_{<t}), \quad (4)$$

where x denotes the input sentence and y denotes the corresponding target syntactic output as described in Section 3.1.

This stage serves two purposes. First, it transforms a general-purpose LLM into a task-adapted generator for syntactic prediction. Second, it provides a stable initialization for the subsequent RL stage, avoiding policy optimization from an unadapted base model.

3.2.2 Stage 2: Reinforcement Learning for Policy Optimization

After cold-start SFT, we further optimize the model with RL. This stage aims to move beyond the imitative behavior learned under teacher forcing and directly optimize task-level syntactic outcomes. Starting from the SFT checkpoint, the model generates complete structured outputs for each input sentence, and receives scalar rewards based on both output validity and task correctness.

Policy Optimization. Given an input sentence x , we sample a group of candidate outputs

$$\mathcal{Y}(x) = \{y_1, \dots, y_G\}, \quad y_i \sim \pi_{\theta}(\cdot | x), \quad (5)$$

where π_{θ} denotes the current policy and G is the group size. Each sampled output y_i is assigned a scalar reward $r_i = R(x, y_i)$. GRPO computes a group-relative normalized advantage:

$$A_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r}) + \epsilon}, \quad \mathbf{r} = (r_1, \dots, r_G), \quad (6)$$

where ϵ is a small constant for numerical stability. We then optimize the policy with the objective

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}_{x \sim \mathcal{D}_{\text{RL}}, y_i \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \times \sum_{i=1}^G \min(\rho_i A_i, \text{clip}(\rho_i, 1 - \epsilon, 1 + \epsilon) A_i) \right], \quad (7)$$

where

$$\rho_i = \frac{\pi_{\theta}(y_i | x)}{\pi_{\theta_{\text{old}}}(y_i | x)}, \quad (8)$$

and ϵ is the clipping coefficient. GRPO uses group-relative rewards to estimate advantages and does not require a separately trained value model.

Reward Design. We define a composite reward

$$R(x, y) = \lambda_{\text{fmt}} R_{\text{fmt}}(x, y) + \lambda_{\text{task}} R_{\text{task}}(x, y), \quad (9)$$

The format reward encourages well-formed syntactic outputs:

$$R_{\text{fmt}}(x, y) = \begin{cases} 1, & \text{if } y \text{ is correctly formatted,} \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

For constituency parsing, this term checks whether the output is a valid bracketed tree string. For supertagging, it checks whether the output is a valid serialized token-tag sequence with the correct token alignment.

The task reward is defined differently for the two tasks. For constituency parsing, we use sentence-level parsing quality measured by bracket F1:

$$R_{\text{task}}^{\text{parse}}(x, y) = \text{F1}(y, y^*), \quad (11)$$

where y^* denotes the gold tree. For supertagging, we use token-level accuracy:

$$R_{\text{task}}^{\text{stag}}(x, y) = \text{Acc}(y, y^*), \quad (12)$$

where y^* is the gold supertag sequence.

3.2.3 Data Preparation

An important design choice in Syntax-RL is the separation between the SFT data and the RL data. In preliminary experiments, we found that simply applying RL on the same training set used for supervised fine-tuning did not yield the best performance. In particular, using 100% of the training data for SFT followed by RL on the same 100% subset was consistently worse than a disjoint two-stage setup using 80% of the data for SFT and the remaining 20% for RL.

Based on this observation, we partition each training treebank into two non-overlapping subsets:

$$\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{SFT}} \cup \mathcal{D}_{\text{RL}}, \quad \mathcal{D}_{\text{SFT}} \cap \mathcal{D}_{\text{RL}} = \emptyset, \quad (13)$$

where

$$|\mathcal{D}_{\text{SFT}}| = 0.8 |\mathcal{D}_{\text{train}}|, \quad |\mathcal{D}_{\text{RL}}| = 0.2 |\mathcal{D}_{\text{train}}|. \quad (14)$$

The first subset is used for cold-start SFT, and the second for second-stage optimization. This disjoint split prevents RL from merely re-optimizing examples already fitted under teacher forcing, while enabling a clean comparison with continued SFT under the same second-stage data exposure. SFT provides formatting and basic syntactic mapping through token-level imitation, whereas RL on fresh examples offers a cleaner structure-level corrective signal. The 80/20 split is an empirical choice for PTB/CCG, not a universal rule.

4 Experimental Setup

4.1 Datasets

Constituency Parsing. For in-domain evaluation, we use the Penn Treebank (PTB) (Marcus

et al., 1993), containing about 40k instances from the news domain and serves as a standard benchmark in prior work. We follow the conventional split and use sections 02–21 for training, section 22 for validation, and section 23 for testing.

Supertagging. For evaluation, we use CCGbank (Hockenmaier and Steedman, 2007)² and Rebank (Honnibal et al., 2010), standard benchmark resources for CCG parsing and grammar-related analysis. CCGbank provides the main parsed corpus derived from the Penn Treebank and serves as the primary reference for large-scale empirical evaluation. Rebank offers an alternative reannotation that is useful for examining structural variation and testing the robustness of grammar induction and parsing behavior under different annotation choices. Detailed dataset statistics for CCGbank and CCGrebank are provided in Appendix A.2.

