Self-Correction Makes LLMs Better Parsers

Ziyan Zhang, Yang Hou, Chen Gong*, Zhenghua Li
School of Computer Science and Technology, Soochow University
{zyzhang0509,yhou1}@stu.suda.edu.cn
{gongchen18,zhli13}@suda.edu.cn

Abstract

Large language models (LLMs) have achieved remarkable success across various natural language processing (NLP) tasks. However, recent studies suggest that they still face challenges in performing fundamental NLP tasks essential for deep language understanding, particularly syntactic parsing. In this paper, we conduct an in-depth analysis of LLM parsing capabilities, delving into the underlying causes of why LLMs struggle with this task and the specific shortcomings they exhibit. We find that LLMs may be limited in their ability to fully leverage grammar rules from existing treebanks, restricting their capability to generate syntactic structures. To help LLMs acquire knowledge without additional training, we propose a self-correction method that leverages grammar rules from existing treebanks to guide LLMs in correcting previous errors. Specifically, we automatically detect potential errors and dynamically search for relevant rules, offering hints and examples to guide LLMs in making corrections themselves. Experimental results on three datasets using various LLMs demonstrate that our method significantly improves performance in both in-domain and cross-domain settings.

1 Introduction

LLMs have exhibited significant success across a wide range of NLP tasks and applications (Kojima et al., 2022; Raunak et al., 2023; Chiang and Lee, 2023). However, in the field of syntax parsing, enabling LLMs to achieve a deep structural understanding of sentences remains challenging (Bai et al., 2023; Li et al., 2023; Tian et al., 2024). Although non-LLM parsers have achieved over 95% F-scores in the newswire domain, the performance of LLMs falls significantly short. This discrepancy inevitably raises two questions: Where exactly does the parsing capability of LLMs fall short? What specific knowledge are LLMs lacking?

Prior works have provided some findings about the parsing capabilities of LLMs. For instance, Bai et al. (2023) pointed out hallucinations in LLM outputs, while Tian et al. (2024) highlighted that they tend to be much flatter than gold trees. However, existing works primarily analyze high-level performance metrics without delving into the detailed shortcomings or the underlying causes. To address these gaps, we conduct an in-depth analysis of LLM parsing capabilities, examining both the structural characteristics of their outputs and the specific types of errors they exhibit. Our findings suggest that the inability to fully utilize grammar rules in the training data is a potential reason why LLMs fail. Unlike non-LLM parsers, which can comprehensively learn rules from extensive training data, LLMs solely rely on limited contextual examples provided during few-shot learning. This fundamental difference distinguishes LLM parsing performance from non-LLM parsers.

Motivated by the above analysis, we are interested in methods that allow LLMs to learn structural knowledge without additional training. The method is not intended for making LLM parsers practical (competitive to non-LLM parsers), but for analyzing the performance of LLMs on the parsing task. Therefore, in this work, we propose a self-correction method that first identifies errors in the initial answer, then searches the treebank for relevant examples based on the characteristics of different types of errors, and finally guides LLMs in correcting errors via hints and examples. This method effectively enables LLMs to reduce all types of errors in their parsing results, thereby improving the overall parsing performance. Experimental results prove that our method achieves significant improvement in both in-domain and cross-domain settings.

In conclusion, our contributions are three-fold:

- We conduct an in-depth analysis of LLM parsing capabilities from the perspective of the overall

* Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 6749–6762

parsing performance, the characteristics of their parsing results over grammar rules, and the four types of errors they make.

- We propose a self-correction method that enables LLMs to learn from existing treebanks and correct all types of errors in their previous predictions, enhancing parsing capability without the need for additional training.
- We conduct extensive experiments on various LLMs, and our method achieves consistent improvements across different datasets. Further analysis demonstrates that our method effectively mitigates the limitation of LLM parsing.

Our code is available at https://github.com/zzy0509/Self-Correction-Parsing.

2 In-depth Analysis of LLM Parsing Capability

LLMs have demonstrated impressive performance in a wide range of NLP tasks (Brown et al., 2020; Wei et al., 2022). However, recent studies indicate that LLMs face significant challenges when performing syntactic parsing tasks, such as producing parse trees that are flatter than gold trees (Bai et al., 2023; Tian et al., 2024). Although previous studies have explored the parsing capability of LLMs to some extent, they remain relatively limited in providing deep insights into the potential reason why LLMs fail. Both studies lack a comprehensive analysis of the reasons behind the weak performance and the specific types of errors LLMs make.

To gain a deeper understanding of the constituency parsing capacity of LLMs, we conduct an in-depth analysis from the perspectives of the overall parsing performance, the characteristics of their parsing results, and the types of errors they make. Specifically, we first evaluate the parsing performance of different LLMs under the few-shot setting, comparing these results to those from one of the non-LLM SOTA models, the Berkeley parser (Kitaev and Klein, 2018). Second, we compare the parsing results of LLMs with both gold trees and those produced by SOTA non-LLM parsers, aiming to analyze the distinct characteristics of LLMs and non-LLM parsers and uncover reasons for the significant performance gap between them. Finally, we categorize the parsing errors made by LLMs into four types and analyze the error distribution, and conduct an in-depth examination to reveal which types of errors LLMs commonly make.

Dataset	Model	R	P	F
PTB	Berkeley	95.56	96.10	95.82
	LLaMa-8B	11.39	11.91	11.64
	LLaMa-70B	45.02	51.24	47.92
	GPT-3.5	62.24	71.62	66.60
	GPT-4	69.97	77.14	73.38
CTB5	Berkeley	88.76	92.82	90.74
	LLaMa-8B	7.01	11.68	8.77
	Qwen-72B	34.45	45.60	39.24
	DeepSeek-v3	43.49	52.71	47.65
	GPT-3.5	27.19	45.84	34.14
	GPT-4	40.44	49.95	44.69
MCTB	Berkeley	86.38	88.50	87.43
	LLaMa-8B	9.67	9.48	9.57
	LLaMa-70B	40.77	48.40	44.26
	GPT-3.5	51.71	62.71	56.68
	GPT-4	62.29	67.98	65.01

Table 1: The overall parsing performance of different models on the PTB, CTB5, and MCTB datasets.

