# Large Language Models Are No Longer Shallow Parsers

**Yuanhe Tian**♠♥, **Fei Xia**♥, **Yan Song**♠†
♠University of Science and Technology of China ♥University of Washington
♥{yhtian, fxia}@uw.edu ♠clksong@gmail.com

## Abstract

The development of large language models (LLMs) brings significant changes to the field of natural language processing (NLP), enabling remarkable performance in various high-level tasks, such as machine translation, question-answering, dialogue generation, etc., under end-to-end settings without requiring much training data. Meanwhile, fundamental NLP tasks, particularly syntactic parsing, are also essential for language study as well as evaluating the capability of LLMs for instruction understanding and usage. In this paper, we focus on analyzing and improving the capability of current state-of-the-art LLMs on a classic fundamental task, namely constituency parsing, which is the representative syntactic task in both linguistics and natural language processing. We observe that these LLMs are effective in shallow parsing but struggle with creating correct full parse trees. To improve the performance of LLMs on deep syntactic parsing, we propose a three-step approach that firstly prompts LLMs for chunking, then filters out low-quality chunks, and finally adds the remaining chunks to prompts to instruct LLMs for parsing, with later enhancement by chain-of-thought prompting. Experimental results on English and Chinese benchmark datasets demonstrate the effectiveness of our approach on improving LLMs' performance on constituency parsing.[1]

## 1 Introduction

Recently, the development of large language models (LLMs) (Achiam et al., 2023; Touvron et al., 2023a,b; Taori et al., 2023; Chiang et al., 2023) significantly reshapes the field of natural language processing (NLP), where they achieve impressive performance on a variety of NLP applications such as question-answering and dialogue interaction. Nevertheless, in the era of LLM, fundamental NLP tasks that involve linguistic analysis still play essential roles in the field of NLP (Mahowald et al., 2023; Blevins et al., 2023). For example, syntax parsing provides structural understanding of sentences and thus is valuable in evaluating the capability of LLMs to understand and use instructions (McCoy et al., 2023; Muñoz-Ortiz et al., 2023; Blevins et al., 2023).

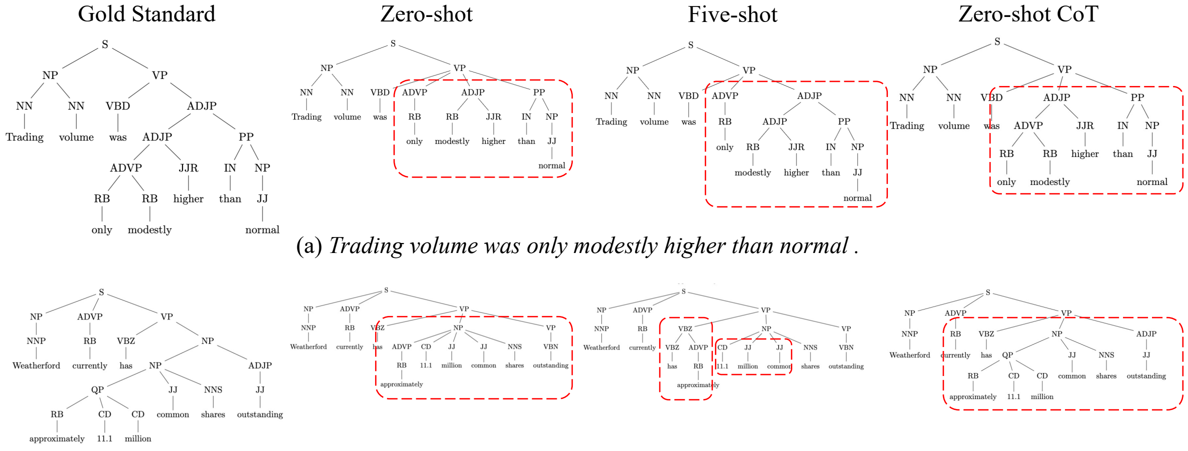
In general, syntactic parsing is an essential NLP task that has been studied for decades (Collins, 1997; Glaysher and Moldovan, 2006; Cai et al., 2009; Liu and Zhang, 2017; Kitaev and Klein, 2018; Tian et al., 2020b, 2022; Gu et al., 2022) and has two widely used conventional variants with phrase-structure and dependency grammar, of which the former drives constituency parsing that provides deep structural analysis of sentences with informative phrase and function markups. Recent studies (Bai et al., 2023; Lin et al., 2023) reveal that LLMs are ineffective in such parsing task, and we also confirm the fact in Figure 1 with parse trees generated by GPT-4 (Achiam et al., 2023) under various prompts, where they all look very different from human-crafted gold-standard trees, indicating syntactic parsing is challenging to LLMs. Although some studies (Blevins et al., 2023; Bai et al., 2023; Li et al., 2023) design prompts to instruct LLMs for better parsing, it is still limited in performing one-step (i.e., end-to-end) parsing, which does not fully benefit from the advantages brought by the strengths of LLMs.