4.2 Implementation Details

We use LLAMA-FACTORY (Zheng et al., 2024) for the SFT stage and VERL (Sheng et al., 2025) for the RL stage. The SFT stage is run on 4xH200 GPUs. The RL stage is run on 8xH200 GPUs. Full hyperparameter details for both the stages are provided in Appendix B. Also, generated outputs are not always perfectly aligned with the input tokens. Following Zhao and Penn (2024), we use the longest common subsequence (LCS) to align the generated and gold token sequences before computing token-level accuracy.

4.3 Baselines

Constituency Parsing. For constituency parsing, we include representative neural parser baselines, including Kitaev et al. (2019), Head-Driven Phrase Structure Grammar Parsing on Penn Treebank (HPSG) (Zhou and Zhao, 2019), Attach-Juxtapose (Yang and Deng, 2020), Span Attention parser (Tian et al., 2020b), and LAL-Parser (Mrini et al., 2020), together with recent LLM-based baselines from Bai et al. (2025).

Supertagging. We compare our method against a range of representative baselines on both CCG supertagging and constituency parsing benchmarks. For CCG supertagging, the baselines include Cross-View Training (CVT) (Clark et al., 2018), Bhargava and Penn (2020), Liu et al. (2021), Tree-Structured Decoding. (Prange et al.,

²<https://catalog.ldc.upenn.edu/LDC2005T13>

2021), Attentive Graph Convolutional Networks (AGCN) (Tian et al., 2020a), Geometry-Aware Supertagging (Kogkalidis and Moortgat, 2023), Holographic CCG Parsing (Yamaki et al., 2023), and LLM-supertagger (Zhao and Penn, 2024). Detailed descriptions of all baseline methods are provided in Appendix C.

5 Results

We first report the main results and then analyze the effect of training design choices in Syntax-RL.

5.1 Main Results

Constituency Parsing. Table 1 presents constituency parsing results on PTB and on the subset of PTB sentences with length smaller than 40 ($PTB < 40$). Syntax-RL consistently outperforms both task-specific neural parsers and supervised LLM baselines. On the full PTB benchmark, the best prior neural parser reaches 96.38 F1, while the strongest supervised LLM baseline reaches 96.21 F1. In contrast, Syntax-RL with Qwen2.5-32B achieves 96.51 F1, improving over the strongest supervised baseline by 0.30 points and over the best prior neural parser by 0.13 points. On $PTB < 40$, Syntax-RL further achieves 96.59 F1, again establishing the best result among all compared systems.

Another important pattern is that the gains from Syntax-RL are consistent across model sizes. Relative to their supervised counterparts, the 7B, 14B, and 32B versions of Syntax-RL all obtain clear improvements on both PTB and $PTB < 40$. This shows that the benefit of the second-stage RL optimization is not confined to a single backbone or scale regime. Instead, it provides a general performance gain on hierarchical syntactic prediction

Supertagging. Table 2 reports the main results on CCGbank and CCGrebank supertagging. Syntax-RL achieves the best performance on both benchmarks, reaching 96.70 accuracy on CCGbank and 95.55 accuracy on CCGrebank. On CCGbank, The results show that Syntax-RL remains effective across both the standard CCGbank benchmark and the more challenging CCGrebank setting. This suggests that the benefit of the second-stage RL optimization is not limited to a single dataset or evaluation condition. Instead, the same two-stage post-training pipeline consistently improves supertagging performance in different settings. These results support our claim that RL provides a useful

structure-level optimization signal for fine-grained syntactic prediction.

Generalization Across Languages. To verify that the gains of Syntax-RL are not specific to English, we conduct additional experiments. We further evaluate on the Chinese Penn Treebank (CTB). As shown in Table 3, Syntax-RL with Qwen2.5-32B improves F1 from 92.35 to 92.68, and surpasses representative neural parsers.

5.2 Further Analysis

Unless otherwise stated, all experiments in this section are conducted on constituency parsing using Qwen2.5-32B.

Effect of SFT-RL Data Allocation. Table 4 shows that data allocation across the two stages has a clear impact on final performance. Compared with $20k\ SFT + 20k\ RL$, the $32k\ SFT + 8k\ RL$ setting improves F1 from 96.25 to 96.52, even though both settings use the same total amount of training data. This suggests that a sufficiently strong SFT stage is more important than allocating a larger portion of the 40k training set to RL.

The comparison between $32k\ SFT + 40k\ RL$ and $32k\ SFT + 8k\ RL$ further clarifies the role of data overlap. In both settings, the first-stage SFT model is initialized from the same 32k subset, and the number of second-stage RL update steps is kept the same. The only difference is the data used in the RL stage. $32k\ SFT + 8k\ RL$ uses only the disjoint remaining 8k examples, whereas $32k\ SFT + 40k\ RL$ applies RL on the full 40k training set and therefore reuses the 32k SFT data. Despite having access to the full training set in the RL stage, $32k\ SFT + 40k\ RL$ still performs worse than $32k\ SFT + 8k\ RL$. This result suggests that the second-stage RL signal is most effective when it is applied to a disjoint subset, rather than repeatedly optimizing on examples already used in the supervised stage.

RL Beyond Continued Supervised Fine-tuning. We compare three settings under the same training pool: $40k\ SFT$, $32k\ SFT + 8k\ SFT$, and $32k\ SFT + 8k\ RL$. The first setting applies SFT only on the full 40k training set. The second setting first performs SFT on 32k examples and then continues SFT on the remaining 8k examples. The third setting uses the same 32k first-stage initialization and the same 8k second-stage subset, but replaces the second-stage supervised objective with RL.