2.1 Overall Parsing Performance of LLMs

We evaluate the capability of LLMs in both in-domain and cross-domain settings. For in-domain evaluation, we use the original test sets of the Penn Treebank (PTB) (Marcus et al., 1993) and Chinese Penn Treebank 5 (CTB5) (Xue et al., 2005). For cross-domain evaluation, to manage costs, we use a subset of the Multi-domain Constituent Treebank (MCTB) (Yang et al., 2022b), randomly sampling 200 sentences from each domain of MCTB, which covers Dialogue, Forum, Law, Literature, and Review domains, resulting in a 1,000 sentences test set. We examine whether these datasets have been leaked to the LLMs in Appendix A.3. Our experiments include both open-sourced LLMs, i.e., LLaMA-3-8B [1], LLaMA-3-70B (Meta, 2024), Qwen-2.5-72B (Yang et al., 2024) and DeepSeek-v3 (DeepSeek-AI et al., 2024), and closed-sourced LLMs, i.e., GPT-3.5 [2] and GPT-4 [3]. To understand the gap between LLMs and non-LLMs, we also report the performance of Berkeley Neural Parser (Kitaev and Klein, 2018), an existing SOTA neural parser. For the in-domain setting, we train Berkeley on the train set of PTB and CTB5, respectively. For the cross-domain setting, we train Berkeley on the train set of PTB and evaluate on the MCTB.

Following standard practice in previous works, we sample five examples in PTB/CTB-train for the few-shot setting. We set temperature to 0 follow-

[1]We use the English version of LLaMA-3 from https://huggingface.co/meta-llama and the Chinese version from https://github.com/ymcui/Chinese-LLaMA-Alpaca-3.

[2]https://platform.openai.com/docs/models/gpt-3.5-turbo

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

ing prior works while retaining default settings for other parameters. After generation, we process the tree valid by adding left or right brackets when the brackets of LLMs are mismatched. Finally, we evaluate the performance of models with the standard evaluation toolkit EVALB[4]. Details of the LLM parsing prompt are shown in Appendix A.1.

Table 1 presents the results of various LLMs on different test sets. From the results, we can draw the following conclusions. First, the performance of all LLMs shows a significant decrease compared to that of the traditional non-LLM parser. This may be because the non-LLM parser has thoroughly learned the structures present in the existing treebanks while LLMs may not. We will discuss this in more detail in Section 2.2. Second, among all LLMs, closed-source LLMs (GPT-3.5 and GPT-4) deliver better results than open-source LLMs. Third, when adapted to the cross-domain test set, all the parsers exhibit a large performance drop. This indicates a notable gap between specific-domain and general-domain data, highlighting the greater challenges posed by cross-domain parsing.

The above results naturally give rise to two important considerations: 1) the specific limitations of LLMs in parsing capabilities compared to non-LLM parsers, and 2) the unique characteristics of LLM-generated parse results in comparison to gold-standard trees. Therefore, we conduct a more in-depth analysis in Section 2.2 and Section 2.3.

2.2 Analysis over Grammar Rules

Previous study (Dakota and Kübler, 2021) has proved that parsers tend to generate structures that have been seen in the train set, and most of these known structures are parsed correctly.[5] Unlike non-LLM parsers that can learn from the entire train set, LLMs are limited to a few examples in the prompt, thus lacking approaches to all grammar rules. To investigate whether these known grammar rules are significant factors that affect performance, we present rule statistics of the parsing results in Table 2. The third column represents the absolute number of rules in the parsing results. The fourth column gives the number of known rules. The fifth and sixth columns show the accuracy of known rules and unknown rules, respectively.

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

[5] In this paper, "structures" refers to rules, which are defined as subtrees invlove a parent node and its child nodes. For example, the subtree "the proposed $ 7 billion bill" in Figure 1(b) corresponds to the rule "NP → DT VBN ADJP".

Dataset	Model	# Total		Accuracy (%)	
		parsed	known	known	unknown
PTB	Berkeley	46,441	26,745	96.20	60.18
	LLaMa-8B	42,710	13,616	26.88	1.93
	LLaMa-70B	38,895	19,597	68.11	19.82
	GPT-3.5	38,958	18,409	79.47	33.11
	GPT-4	40,166	20,291	81.44	37.14
CTB5	Berkeley	8,691	6,198	90.19	76.82
	LLaMa-8B	5,792	1,993	36.78	4.53
	Qwen-72B	6,851	3,043	59.19	13.81
	DeepSeek-v3	7,491	4,259	64.94	23.33
	GPT-3.5	5,480	2,489	55.85	11.07
	GPT-4	7,342	3,479	59.73	17.19
MCTB	Berkeley	17,744	9,916	86.24	68.19
	LLaMa-8B	17,587	5,688	35.07	2.15
	LLaMa-70B	14,457	7,518	64.34	16.50
	GPT-3.5	14,155	7,091	73.01	27.34
	GPT-4	15,731	8,329	73.54	33.27

Table 2: Rule statistics of the parsing results from different models across different datasets.

We observe that the parsing results from non-LLM parsers contain more numbers of known rules compared to those from LLMs. Further analyzing the accuracy of these rules, we find that known rules from non-LLM achieve a high accuracy rate of 86.24% to 96.20% , whereas those from LLMs are much lower. This suggests that non-LLM parsers effectively learn common and valid rules from existing treebanks and apply them during parsing on different domains. For LLMs, they show a significant decrease in performance on both known rules and unknown rules. However, the deeper reason behind this is that a portion of known rules in the parsing results are incorrectly predicted as unknown rules, as the absolute number and proportion of known rules differ significantly from those of the non-LLM. Moreover, performance improves as the total number and accuracy of known rules increase. Therefore, we speculate that LLMs may have rarely encountered the grammar rules during pre-training, leading to a failure to apply them during parsing. The total number and accuracy of known rules are the main factors affecting the performance, and the known rules are the specific knowledge that LLMs lack. The significant performance gap between LLMs and non-LLM parsers may arise from LLMs not systematically learning these rules during training, which increases the likelihood of generating invalid structures during parsing.