In this paper, we explore the effectiveness of directly using state-of-the-art LLMs for constituency parsing, and find that they generate parse trees that are shallow and with relatively low recalls on retrieving correct text spans, while in the meantime LLMs show relatively better performance in chunking, especially in identifying short chunks. Therefore, based on our analyses and motivated by existing studies that decompose complex tasks into several steps to improve LLM's performance

[†] Corresponding author.

[1] Resources of this paper are available at https://github.com/synlp/LLMPar.

*Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7131–7142
August 11-16, 2024 ©2024 Association for Computational Linguistics



(a) *Trading volume was only modestly higher than normal .*

(b) *Weatherford currently has approximately 11.1 million common shares outstanding .*

Figure 1: Two example sentences with gold-standard parse trees followed by parse trees generated by GPT-4 with different types of prompts–namely, zero-shot, five-shot, and standard zero-shot CoT (i.e., adding "*let's think step-by-step*" to the prompt). The parts where the LLMs do not match the gold standard are marked by red boxes.

(Wei et al., 2022; Valmeekam et al., 2023; Zhao et al., 2023; Guan et al., 2023; Ranaldi and Zanzotto, 2023), we propose a three-step approach to improve LLMs for parsing. The approach firstly prompts LLMs to extract chunks in the text, which is considered partial parsing and is demonstrated to be an essential step for full parsing (Gu et al., 2022; Yang and Tu, 2023). Then, our approach filters out long chunks that are more likely to carry noise than the short ones. Finally, it uses the remaining chunks as conditions in the prompts to instruct LLM for parsing, allowing LLMs to directly use the chunks to generate the parse tree of a sentence. We validate our approach on several benchmark datasets, including English Penn Treebank 3 (PTB) (Marcus et al., 1993), Chinese Penn Treebank 5 (CTB5) (Xue et al., 2005), and Genia (Tateisi et al., 2005), where the results prove the effectiveness of our approach over existing LLMs used directly for constituency parsing.

The contributions of this paper are summarized as follows. First, we provide comprehensive analyses of existing state-of-the-art LLMs on constituency parsing. Second, we propose a three-step approach that involves chunking, filtering, and parsing, to improve LLMs for constituency parsing. Third, the experiment results and further analyses demonstrate the effectiveness of the proposed approach for constituency parsing.

## 2 Analysis of Current LLMs for Parsing

In order to understand the issues and improvement directions in existing LLMs for parsing, we analyze the performance of LLMs, which can be divided into two groups. The first group includes GPT-3.5 and GPT-4, which are accessible only through online API. The second group contains publicly available LLMs; namely, the 7B and 65B versions of LLaMA-2 for English (Touvron et al., 2023b), and the 7B and 13B versions of Alpaca for Chinese (Cui et al., 2023) (we denote these English and Chinese LLMs as LLaMA for the sake of convenience).[2]

| Datasets | | Sent. # | Token # | ASL |
|---|---|---|---|---|
| CTB5 | Train | 17K | 478K | 27.4 |
| | Dev | 350 | 7K | 19.5 |
| | Test | 348 | 8K | 23.0 |
| PTB | Train | 40K | 950K | 23.9 |
| | Dev | 2K | 40K | 23.6 |
| | Test | 2K | 57K | 23.5 |
| Genia (Full) | | 17K | 446K | 26.2 |

Table 1: The statistics of all experimental datasets (with splits) in terms of sentence and token numbers, and average word-based sentence length (ASL).