Category	Models	PTB			PTB < 40		
		LR	LP	F1	LR	LP	F1
Neural Parser	Kitaev et al. (2019) (BERT)	95.15	95.96	95.55	95.38	96.15	95.76
	Zhou and Zhao (2019) (XLNet)	96.21	96.46	96.33	–	–	–
	Yang and Deng (2020) (XLNet)	96.55	96.13	96.34	–	–	–
	Tian et al. (2020b) (ZEN/XLNet)	96.07	<u>96.64</u>	96.36	–	–	–
	Mrini et al. (2020) (XLNet-large-cased)	96.24	96.53	<u>96.38</u>	–	–	–
LLM (Bai et al., 2025)	Llama3.1-8B	95.74	96.05	95.89	95.93	96.27	96.10
	Qwen2.5-7B	95.78	96.20	95.99	95.97	96.30	96.13
	Qwen2.5-14B	96.02	96.29	96.15	96.05	96.37	96.21
	Qwen2.5-32B	96.07	96.35	96.21	96.15	96.45	96.30
Syntax-RL	Qwen2.5-7B	95.93	96.36	96.14	96.10	96.53	96.31
	Qwen2.5-14B	96.17	96.56	96.36	96.26	96.64	96.45
	Qwen2.5-32B	<u>96.25</u>	96.79	96.52	96.30	96.88	96.59

Table 1: Constituency parsing performance on PTB and the PTB subset with sentence length smaller than 40 (PTB < 40), comparing task-specific neural parsers, supervised LLM baselines, and Syntax-RL.

Model	Acc
Clark et al. (2018)	96.10
Bhargava and Penn (2020)	96.00
Liu et al. (2021)	96.05
Prange et al. (2021)	96.22
Tian et al. (2020a)	96.25
Kogkalidis and Moortgat (2023)	96.29
Yamaki et al. (2023)	96.60
Zhao and Penn (2024)	96.64
Syntax-RL (Qwen2.5-32B)	96.70

(a) CCGbank supertagging accuracy comparison.

Model	Acc
Prange et al. (2021)	94.83
Kogkalidis and Moortgat (2023)	95.07
Yamaki et al. (2023)	95.48
Zhao and Penn (2024)	95.50
Syntax-RL (Qwen2.5-32B)	95.55

(b) CCGbank supertagging accuracy comparison.

Table 2: Supertagging results on two Categorical Grammar benchmarks.

Table 5 shows that continued SFT does not improve performance. The *32k SFT + 8k SFT* setting reaches 96.00 F1. This is lower than the 96.21 F1 achieved by full-data SFT. Additional supervised updates on the second-stage subset therefore do not move the model beyond the first-stage plateau. Replacing the second-stage supervised objective with RL leads to a large gain. The *32k SFT + 8k RL* set-

Model	Backbone	F1
Kitaev et al. (2019)	BERT	91.75
Zhou and Zhao (2019)	BERT	92.18
Tian et al. (2020b)	ZEN/XLNet	92.24
Bai et al. (2025)	Qwen2.5-32B	92.35
Syntax-RL	Qwen2.5-32B	92.68

Table 3: Chinese constituency parsing results on CTB.

SFT	RL	LR	LP	F1
20k	20k	95.96	96.53	96.25
32k	40k	96.26	96.71	96.48
32k	8k	96.25	96.79	96.52

Table 4: Effect of data allocation across the two stages.

ting reaches 96.52 F1. The first-stage checkpoint is the same. The second-stage data are also the same. The only difference is the optimization objective. This comparison shows that the gain of Syntax-RL does not come from extra training alone. It comes from the RL stage itself.

Effect of Intermediate Thought Formats. We also study whether adding intermediate thought traces to the supervision format improves constituency parsing. To this end, we compare three settings: direct answer-only supervision, span-based thought traces followed by the final parse, and coarse tree-sketch thought traces followed by the final parse. The span-based format lists intermediate spans together with coarse constituent

Setting	LR	LP	F1
40k SFT	96.07	96.35	96.21
32k SFT + 8k SFT	95.88	96.12	96.00
32k SFT + 8k RL	96.25	96.79	96.52

Table 5: Comparison of full-data SFT, continued SFT, and second-stage RL.

Supervision Format	LR	LP	F1
Span-based thought + answer	95.65	96.30	95.97
Tree-sketch thought + answer	96.10	96.52	96.31
Answer only	96.25	96.79	96.52

Table 6: Effect of intermediate thought formats on constituency parsing.

labels. The tree-sketch format provides a coarse bracketed sketch before the final tree. As shown in Table 6, both thought-augmented formats underperform the simplest answer-only supervision. Full examples of the two thought formats are provided in Appendix D. In contrast to prior findings that chain-of-thought guidance and longer intermediate reasoning can benefit other structured prediction and general reasoning tasks (Li, 2024; Jin et al., 2024, 2025; Cai et al., 2025), explicit intermediate reasoning provides no benefit for syntactic prediction in our setting. Constituency parsing already requires the model to produce a fully structured output, and additional thought sequences may therefore introduce redundant supervision or distract from the final structural target.