2.3 Analysis over Four Types of Errors

Motivated by previous works (Dakota and Kübler, 2021; Kummerfeld et al., 2013), we divide parsing

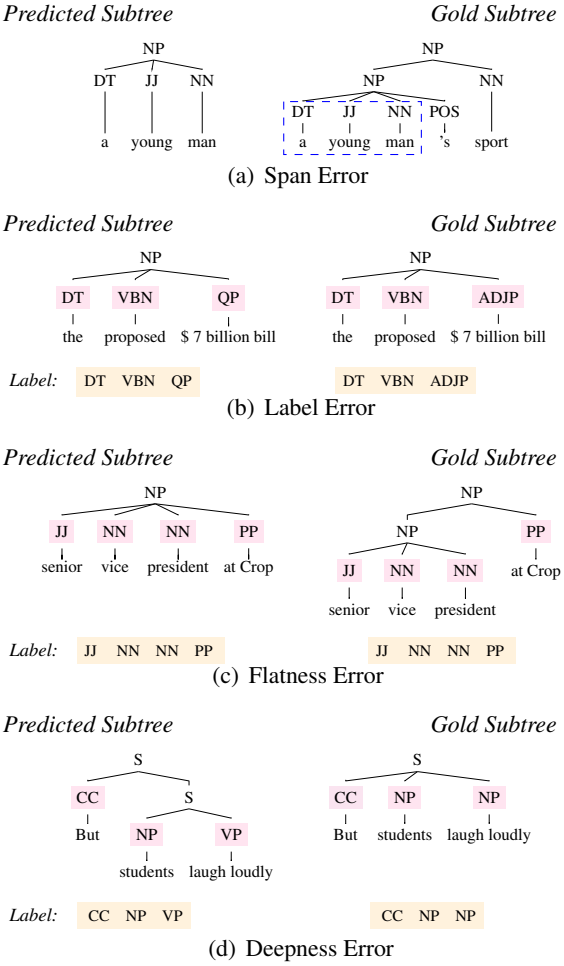


Figure 1: Four types of errors.

errors into four types, as illustrated in Figure 1. Note that we define parsing errors of subtrees by considering two levels: the root node and its child nodes, while disregarding the internal structure of the lower layers within the subtree.

Span Errors. We define "span" as a sequence of words from $word_i$ to $word_j$ that forms a constituent and we do not consider the label, as it will be involved when identifying label errors. Therefore, span error represents that the boundary of the parent node does not align with the gold. For instance, the span of the subtree "a young man" in Figure 1(a) ranging from "a" to "man" does not exist in the gold "a young man 's sport".

Label Errors. The predicted subtree has the same span with the gold subtree, but one or more constituent labels are predicted incorrectly, as shown in Figure 1(b). It could be an error in the label of the parent node or the child nodes.

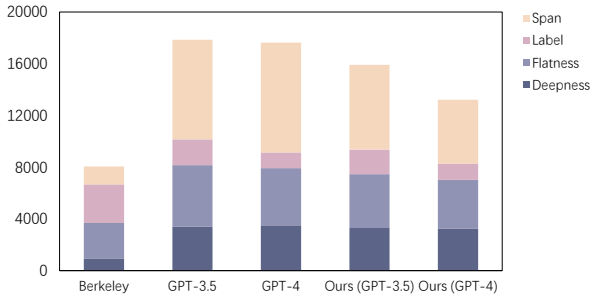


Figure 2: The overview of the four types of errors made by different models on the PTB.

Flatness Errors. The predicted subtree is flatter than the gold subtree, which represents the parent node in the predicted subtree contains more child nodes, as shown in Figure 1(c).

Deepness Errors. Contrary to flatness errors, the predicted subtree is deeper than the gold subtree, which represents the parent node contains fewer child nodes, as shown in Figure 1(d).

Given a subtree, we first determine whether it is a span error or one of the other three error types by checking if the span of the parent node corresponding to the subtree appears in the gold tree. Furthermore, we use the number of child nodes to determine which of the three error types it is. In this way, these four types of errors can encompass all the mistakes made by the parsers. Certainly, compound situations may occur, such as a label error within a flatness error or a deepness error, but we do not further categorize these.

Figure 2 shows an overview of the error types made by Berkeley, GPT-3.5 and GPT-4 on the PTB. As expected, the total number of errors increases as the performance of models declines, and LLMs make significantly more errors than non-LLMs. Among all four types of errors, span errors account for the largest proportion and the direct cause might be the inability of LLMs to correctly delineate spans. For flatness errors, they make up a relatively larger proportion among the remaining three types of errors that occur within subtrees. For deepness error, the parsing results of LLM contain fewer deepness errors than those of non-LLM in the in-domain setting. We hypothesize that this is due to the characteristic of LLMs to generate flatter structures, which results in a greater number of flatness errors. Because of this characteristic, there are fewer instances where the predicted trees are deeper than the gold trees, leading to a fewer number of deepness errors compared to non-LLMs

in the in-domain setting. We provide the distributions of errors across different models and different dataset in Appendix A.8.

3 Our Self-Correction Approach

The above results and analysis help us better understand the shortcomings of LLM parsing results and encourage us to propose targeted methods for enhancing LLM parsing capabilities. Since Section 2.2 has proved that the low performance of LLMs may stem from their lack of known rules in the existing treebanks, thereby we are interested in exploring methodologies that enable LLMs to effectively assimilate knowledge from the existing treebanks without the need for additional training to improve their constituency parsing capabilities. Furthermore, we aim at a method based on the four types of parsing errors mentioned in Section 2.3 to effectively address all errors made by the LLM. In this work, we propose a self-correction method based on the rules from the existing treebank, guiding LLMs to correct errors in the previous answers. Consistent with the common process of self-correction in previous works (Pan et al., 2024; Tong et al., 2024), we first prompt LLM to generate a base answer, identify potential errors in it, and then guide LLM to correct these errors through specific hints and examples.