We use these LLMs to parse the test sets of PTB, CTB5, and Genia, where PTB and CTB5 are general domain English and Chinese datasets, respectively, and Genia is an English medical dataset. The statistics of the datasets are reported in Table 1. To perform efficient evaluation, we randomly

[2] We obtain English "*Llama2-chat-hf*" from `https://huggingface.co/meta-llama` and the Chinese "*Chinese-Alpaca-2*" from `https://github.com/ymcui/Chinese-LLaMA-Alpaca-2`, respectively.

**Prompt Template**:
*The following is a sentence that has already been tokenized. Words are separated by white spaces. Please parse the sentence with given words.*
<few-shot examples>
*[sentence]:* <sentence>
*[parse tree]:*

**Input**:
*Is this what the home builders want ?*

**Output**:
*(SQ (VBZ Is) (NP (DT this)) (SBAR (WHNP (WP what)) (S (NP (DT the) (NN home) (NNS builders)) (VP (VBP want)))) (. ?))*

Table 2: The prompt with example input and output used for constituency parsing. "<few-shot examples>" and "<sentence>" are placeholders for few-shot demonstration examples and the test sentence, respectively.

sample 2,000 sentences from Genia dataset and run experiments on it.

Generally, there are two approaches to linearizing a parse tree into a string to facilitate LLM processing (Vinyals et al., 2015; Liu and Zhang, 2017). The first is the bracket-based approach, which uses brackets to represent the structure of a parse tree, such as "*(S (NP he) (VP jumps))*". The second is the transition-based approach that utilizes the sequence of transition actions to construct the parse tree, such as "*SHIFT he; NP; SHIFT jumps; VP; REDUCE; S*". According to Bai et al. (2023), when working with LLMs, the bracket-based approach outperforms the transition-based one; therefore, we adopt the bracket-based representation in our study.

Motivated by Bai et al. (2023), we design prompts (see Table 2)[3] to instruct LLMs to generate the parse trees of sentences under the five-shot setting. The demonstration examples are (*sentence*, *parse tree*) pairs and can be obtained by various methods such as random sampling from a treebank or automatic generation with rules. When LLMs produce invalid parse trees with mismatched left or right brackets, we add a post-processing step to ensure bracket pairing by adding the necessary left or right brackets to the parse tree until they are matched. Finally, LLM outputs are evaluated with the standard evaluation toolkit EVALB[4].

| Dataset | Models | P | R | F |
|---|---|---|---|---|
| PTB | *Fried et al. (2019) | - | - | 95.71 |
| | *Yang and Tu (2023) | 96.64 | 96.34 | 96.48 |
| | †LLaMA-7B | - | - | 8.82 |
| | †LLaMA-65B | - | - | 26.85 |
| | †GPT-3.5 | - | - | 66.41 |
| | †GPT-4 | - | - | 73.28 |
| | LLaMA-7B | 15.76 | 5.84 | 8.52 |
| | LLaMA-65B | 40.39 | 18.54 | 25.41 |
| | GPT-3.5 | 79.38 | 57.19 | 66.48 |
| | GPT-4 | 87.37 | 63.23 | 73.36 |
| CTB5 | *Fried et al. (2019) | - | - | 92.14 |
| | *Yang and Deng (2020) | 93.80 | 93.40 | 93.59 |
| | *Yang and Tu (2023) | 92.83 | 91.97 | 92.41 |
| | LLaMA-7B | 12.58 | 6.49 | 8.56 |
| | LLaMA-13B | 18.40 | 6.85 | 9.98 |
| | GPT-3.5 | 73.35 | 50.46 | 59.79 |
| | GPT-4 | 80.47 | 55.84 | 65.93 |
| Genia | *Fried et al. (2019) | - | - | 87.54 |
| | *Tian et al. (2020c) | - | - | 87.58 |
| | LLaMA-7B | 6.24 | 2.58 | 3.65 |
| | LLaMA-65B | 32.84 | 15.42 | 20.99 |
| | GPT-3.5 | 69.53 | 48.36 | 57.04 |
| | GPT-4 | 73.27 | 53.06 | 61.55 |

Table 3: Parsing performance (in labeled precision (P), recall (R), and F1 scores) of different models on PTB, CTB5, and Genia. Models fine-tuned on the training data are marked by "*"; LLM results from Bai et al. (2023) under the five-shot setting are marked by "†"; the rest are our own results when running LLMs under the five-shot setting.