Training Dynamics of the RL Stage. Figure 3 shows that the second-stage RL optimization consistently pushes the model beyond the performance plateau reached by SFT. Starting from the same 32k first-stage checkpoint, RL steadily improves PTB F1 from 96.02 to 96.52. The gain is larger in the early phase and gradually tapers off in later steps, indicating stable optimization with diminishing returns near convergence. In contrast, continued SFT provides only marginal benefit: even when allowed to continue on the full 40k supervised training set, it reaches a best F1 of only 96.23 and quickly saturates. This saturation echoes broader findings that continued likelihood-based optimization does not necessarily yield corresponding improvements in generation quality (Meng et al., 2026). Importantly, the RL stage uses only 8k second-stage examples, yet still outperforms continued SFT by a large margin. Moreover, continued SFT remains clearly be-

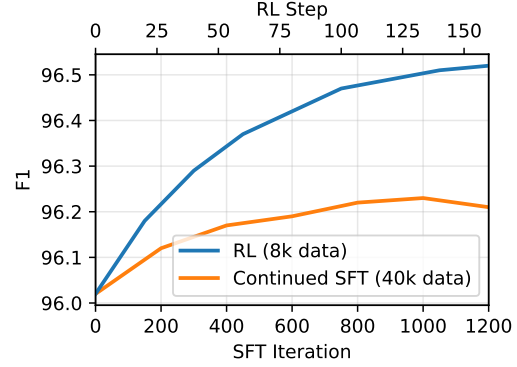


Figure 3: Training curve of the second-stage RL optimization on constituency parsing. The model steadily improves from 96.02 to 96.52 F1 as RL progresses.

Type	Category	Recall Imp.
Attachment-sensitive	Coordination	+4.78
	SBAR	+4.51
	Long NP	+3.18
	PP	+2.88
Frequent label	VP	+3.44
	S	+3.14
	NP	+2.71

Table 7: Recall improvement of Syntax-RL over the SFT baseline on gold constituents, grouped by category.

low the performance level of strong neural parsers in Table 1, and RL pushes the model to state-of-the-art performance.

Quantitative Structural Analysis. we conduct a full-test-set structural analysis comparing the SFT baseline (Bai et al., 2025) and Syntax-RL against the gold trees, using EVALB-style constituent extraction to measure which gold constituents missed by the SFT model are recovered by Syntax-RL. As shown in Table 7, Syntax-RL improves recall on attachment-sensitive categories, including coordination, SBAR, long NP, and PP, and also shows consistent gains on frequent phrase categories such as VP, S, and NP. This provides quantitative evidence that the RL stage improves structural prediction. A detailed case study is listed in Appendix E.

Comparison with a Specialized Parser. We compare Syntax-RL with a strong traditional neural parser (Kitaev et al., 2019). As shown in Table 8, Syntax-RL has slightly higher recall on shorter and medium spans, while benepar_en3_large remains stronger on the longest spans (>20). At the label level, Syntax-RL shows clear advantages

Span Len.	Recall Diff.	Label	Gold	Diff.
1–2	+0.91	PRN	142	+51.41
3–5	+0.89	ADJP	661	+12.25
6–10	+0.68	ADVP	286	−3.15
11–20	+0.67	WHNP	38	−2.63
>20	−0.51	FRAG	48	−2.08

(a) By span length.

(b) By label.

Table 8: Fine-grained recall comparison between Syntax-RL and benepar_en3_large (Kitaev et al., 2019). Positive values mean Syntax-RL has higher recall.

on PRN and ADJP, whereas benepar_en3_large is stronger on ADVP, WHNP, and FRAG. These category-level and span-level differences indicate that Syntax-RL differs structurally from a specialized parser, rather than benefiting uniformly from model scale.

6 Conclusions

We introduce a two-stage post-training framework for syntactic prediction that combines cold-start SFT with second-stage GRPO-based reinforcement learning. Our results show that stronger supervised training alone quickly reaches a supervised plateau on both constituency parsing and supertagging. RL post-training, in contrast, consistently provides additional gains and achieves state-of-the-art performance. Further analyses suggest that these gains come from structure-level optimization on final syntactic outputs, rather than from extra supervised updates, overlapping second-stage data, or explicit intermediate thought traces. These findings highlight reinforcement learning as an effective post-training signal for hierarchical syntactic prediction with large language models.

7 Limitations

First, our experiments focus on two representative syntactic prediction tasks, constituency parsing and CCG supertagging. Although these tasks provide a useful testbed for hierarchical structured prediction, it remains to be seen whether the same training recipe generalizes equally well to other structured prediction problems. Second, the 80/20 SFT/RL allocation is tuned for our current high-resource English and Chinese treebanks. In low-resource settings, where the base model has limited coverage of the target language or domain, a larger SFT portion may be needed to build a reliable cold-start policy before RL becomes effective. We leave adap-

tive data allocation across languages and resource levels to future work.