Specifically, based on the characteristics of parsing results, we categorize self-correction into two parts: sentence mismatch correction and tree structure correction. First, we perform mismatch correction to ensure the alignment of the leaf nodes between the predicted trees and gold trees. Then we make structure correction, which addresses four types of structural errors introduced in 2.3.

3.1 Sentence Mismatch Correction

We find that LLMs often make changes on their own, leading to parse trees containing new words which do not exist in the original sentence. This phenomenon has also been demonstrated by Bai et al. (2023), leading to a significant impact on the performance. Therefore, we first make sentence mismatch correction. After obtaining base answers, we identify errors and design different hints for different errors, letting LLM generate a new answer. The detail of hints is shown in Appendix A.2.

3.2 Tree Structure Correction

After ensuring the alignment of the leaf nodes between predicted trees and gold trees, we make tree structure correction. Specifically, we identify errors by comparing them with grammar rules in the existing treebank and correct subtrees of different heights from the top to the bottom. Compared with previous works that focus on the design of the prompt, the advantages of our method are threefold: 1) Previous work (Huang et al., 2023) has shown that without external knowledge, it is difficult for LLMs to achieve intrinsic self-correction solely on their own capabilities. Therefore, considering that the number and accuracy of known rules are decisive factors influencing performance, **we leverage existing treebanks as external treebank to enable LLMs quickly acquire structural knowledge.** 2) Previous work (Tyen et al., 2023) has shown that the difficulty of intrinsic self-correction lies in the inability of LLMs to identify the location of errors, rather than their inability to correct errors. Therefore, **we directly rely on the rules in the treebank to identify errors, avoiding the issue where LLMs fail to locate the errors.** Our method preserves the correct parts of the parsing results and focuses solely on correcting the potentially incorrect parts, which greatly saves time and resources. 3) **Our correction covers all types of parsing errors.** For different errors, our proposed method can dynamically search rules in the existing treebanks to serve as hints to guide the correction. As shown in Figure 3, our tree structure correction can be divided into four detailed steps:

1) Identifying Errors Based on Rules. To identify errors in the base answer, we extract rules of the subtree and search it in the existing treebanks[6] . If the rule is not found in the treebank, we determine it is likely incorrect and requires correction.

2) Processing Rules based on Errors. Once an error is identified, we aim at searching rules that can guide the correction of the current error. Therefore, we traverse rules in the existing treebanks and process them based on the characteristics of different errors so that the correction involves all types of errors. The main idea is that if the predicted rule and the traversed rule can be transformed into the same form after applying a specific processing method, we can infer that the predicted rule has made the corresponding error. As is shown in Figure 1, for label error, we directly take the predicted rule and the traversed rule. For flatness error, we replace the child nodes of the traversed rule with their

[6]We use the train sets of PTB and CTB5 as the existing treebanks for English and Chinese, respectively.

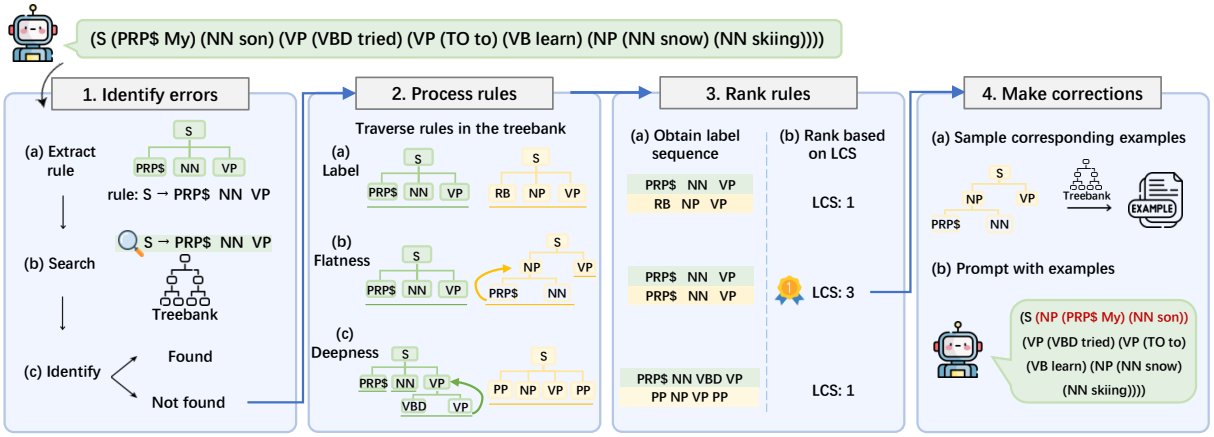


Figure 3: The process of structure correction. For clarify, we provide a typical example of rule for each processing method in the figure. In practice, each rule needs to undergo three types of processing methods and be ranked.

own child nodes. For deepness error, on the contrary to flatness error, we replace the child nodes of the predicted rule with their own child nodes.

3) Ranking Based on the Similarity. After processing, our goal is to find rules most similar to the gold trees to serve as examples. If two rules exhibit significant similarity following the application of specific processing method, the traversed rules are considered close to the gold, and can provide valuable structural information to LLMs. To determine the similarity, we take the constituent labels of the child nodes to form label sequences as shown in Figure 3 and use the Longest Common Subsequence (LCS) as the metric. If there are rules with the same LCS, we prioritize the rule that appears more frequently in the treebank. Then we rank traversed rules and select the top five rules, considering them can guide the LLM in self-correction.

4) Guiding LLM in Making Self-Corrections. Finally, we sample examples corresponding to these rules from the treebank and incorporate them into the prompt to guide the LLM in generating new answers. Notably, our approach avoids excessive interference with the self-correction process. The entire procedure relies solely on the own capabilities of LLMs, allowing it to either reference the provided examples to produce new answers or retain its original results, with the final determination entirely left to the their judgment.