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.
- Xuefeng Bai, Jialong Wu, Yulong Chen, Zhongqing Wang, Kehai Chen, Min Zhang, and Yue Zhang. 2025. [Constituency parsing using llms](#). *IEEE Transactions on Audio, Speech and Language Processing*, 33:3762–3775.
- Srinivas Bangalore and Aravind K. Joshi. 1999. [Supertagging: An approach to almost parsing](#). *Computational Linguistics*, 25(2):237–265.
- 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.
- Chengkun Cai, Xu Zhao, Haoliang Liu, Zhongyu Jiang, Tianfang Zhang, Zongkai Wu, Jenq-Neng Hwang, and Lei Li. 2025. [The role of deductive and inductive reasoning in large language models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16780–16790, Vienna, Austria. Association for Computational Linguistics.
- Jiawei Chen, Yaojie Lu, Hongyu Lin, Jie Lou, Wei Jia, Dai Dai, Hua Wu, Boxi Cao, Xianpei Han, and Le Sun. 2023. [Learning in-context learning for named entity recognition](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13661–13675, Toronto, Canada. 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.
- Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. [Semi-supervised sequence modeling with cross-view training](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

- Stephen Clark. 2002. [Supertagging for Combinatory Categorical Grammar](#). In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pages 19–24, Università di Venezia. Association for Computational Linguistics.
- Stephen Clark and James R. Curran. 2007. [Wide-coverage efficient statistical parsing with CCG and log-linear models](#). *Computational Linguistics*, 33(4):493–552.
- 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.
- Yuchun Fan, Bei Li, Peiguang Li, Yilin Wang, Yongyu Mu, Jian Yang, Xin Chen, Rongxiang Weng, Jingang Wang, Xunliang Cai, JingBo Zhu, and Tong Xiao. 2026. [LANG: Reinforcement learning for multilingual reasoning with language-adaptive hint guidance](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 43838–43866, San Diego, California, United States. Association for Computational Linguistics.
- Daniel Fried, Nikita Kitaev, and Dan Klein. 2019. [Cross-domain generalization of neural constituency parsers](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 323–330, Florence, Italy. Association for Computational Linguistics.
- Michael Ginn and Alexis Palmer. 2025. [LLM dependency parsing with in-context rules](#). In *Proceedings of the 1st Joint Workshop on Large Language Models and Structure Modeling (XLLM 2025)*, pages 186–196, Vienna, Austria. Association for Computational Linguistics.
- Peiming Guo, Meishan Zhang, Jianling Li, Min Zhang, and Yue Zhang. 2025. [Contrastive learning on LLM back generation treebank for cross-domain constituency parsing](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27446–27458, Vienna, Austria. Association for Computational Linguistics.
- Weiyang Guo, Zesheng Shi, Liye Zhao, Jiayuan Ma, Zeen Zhu, Junxian He, Min Zhang, and Jing Li. 2026. [E3-TIR: enhanced experience exploitation for tool-integrated reasoning](#). *CoRR*, abs/2604.09455.
- Han He and Jinho D. Choi. 2023. [Unleashing the true potential of sequence-to-sequence models for sequence tagging and structure parsing](#). *Transactions of the Association for Computational Linguistics*, 11:582–599.
- Julia Hockenmaier and Mark Steedman. 2007. [CCG-bank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank](#). *Computational Linguistics*, 33(3):355–396.
- Matthew Honnibal, James R. Curran, and Johan Bos. 2010. [Rebanking CCGbank for improved NP interpretation](#). In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215, Uppsala, Sweden. Association for Computational Linguistics.
- Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2025. [Disentangling memory and reasoning ability in large language models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1681–1701, Vienna, Austria. Association for Computational Linguistics.
- Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1830–1842.
- 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.
- 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.
- Zhiqiang Kou, Junyang Chen, Xin-Qiang Cai, Xiaobo Xia, Ming-Kun Xie, Dong-Dong Wu, Biao Liu, Yuheng Jia, Xin Geng, Masashi Sugiyama, and Tat-Seng Chua. 2026. [Positive-unlabeled reinforcement learning distillation for on-premise small models](#). *Preprint*, arXiv:2601.20687.
- Zhiqiang Kou, Junyang Chen, Xin-Qiang Cai, Ming-Kun Xie, Biao Liu, Changwei Wang, Lei Feng, Yuheng Jia, Gang Niu, Masashi Sugiyama, and 1 others. 2025. Rethinking toxicity evaluation in large language models: A multi-label perspective. *arXiv preprint arXiv:2510.15007*.
- Zhiqiang Kou, Jing Wang, Jiawei Tang, Yuheng Jia, Boyu Shi, and Xin Geng. 2024. [Exploiting multi-label correlation in label distribution learning](#). In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages

- 4326–4334. International Joint Conferences on Artificial Intelligence Organization. Main Track.
- Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. [LSTM CCG parsing](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, San Diego, California. Association for Computational Linguistics.
- Lei Li. 2024. [Cpseg: Finer-grained image semantic segmentation via chain-of-thought language prompting](#). In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 502–511.
- Yinghao Li, Rampi Ramprasad, and Chao Zhang. 2024. [A simple but effective approach to improve structured language model output for information extraction](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5133–5148, Miami, Florida, USA. Association for Computational Linguistics.
- Haoliang Liu, Chengkun Cai, Xu Zhao, Han Zhu, Shizhou Huang, Xinglin Zhang, Tao Chen, Jenq-Neng Hwang, Zhang Huaping, and Lei Li. 2026. [BCL: Bayesian in-context learning framework for information extraction](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 31172–31189, San Diego, California, United States. 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.
- Yufang Liu, Tao Ji, Yuanbin Wu, and Man Lan. 2021. Generating ccg categories. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13443–13451.
- Xutao Mao, Dixuan Zhang, Yifan Li, Jiaming Hou, Rui Li, YunLong Li, Aoze Zheng, Zhiqiang Zhang, Luo Zhewei, Hongying Zan, and 1 others. 2026. Logiccat: A chain-of-thought text-to-sql benchmark for complex reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 29958–29966.
- 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.
- Hiroshi Matsuda, Chunpeng Ma, and Masayuki Asahara. 2025. [Step-by-step instructions and a simple tabular output format improve the dependency parsing accuracy of LLMs](#). In *Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025)*, pages 11–19, Ljubljana, Slovenia. Association for Computational Linguistics.
- Xianzhe Meng, Qiangsheng Zeng, Ling Luo, Qinghan Yang, Jiarui Hao, Wenbo Wu, Qinyu Wang, Rui Yin, Lin Qi, and Renzhi Lu. 2026. [Stability as a liability: systematic breakdown of linguistic structure in llms](#). *Preprint*, arXiv:2601.18588.
- Khalil Mrini, Franck Démoncourt, 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.
- 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, 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.
- OpenAI. 2024. [Gpt-4 technical report](#). *Preprint*, arXiv:2303.08774.
- Jakob Prange, Nathan Schneider, and Vivek Srikumar. 2021. [Supertagging the Long Tail with Tree-Structured Decoding of Complex Categories](#). *Transactions of the Association for Computational Linguistics*, 9:243–260.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. [Direct preference optimization: Your language model is secretly a reward model](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *Preprint*, arXiv:1707.06347.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *Preprint*, arXiv:2402.03300.
- Behzad Shayegh, Yanshuai Cao, Xiaodan Zhu, Jackie CK Cheung, and Lili Mou. 2024. [Ensemble distillation for unsupervised constituency parsing](#). In *The Twelfth International Conference on Learning Representations*.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the*

- Twentieth European Conference on Computer Systems*, pages 1279–1297.
- Jingwei Shi, Shengyu Tao, Xinxiang Yin, Chen Huang, Wenqiang Lei, and See-Kiong Ng. 2026. [Letgames: An llm-powered gamified approach to cognitive training for patients with cognitive impairment](#). *Preprint*, arXiv:2604.09566.
- Yifei Song, Kun Efimov-Zhang, and Claire Gardent. 2026. [Cross-domain, multi-task data-to-text generation without in-domain training data](#). *Preprint*, arXiv:2608.23391.
- Yuanhe Tian and Yan Song. 2022. [Combinatory grammar tells underlying relevance among entities](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5780–5786, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Yuanhe Tian, Yan Song, and Fei Xia. 2020a. [Supertagging Combinatory Categorical 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.
- Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. 2020b. [Improving constituency parsing with span attention](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1691–1703, Online. Association for Computational Linguistics.
- Yuanhe Tian, Fei Xia, and Yan Song. 2024. [Large language models are no longer shallow parsers](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7131–7142, Bangkok, Thailand. Association for Computational Linguistics.
- Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023. [Gpt-ner: Named entity recognition via large language models](#). *Preprint*, arXiv:2304.10428.
- 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.
- Ronald J. Williams. 1992. [Simple statistical gradient-following algorithms for connectionist reinforcement learning](#). *Machine Learning*, 8(3-4):229–256.
- Tingyu Xie, Qi Li, Jian Zhang, Yan Zhang, Zuozhu Liu, and Hongwei Wang. 2023. [Empirical study of zero-shot NER with ChatGPT](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7935–7956, Singapore. 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.
- Wenduan Xu, Michael Auli, and Stephen Clark. 2015. [CCG supertagging with a recurrent neural network](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 250–255, Beijing, China. Association for Computational Linguistics.
- 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.
- Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, Red Hook, NY, USA. Curran Associates Inc.
- 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.
- Dechen Zhang, Xuan Tang, Xinxiang Yin, Xingwu Chen, Jian Qian, and Difan Zou. 2026. [Valg: An agentic system for ml theory research](#). *Preprint*, arXiv:2608.13060.
- Kun Zhang, Oana Balalau, and Ioana Manolescu. 2025. [Structured discourse representation for factual consistency verification](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 820–838, Vienna, Austria. Association for 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-20*, pages 4046–4053. International Joint Conferences on Artificial Intelligence Organization. Main track.
- Jinman Zhao and Gerald Penn. 2024. [LLM-supertagger: Categorical 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.
- Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyang Luo. 2024. [LlamaFactory: Unified efficient fine-tuning of 100+ language models](#). In

Split	PTB (News)
Train	39,832
Development	1,700
Test	2,426

Table 9: Dataset statistics for the in-domain PTB split. PTB contains news-domain data with standard train/development/test splits.

Dataset		Train	Valid	Test
CCGbank	Token #	929552	45422	55371
	Sentence #	39604	1913	2407
	Category #	1286	394	435
CCG rebank	Token #	943204	46201	56395
	Sentence #	39604	1913	2407
	Category #	1574	478	538

Table 10: Statistics of two categorical grammar banks (CCGbank and CCG rebank) across the train, validation, and test splits.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 400–410, Bangkok, Thailand. 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.

Yixiao Zhou, Dongzhou Cheng, Zhiliang Wu, Yi Yang, Yu Cheng, and Hehe Fan. 2026. One refiner to unlock them all: Inference-time reasoning elicitation via reinforcement query refinement. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 38957–38978.

A Dataset Statistics

A.1 PTB Statistics

Table 9 shows the dataset statistics for the in-domain PTB split. PTB contains training, development, and test splits from the news domain.

A.2 CG Bank Statistics

Table 10 shows the statistics of the two categorical grammar banks used in our experiments, including the numbers of tokens, sentences, and categories in the train, validation, and test splits. CCG rebank has the same sentence split as CCGbank, but contains a larger category inventory and slightly more tokens due to its revised annotation scheme.