Through different processing methods, our method can directly correct label , flatness, and deepness error. Different from these three types, span error occurs outside the subtree (rules) and cannot be resolved by making corrections to the subtrees. However, since our method starts from the top of the entire tree and correct subtrees of different heights from top to bottom, we speculate that this process can also indirectly help resolve span errors, which has been proven in Experiments 4.4. Additionally, we only consider the constituency labels within the subtree, ignoring the label of root node. Because we assume that the label of root node has already been corrected during the correction of higher subtrees. Also, setting a constraint that the label of root node must match the predicted rules during traversing significantly helps reduce time. We also compare our method with random selection in Appendix A.4 and investigate the effect of different numbers of final rules and different searching strategies in Appendix A.5 and A.6.

Dataset	Model	R	P	F
PTB	GPT-3.5†	72.48	84.86	78.18
	GPT-4†	77.74	**89.04**	83.00
	LLaMa-8B	29.42$_{+18.03}$	38.27$_{+26.36}$	33.27$_{+21.63}$
	LLaMa-70B	54.90$_{+9.88}$	61.49$_{+10.25}$	58.01$_{+10.09}$
	GPT-3.5	74.59$_{+12.35}$	80.11 $_{+8.49}$	77.25$_{+10.65}$
	GPT-4	**82.27**$_{+12.30}$	84.78 $_{+7.64}$	**83.50**$_{+10.12}$
CTB5	LLaMa-8B	22.63$_{+15.62}$	34.22$_{+22.54}$	27.24$_{+18.47}$
	Qwen-72B	40.12 $_{+5.67}$	52.18 $_{+6.58}$	45.36 $_{+6.12}$
	DeepSeek-v3	57.33$_{+13.84}$	65.08$_{+12.37}$	60.96$_{+13.31}$
	GPT-3.5	39.50$_{+12.31}$	57.12$_{+11.28}$	46.70$_{+12.56}$
	GPT-4	**61.67**$_{+21.23}$	**68.69**$_{+18.74}$	**64.99**$_{+20.30}$
MCTB	LLaMa-8B	25.91$_{+16.24}$	34.01$_{+24.53}$	29.41$_{+19.84}$
	LLaMa-70B	48.31 $_{+7.54}$	55.68 $_{+7.28}$	51.73 $_{+7.47}$
	GPT-3.5	60.14 $_{+8.43}$	71.71 $_{+8.99}$	65.41 $_{+8.73}$
	GPT-4	**71.71** $_{+9.42}$	**77.13** $_{+9.15}$	**74.32** $_{+9.31}$

Table 3: The main result of our self-correction method. The "**Bold**" identifies the best performance. The † denotes the results referred from Tian et al. (2024). The subscript $_+$ indicates the improvement.

4 Experiments

We conduct experiments to verify the effectiveness of our self-correction method and perform a de-

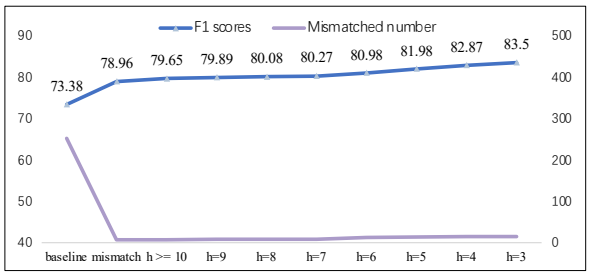
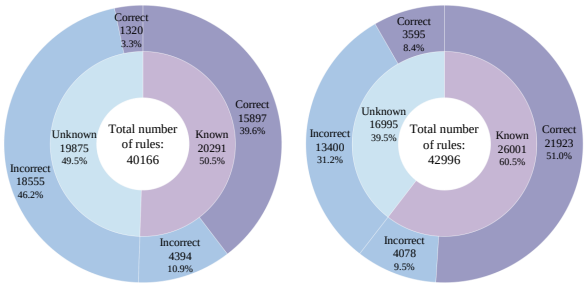


Figure 4: The effect of different corrections. "mismatch" represents sentence mismatch correction and "h" represents the height of subtrees in tree structure correction.



(a) GPT-4 (b) GPT-4 (Ours)

Figure 5: Rule statistics of the parsing results generated from GPT-4 before and after applying our self-correction method on the PTB dataset.

tailed analysis to gain insights into what LLMs have learned through our method. All experimental settings remain consistent with those in direct LLM-based parsing, as described in Section 2.1.

4.1 Main Results

The main experimental results of our method are listed in Table 3. Compared to the results of directly using LLM for parsing with five-shot learning in Table 1 , our proposed self-correction method achieves substantial and consistent improvements across all datasets. In particular, for the in-domain setting, our method achieves an improvement of over 10 F1 scores on both PTB and CTB5. For cross-domain settings, despite the gap between the general treebank and the target domain test set, our method still brings consistent improvements.

Compared to the result of Tian et al. (2024), which first prompts LLMs for chunking and then use the chunking results to guide LLM parsing, though our result with GPT-3.5 is 0.93 F1 score lower on PTB, the result with GPT-4 is 0.5 F1 score higher. We hypothesize that this is because our method requires LLMs to learn from examples and make independent judgments by themselves. Therefore, the stronger the capability of an LLM, the greater its performance improves. *Overall, our method consistently achieves significant improvements under both in-domain and cross-domain settings , effectively mitigating the limitation of overly flatter parsing results from LLMs.*

4.2 The Effect of Different Corrections

To analyze the effectiveness of sentence mismatch correction and tree structure correction, we evaluate their respective improvements across varying tree heights with GPT-4 on the PTB. As shown in Figure 4, first, LLMs generate a substantial number of mismatched trees during parsing, which greatly affects the parsing performance. By implementing our method, the number of mismatched trees is significantly reduced, yielding substantial performance gains. Second, tree structure correction also significantly enhances the performance. Specifically, the improvement rate of the performance increases as the height of subtrees decrease, which means correcting lower subtrees is more effective. This may be because the rules of lower subtrees have a higher overlap rate between the treebank and the test set.

4.3 Has LLMs Truly Acquired Knowledge?

To investigate whether the LLM truly learns structural knowledge during self-correction, we compare the rule statistic before and after applying our method with GPT-4 on the PTB. As shown in Figure 5, we categorize rules into known rules and unknown rules and further analyze their accuracy. First, our method significantly increases the absolute number of rules, bringing it closer to that of the gold trees. Second, consistent with our motivation that enable LLMs to assimilate knowledge from existing treebank without training as illustrated in Section 3, our method effectively enhances both the number of known rules and their accuracy. Surprisingly, our method also raises the accuracy of unknown rules. This demonstrates that our method not only enables LLMs to learn knowledge from the treebank, but also enhances their capability to generate correct rules, regardless of whether they have encountered those rules before.

4.4 Which Types Do LLMs Correct?

Although our method specifically targets four types of parsing errors, we further investigate which error types are most effectively mitigated by LLMs. Therefore, we analyze the number of different errors after applying our method with GPT-3.5 and

	NP Interal Structure	PP Attachment	Single Word Phrase	Different Label	Clause Attachment	Modifier Attachment	NP Attachment	VP Attachment
GPT-4	1736	1390	1258	728	664	454	451	130
GPT-4 (Ours)	774	1263	973	688	673	420	453	128

Table 4: Deeper error analysis of GPT-4 before and after applying our self-correction method on the PTB dataset.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
GPT-4	40.47	41.66	32.68	52.16	61.07	47.77	61.68	61.97
GPT-4 (Ours)	**46.39**	**50.42**	**42.91**	**63.31**	**69.33**	**55.63**	**73.50**	**66.17**

Table 5: Results on the SPMRL dataset.

GPT-4. As is shown in Figure 2, we observe consistent reduction across all error categories, confirming the empirical efficacy of our approach. Our method addresses span errors the most, which are the most frequent errors in the parsing results. As we hypothesized, although span errors cannot be directly corrected, our method can indirectly reduce these errors through the top-down correction process. The number of the other three error types also reduce significantly, demonstrating that our method enables LLMs to identify and prioritize rules similar to the gold ones.

Moreover, we also incorporate enhanced analysis from the perspective of grammar rules. Our approach demonstrated significant effectiveness in reducing structural errors for NP and ADVP through treebank rule enforcement: specifically, the "NP $\rightarrow$ CD NN" error decreased from 57 to 22 cases, "NP $\rightarrow$ NNP NNP NNP" errors dropped from 39 to 18 instances. This phenomenon may stem from the comparative simplicity of these structures and their high frequency of occurrence in treebanks. However, quantifier-related structural errors, an inherent weakness in LLMs, showed limited improvement, exemplified by persistent challenges in "QP $\rightarrow$ CD CD" and "QP $\rightarrow$ RB CD" patterns where rule-based corrections proved insufficient.

4.5 The Effect of Our Method on the Unknown Rules

Our method primarily aims to enhance the parsing performance of LLMs through known rules, but we are also curious whether it can contribute to predicting unknown rules. Experimental results show that after applying our method, the accuracy of unknown rules increased from 35.3% to 48.0%. This indicates that our approach not only preserves the inherent capability of LLMs to handle unknown rules but also systematically enhances their overall parsing capacity through known rules, consequently achieving significant accuracy improvement on unknown rules. This phenomenon may be attributed to the ability of LLMs to extract similar structural patterns or localized information from the exemplars, even when some rules are not fully aligned with those in the treebank, thereby still contributing to parsing capability.

4.6 Deeper error analysis of LLM behavior

To conduct a deeper error analysis of LLM behavior, we also conduct additional experiments following the error analysis framework of Kummerfeld et al. (2012) on PTB with GPT-4. As shown in Table 4, after applying our method, most error types have been significantly reduced. Specifically, our approach demonstrates particular effectiveness in addressing NP Internal Structure, PP Attachment, and Single Word Phrase errors, with their counts markedly decreasing. However, error types such as VP Attachment and NP Attachment remain challenging for LLMs, maintaining nearly the same error rates even after implementing our method. This provides critical insights into the parsing capabilities of LLMs, particularly their strengths and limitations.

4.7 Results on the Other Dataset

We also report our results on the SPMRL (Seddah et al., 2013) with GPT-4, which contains eight different languages. We randomly sample 500 data from each as the test set. As is shown in Table 5, our method shows significant improvements in both of these languages, indicating the effectiveness of our method across different types of languages.

4.8 Whether Our method Add New Errors?

we conducte a detailed analysis of the instances where originally correct parsing trees are corrected

to be wrong during the process. Taking the parsing results of GPT-4 on the PTB as an example, initially, there are 24,203 grammar rules that were correct and consistent with the gold trees. After applying our method, 23,279 of these rules remain correct, indicating that our method has a wrong correction rate of only 4%. Considering that the correction brings the improvement of over 10 F1 scores, it indicates that the correct corrections far outweigh the adding errors.

5 Related Work

LLM Parsing. LLMs have achieved remarkable success in various tasks and applications (Kojima et al., 2022; Wei et al., 2022). However, recent studies indicate that LLMs exhibit week capability in parsing and show a noticeable gap compared to that of non-LLM parsers. Bai et al. (2023) revealed that LLMs suffer from hallucinations and have limited ability to learn extremely long constituents. Zhou et al. (2023) conducted experiments on 24 LLMs and found that most of them have a limited grasp of syntactic knowledge. Tian et al. (2024) observed that LLMs are shallow parsers in that they are ineffective at conducting fullparsing. They propose a three-step approach that firstly use LLMs for chunking and then add the chunks to guide LLMs for parsing. These previous works have gained insights into the parsing capability of LLMs. However, they lack deeper analysis of the reasons behind the low performance and the characteristics of their parsing results. We first conduct an in-depth analysis of LLM parsing capability, investigating the detailed shortcomings of their parsing results and the underlying causes. Based on these analyses, we design a method that enables LLMs to obtain knowledge from the existing treebank, enhancing their parsing ability significantly.