Hyperparameter	Value
Batch size	32
Learning rate	1.0×10^{-5}
Number of training epochs	3.0
LR scheduler type	cosine
Warmup ratio	0.1
Epoch	2

Table 11: Hyperparameter settings for SFT.

Hyperparameter	Value
data.train_batch_size	256
actor_rollout_ref.actor.ppo_mini_batch_size	64
actor_rollout_ref.actor.optim.lr	5×10^{-7}
actor_rollout_ref.actor.kl_loss_coef	0.001
actor_rollout_ref.actor.entropy_coef	0
actor_rollout_ref.rollout.n	8

Table 12: Hyperparameter settings for RL.

B Hyperparameter

Table 11 shows the hyperparameter settings for SFT and table 12 shows the hyperparameter settings for RL.

C Baseline Methods

In this section, we provide brief descriptions of the baseline methods used in our experiments. We organize them by task, namely constituency parsing and CCG supertagging.

C.1 Baselines for Constituency Parsing

Kitaev et al. (2019). [Kitaev et al. \(2019\)](#) propose a self-attentive constituency parser built on top of pretrained contextual encoders. Their method scores labeled spans and uses CKY-style decoding to recover the highest-scoring tree. A central feature of this parser is the use of large pretrained language models, such as BERT, which substantially improve span representations and parsing accuracy.

HPSG. HPSG ([Zhou and Zhao, 2019](#)) formulate constituency parsing within a simplified head-driven phrase structure grammar framework. Their model integrates constituent and dependency information into a unified representation and introduces parsing algorithms based on converted span structures. This baseline is representative of joint modeling approaches that exploit both phrase-structure and head information.

Attach-Juxtapose. Attach-Juxtapose ([Yang and Deng, 2020](#)) propose a strongly incremental con-

stituency parser with graph neural networks. Their method uses a novel attach-juxtapose transition system, which builds a single partial tree as each token is processed. Compared with chart-based parsers, this baseline represents a transition-based and incrementally motivated parsing approach.

Span Attention Parser. Span Attention Parser (Tian et al., 2020b) improve neural chart-based constituency parsing with span attention. Their method enriches span representations by incorporating n-gram information and weighting these local patterns according to their relevance to the parsing decision. They further introduce categorical span attention to better model spans of different lengths.

LAL-Parser Mrini et al. (Mrini et al., 2020) propose LAL-Parser, a neural constituency parser with label attention. Instead of relying only on generic self-attention, their model introduces label-aware attention mechanisms that encourage the encoder to capture syntactic distinctions that are directly useful for constituent prediction. This baseline is representative of parser architectures that improve interpretability and syntactic bias through task-specific attention design.

Bai et al. (2023). Bai et al. (Bai et al., 2025) study constituency parsing with large language models. They reformulate parsing as a sequence generation problem by linearizing constituency trees and evaluate zero-shot, few-shot, and supervised settings across multiple LLMs. In our comparison, this line of work serves as the main LLM-based baseline for constituency parsing.

C.2 Baselines for CCG Supertagging

CVT Clark et al. (2018) use cross-view training to improve sequence labeling with unlabeled data. Their approach learns from auxiliary prediction views during semi-supervised training and was shown to be effective on CCG supertagging. This baseline represents strong semi-supervised neural tagging without explicit category structure modeling.

Bhargava and Penn (2020). Bhargava and Penn (2020) treat CCG categories as structured objects rather than atomic labels. They decompose categories into sequences of primitives and operators and train an LSTM-based decoder to generate these sequences. This constructive view makes the model better suited to handling rare and unseen categories.

Liu et al. (2021). Liu et al. (2021) propose to generate CCG categories instead of predicting them with multiclass classification. Their method decomposes each category into a sequence of smaller atomic tags and models supertagging as category generation. This design improves sharing across related categories and offers better robustness on infrequent and unseen tags.

Tree-Structured Decoding. Tree-Structured Decoding (Prange et al., 2021) address the long-tail nature of CCG supertagging with constructive decoding methods. They explicitly model the internal structure of supertags and introduce tree-structured decoding strategies. Their approach is designed to recover rare categories and even generate categories that do not appear in the training set.

AGCN. AgCN (Tian et al., 2020a) propose attentive graph convolutional networks for CCG supertagging. Their method leverages graph-based contextual interactions to enhance token representations beyond standard sequential encoders. This baseline is representative of context-enhanced neural supertaggers that use explicit relational modeling.

Geometry-Aware Supertagging. Geometry-Aware Supertagging (Kogkalidis and Moortgat, 2023) revisit constructive supertagging from a graph-theoretic perspective. They propose geometry-aware supertagging with heterogeneous dynamic graph convolutions to better exploit the structure of the output space. This method is particularly motivated by complex categorial grammars with large and highly structured tag inventories.

Holographic CCG Parsing. Holographic CCG Parsing (Yamaki et al., 2023) propose a holographic formulation of CCG parsing and supertagging. Their method uses holographic embeddings as a compositional operator to explicitly model dependencies between words and phrase structures in continuous space. This baseline represents a compositional neural approach that builds syntactic structure directly in the embedding space.