Self-Correction. To unleash the reasoning abilities of LLMs, many recent work focus on design prompt to instruct LLMs to solve problems with human-like logic, such as Chain-of-Thought (Wei et al., 2022), Self-Consistency (Wang et al., 2023) and Self-Correction (Gao et al., 2023b; Madaan et al., 2024). Self-Correction, which is defined as the process of continuously evaluating and adjusting previous responses to get better answers, has attracted widespread researches due to its alignment with typical human learning strategy and its notable effectiveness across various NLP tasks. It can be divided into two types: one is that LLMs correct entirely on their own knowledge and ability; The other is for LLMs to correct with external feedback, such as knowledge sources Gao et al. (2023a), other models or tools (Yang et al., 2022a). Recent studies have questioned the efficiency of intrinsic Self-Correction of LLMs. Huang et al. (2023) revealed that without external feedback, it is difficult for LLMs to correct their own mistakes. Tyen et al. (2023) proved that LLMs mainly fail in finding errors in the answers, but can correct them given the error location. To tackle these challenges, we propose a self-correction method that automatically identifies errors in the parsing results and leverages the existing treebanks as external feedback to guide the corrections, thereby enhancing the parsing capability of LLMs.

6 Conclusion

In this paper, we conduct an in-depth analysis of LLM parsing capability from the perspective of the overall parsing performance, the characteristics of their parsing results based on the rules, and the types of errors they made. Based on the analysis that the low performance of LLMs may stem from their lack of known rules in existing treebanks, we propose a self-correction method that enables LLMs to efficiently acquire rules from treebanks without additional training, thereby correcting errors in the previous answer. Experimental results demonstrate that our method consistently improves performance, effectively guiding LLMs to correct all types of errors made by themselves.

7 Limitations

There are several unexplored avenues of interest, such as determining whether rules similar but not identical to the gold trees can guide LLMs in making corrections, and assessing the extent to which LLMs actually learn from examples. These will be addressed in future studies.

Acknowledgments

First of all, we would like to thank the anonymous reviewers for their valuable comments and suggestions. We are very grateful to Houquan Zhou to discuss with us and provide valuable suggestions, and Haozhe Zhou to help us with the deeper error analysis of LLM behaviour during the rebuttal period. This work was supported by National Natural Science Foundation of China (Grant No. 62306202 and 62176173).

References

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

Daniel Dakota and Sandra Kübler. 2021. What's in a span? evaluating the creativity of a span-based neural constituency parser. In *Proceedings of the Society for Computation in Linguistics 2021*, pages 323–333, Online. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023a. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508, Toronto, Canada. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. 2023b. Rarr: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

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.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural*

Language Processing and Computational Natural Language Learning, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

Jonathan K. Kummerfeld, Daniel Tse, James R. Curran, and Dan Klein. 2013. An empirical examination of challenges in Chinese parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 98–103, Sofia, Bulgaria. Association for Computational Linguistics.

Jianling Li, Meishan Zhang, Peiming Guo, Min Zhang, and Yue Zhang. 2023. LLM-enhanced self-training for cross-domain constituency parsing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8174–8185, Singapore. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

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.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506.

Vikas Raunak, Amr Sharaf, Yiren Wang, Hany Awadalla, and Arul Menezes. 2023. Leveraging GPT-4 for automatic translation post-editing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12009–12024, Singapore. 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.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2023. Detecting pretraining data from large language models. *Preprint*, arXiv:2310.16789.

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.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can LLMs learn from previous mistakes? investigating LLMs' errors to boost for reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3065–3080, Bangkok, Thailand. Association for Computational Linguistics.

Gladys Tyen, Hassan Mansoor, Peter Chen, Tony Mak, and Victor Cărbune. 2023. Llms cannot find reasoning errors, but can correct them! *arXiv preprint arXiv:2311.08516*.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

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

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang

Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Kevin Yang, Yuandong Tian, Nanyun Peng, and Dan Klein. 2022a. Re3: Generating longer stories with recursive reprompting and revision. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4393–4479, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sen Yang, Leyang Cui, Ruoxi Ning, Di Wu, and Yue Zhang. 2022b. Challenges to open-domain constituency parsing. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 112–127, Dublin, Ireland. Association for Computational Linguistics.

Houquan Zhou, Yang Hou, Zhenghua Li, Xuebin Wang, Zhefeng Wang, Xinyu Duan, and Min Zhang. 2023. How well do large language models understand syntax? an evaluation by asking natural language questions. *Preprint*, arXiv:2311.08287.

LLM Prompt

You will be given one sentence for constituency parsing. Every word that is separated by a space should be considered an independent word and have its own constituency label. Please parse the sentence with given words.
Here are some examples:
< sentence 1 > < parse tree 1 >
< sentence 2 > < parse tree 2 >
...
Input: Albany escaped embarrassingly unscathed .
Output: (S (NP (NNP Albany)) (VP (VBD escaped) (S (ADJP (RB embarrassingly) (JJ unscathed)))) (. .))

Figure 6: The prompt for few-shot learning.

Method	R	P	F
Other linearization method	65.36	75.20	69.93
Our method	**69.97**	**77.14**	**73.38**

Table 6: Results of different linearization methods.

A Appendix

A.1 Prompt for Constituency Parsing

Previous research (Bai et al., 2023; Tian et al., 2024) has demonstrated that linearizing constituency trees into sequences is effective when using LLMs for tree parsing. Therefore, following Bai et al. (2023), we represent constituency tree structures using bracket notation, such as "(S (NP (PRP He)) (VP (MD could) (RB not) (VP (VB speak))))". We conduct experiments under the five-shot setting, randomly sampling five examples from existing treebanks, as is shwon in Figure 6. Additionally, we experimentally validate the linearization format metioned in Vinyals et al. (2015), such as "(S (NP PRP) (VP MD RB (VP VB)))" by testing on PTB with GPT-4. As is shown in Table 6, our results show that the linearization method employed in this paper outperforms this alternative approach metioned in Vinyals et al. (2015).

A.2 Hint for Sentence Mismatch Correction

Following the evaluation script of EVALB, we categorize mismatch errors into two types: length mismatch and word mismatch. Length mismatch means the number of words in the predicted tree differ from the original sentence, while word mismatch indicates a word in the sentence has been altered. As is shown in Figure 7, we design specific hints for different errors. By highlighting in the prompt the need to avoid mistakes from the previous answer, we guide the LLMs to generate a new answer.

Type	Error	LLM Output	Hint
Length Mismatch	extra	(S (NP (DT That)) (VP (VBZ 's) (ADJP (RB really) (RB so) (JJ cool))))	Do not additionally add any words, especially "really"
	missing	(S (NP (DT That)) (VP (VBZ 's) (ADJP (JJ cool))))	Do not omit any words, especially "so"
	split	(S (NP (DT That)) (VP (POS ') (VBZ s) (ADJP (RB so) (JJ cool))))	" " and "s" are considered as one word and should have a common label
	combined	(S (NP (DT That's)) (ADJP (RB so) (JJ cool)))	"That" and "'s" are considered as two words and should be separated with each having its own label
Word Mismatch	modified	(S (NP (DT That's)) (ADJP (RB very) (JJ cool)))	Do not make any changes to the word, especially "so"

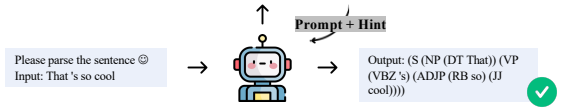


Figure 7: The process of sentence mismatch correction.

	WikiMIA	PTB	CTB	MCTB
MKP	7.13	9.59	8.95	9.53

Table 7: The MKP of different datasets with Llama-8B. A higher MKP suggests that the sentence is less likely to be included in the pre-training data of LLMs.

A.3 The Examination of the Possibility of Contamination in Test Set

To examine the possibility of contamination in test set, we use the Min K% probability (MKP) (Shi et al., 2023), as a metric to determine the extent to which the datasets are included in the pre-training data of LLMs. We follow the recommendation in Shi et al. (2023), setting K value to 20% and calculating the MKP of sentences in the three datasets under the Llama-8B model. We also report the MKP of the WikiMIA dataset for reference, which is used by Shi et al. (2023) to evaluate the identification capabilities for detecting pre-training data, consists of sentences from Wikipedia articles. Given the popularity of Wikipedia as a pre-training source, these sentences are highly likely to be part of the pre-training data for LLMs. As is shown in Table 7, the MKP of all three datasets is significantly higher than that of WikiMIA. This suggests that these datasets are unlikely to have been included in the pre-training data of Llama-8B.

A.4 Random Selection vs. Our Method

To further validate the effectiveness of our method, we also compared it with randomly selecting examples for few-shot learning on the PTB with GPT-4. As indicated in Table 8, our results significantly surpass that of random selection, demonstrating that our method can select more suitable examples for LLMs via ranking based on LCS.

Method	R	P	F
Random selection	71.47	79.25	75.16
Our method	**82.27**	**84.78**	**83.50**

Table 8: Results of random selection and our method on the PTB with GPT-4.

	R	P	F
3 rules*2=6 examples	80.75	85.08	82.85
5 rules*1=5 examples	**82.27**	**84.78**	**83.50**

Table 9: The results of different numbers of the final examples.

A.5 The Effect of Different Numbers of the Final Rules and Examples

To investigate the effect of different numbers of the final rules and examples, we add the experiment with the setting that select the top three rules with each rule providing two examples as shown in the first row of the Table 9. The second row shows the setting in our paper. Obviously, the original setting in our paper achieves better performance.

A.6 Different Searching Strategies

We also compare our method with searching examples with the most similar POS sequences. As is shown in Table 10, our method based on the error-specific processing and ranking according to the label sequence significantly surpass ranking according to the POS. The POS-based ranking method only considers information from the lowest level leaf nodes, ignoring the structured information at intermediate levels. In contrast, our proposed method takes into account the structural information and can cover four types of errors, thus performing better.

A.7 The Assignment of Weights during the LCS Calculation vs. Our Method

To investigate the effect of frequency on the selection of rules, we also carry out experiments on the assignment of weights during the LCS calculation with GPT-4 on the PTB. Specifically, we multiplie LCS by the proportion of the current rule during similarity calculation. As is shown in Table 11, the results indicate that LCS-based similarity plays a more crucial role than frequency in selecting rules.

Method	R	P	F
POS searching	78.07	84.09	80.97
Our method	**82.27**	**84.78**	**83.50**

Table 10: Results of searching based on the POS-tag and our method on the PTB with GPT-4.

	R	P	F
Assigning weights	75.97	83.35	79.49
Our method	**82.27**	**84.78**	**83.50**

Table 11: Results of assigning weights of frequency and our method on the PTB with GPT-4.

Overemphasizing frequency tends to disrupt the original ranking based on LCS.

A.8 The Distribution of Four Types of Errors across Different models

We provide the distributions of four types of errors across different models and different datasets in Table 12, Table 13 and Table 14. Span error and label error are the most and second most frequent in the parsing results of different models. Moreover, as the performance of models declines, the proportion of span errors increases, indicating that models with weaker parsing capabilities find it more challenging to correctly segment spans.

Model	Span	Label	Flatness	Deepness
Berkeley	1,386	944	2,736	2,985
LLaMa-8B	28,665	7,462	1,524	964
LLaMa-70B	12,404	5,364	3,364	1,492
GPT-3.5	7,712	3,383	4,776	1,991
GPT-4	4,944	3,228	3,791	1,254

Table 12: The overall of four types of errors made by different models on the PTB.

Model	Span	Label	Flatness	Deepness
Berkeley	1,394	1,397	1,153	1,246
LLaMa-8B	2,830	1,679	247	186
Qwen-72B	1,987	1,450	722	596
DeepSeek-v3	1,731	1,110	740	487
GPT-3.5	1,752	1,046	797	514
GPT-4	1,951	1,416	864	732

Table 13: The overall of four types of errors made by different models on the CTB.

Model	Span	Label	Flatness	Deepness
Berkeley	463	562	600	140
LLaMa-8B	11,609	2,962	493	352
LLaMa-70B	4,704	2,434	1,162	449
GPT-3.5	3,364	1,857	1,483	718
GPT-4	1,570	1,379	939	808

Table 14: The overall of four types of errors made by different models on the MCTB.