LLM-supertagger. LLM-supertagger (Zhao and Penn, 2024) explore large language models for categorial grammar supertagging. They evaluate LLMs on both CCG and LCG supertagging and introduce a simple enhancement method that substantially improves performance. This baseline serves as the

main LLM-based comparison for supertagging in our experiments.

D Examples of Intermediate Thought Formats

In this section, we explain the two intermediate thought formats used in the experiment. Both formats prepend a <think> block before the final <answer> tree, but they differ in the style of the intermediate reasoning.

Span-based thought format. This format lists intermediate spans together with coarse constituent labels. An example is shown below:

```
<think>
Who 'd have ... school superintendents
? : SBARQ
'd have thought ... be school
superintendents : SQ
have thought that ... be school
superintendents : VP
thought that the ... be school
superintendents : VP
that the next ... be school
superintendents : SBAR
the next group ... be school
superintendents : S
the next group ... reputations like that
: NP
of tough guys ... reputations like that
: PP
tough guys carrying ... reputations like
that : NP
carrying around reputations like that :
VP
... </think>
<answer>(SBARQ (WHNP (WP Who)) (SQ (MD
'd) ... (. ?))</answer>
```

Tree-sketch thought format. This format provides a coarse unlabeled or partially labeled bracketed sketch before the final parse tree. An example is shown below:

```
<think>
((Who) ('d (have thought that the next
group of tough guys carrying around
reputations like that would be school
superintendents)) ?)
</think>
<answer>(TOP (SBARQ (WHNP (WP Who)) (SQ
(MD 'd) ... (. ?))</answer>
```

These two formats were designed to test whether explicit intermediate decomposition would help constituency parsing. Empirically, however, both performed worse than direct answer-only format.

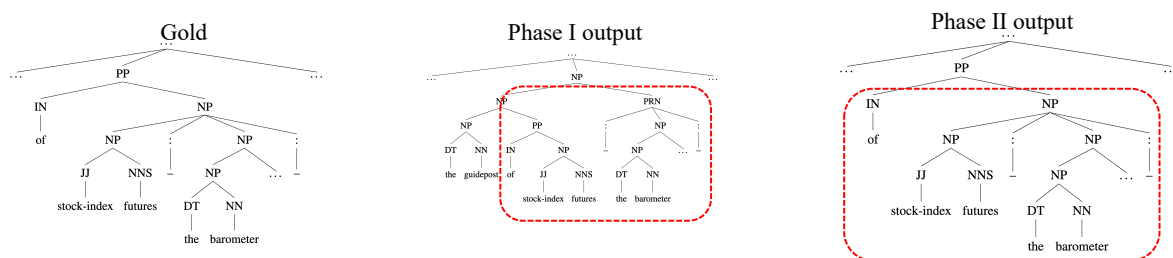
E Case Study

Figure 4 provides two representative examples of how Syntax-RL changes the predicted tree structure

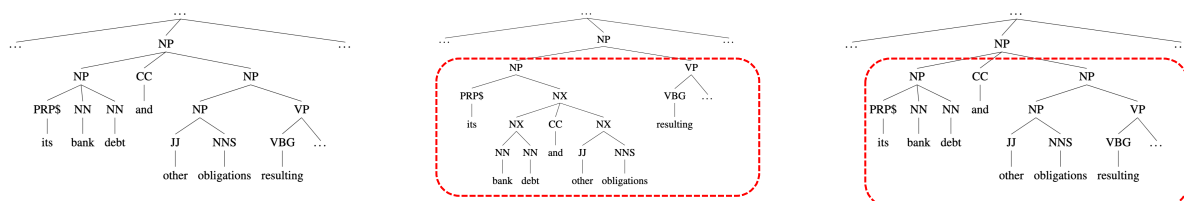
relative to the phase-1 SFT model. In both cases, the improvement is structural rather than lexical. The first example involves a non-canonical dashed insertion, and the second involves coordination together with participial modifier attachment inside a complex noun phrase. These examples illustrate the same broader pattern: the RL stage helps the model make cleaner constituent-boundary and attachment decisions, especially in cases where the phase-1 SFT model produces flatter or externally attached structures.

F AI Use Disclosure.

We used AI-based writing assistance tools only for limited language polishing, such as grammar correction and minor wording refinement. The core ideas, experimental design, analysis, and all technical content were developed by the authors. We did not use AI tools to generate code, run experiments, or produce the reported results.



(a) Case 1: dashed appositive attachment. In the phase-1 SFT prediction, the explanatory phrase introduced by dashes is analyzed as an external PRN constituent outside the of-complement. Syntax-RL recovers the gold high-level structure by placing “*stock-index futures – the barometer ...*” inside a single NP within the of-phrase.



(b) Case 2: coordination and modifier attachment inside a complex noun phrase. In the phase-1 SFT prediction, “*bank debt and other obligations*” is collapsed into a flatter NX-style structure, and the participial modifier “*resulting from ...*” is attached outside the coordinated nominal span. Syntax-RL recovers the gold structure by separating “*its bank debt*” and “*other obligations*” into two coordinated constituents and attaching the participial modifier specifically to the latter.

Figure 4: Case studies comparing the gold tree, the phase-1 SFT prediction, and the phase-2 prediction. In Case 1, RL corrects a dashed appositive boundary by moving the explanatory phrase from an external parenthetical structure into the NP internal to the of-phrase. In Case 2, RL corrects NP-internal coordination and modifier attachment by restructuring the conjuncts and attaching the participial modifier to the second conjunct only.