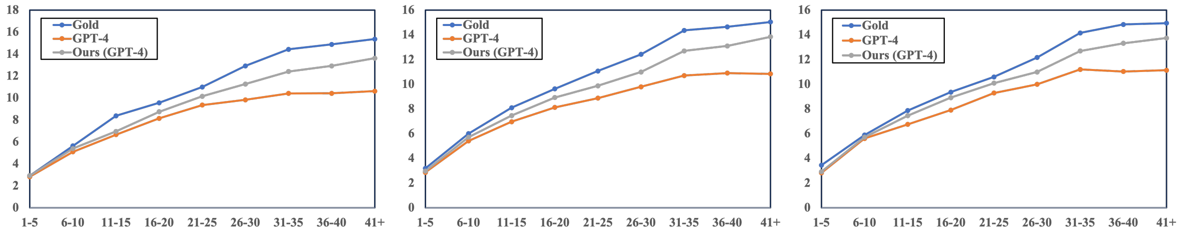
### 2.1 Parsing Results

We report parsing performance by labeled precision, recall and F1 scores in Table 3. For each corpus, the top part (above the horizontal lines) are the results reported in previous studies, and the bottom part are ours when running the LLMs under the five-shot setting. It is shown that the performance of LLMs is much lower than the parsers (marked by "*") that are based on BERT (Devlin et al., 2019) and fine-tuned on the training set. In particular, publicly available LLaMA models with various sizes are unable to produce valid parse trees, confirming the observations in Bai et al. (2023). Moreover, both GPT-3.5 and GPT-4 tend to produce parse trees that are shallower than the gold-standard trees (illustrated in Figure 1) and their precision is higher than the recall.

In addition, we use Figure 2 to show the average depth of gold-standard (i.e., the blue curve) and GPT4-generated (i.e., the orange curve) parse trees with respect to sentence length. On average, the LLM-generated trees are shallower than the gold-

[3] For Chinese datasets, we translate the English prompts into Chinese to instruct the LLMs.

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



(a) PTB (b) CTB (c) Genia

Figure 2: The average depth (Y-axis) of gold-standard and LLM-produced parse trees with respect to sentence length (X-axis). Parse trees in gold standard and produced by the GPT-4 baseline are represented in blue and orange colors, respectively, where the ones from our approach (see Section 3) are marked in gray color.

**Prompt Template**:
*The following is a sentence that has already been tokenized. Words are separated by white spaces. Please find all valid text spans (chunks) in the following sentence based on the phrase structure grammar.*
<few-shot examples>
*[sentence]:* <sentence>
*[chunking]:*

**Input**:
*The Dow was down about 35 points*

**Output**:
*(NP the Dow), (NP 35 points), (NP about 35 points), (PP down about 35 points), (VP was down about 35 points)*

Table 4: The example prompt for the chunking task. The top part is the prompt template, where "<few-shot examples>" and "<sentence>" are placeholders for few-shot demonstration examples and the test sentence, respectively. The bottom part shows an example input test sentence and the output produced by LLMs.

standard ones. Interestingly, while the depth of the gold standard parse tree increases roughly linearly with the increase of sentence length, that of the LLM-generated trees tends to plateau for long sentences, indicating that LLMs have difficulties in generating deep structures.

Overall, our analysis of parsing results show that LLMs are ineffective in producing high-quality full parse trees.

### 2.2 Chunking Results

Previous studies (Wei et al., 2022; Valmeekam et al., 2023; Zhao et al., 2023; Guan et al., 2023; Ranaldi and Zanzotto, 2023) have shown that decomposing complex tasks into several steps is able to improve the performance of LLMs. In this study, we want to test whether the same approach is applicable to parsing.

To decompose the parsing task, we first examine how well LLMs perform at chunking. Here, the term "*chunk*" refers to a text span covered by an internal node in a parse tree. Unlike the standard chunking task where the chunks in the output should not overlap, the goal of the chunking task in this study is to identify all the chunks in a sentence. Our chunking experiments use the same datasets, LLMs and five-shot setting as in our parsing experiments. The only difference is the prompt for parsing is replaced by the prompt in Table 4. For demonstration examples, we extract chunks with various lengths from treebanks.

The chunking results are in Table 5, which also includes the results of the supervised approach with the BERT-base encoder. Not surprisingly, LLMs achieve better performance on chunking than parsing (Table 5 vs. Table 3). Furthermore, as shown in Figure 3, LLMs are better at identifying shorter chunks than longer ones.[5]

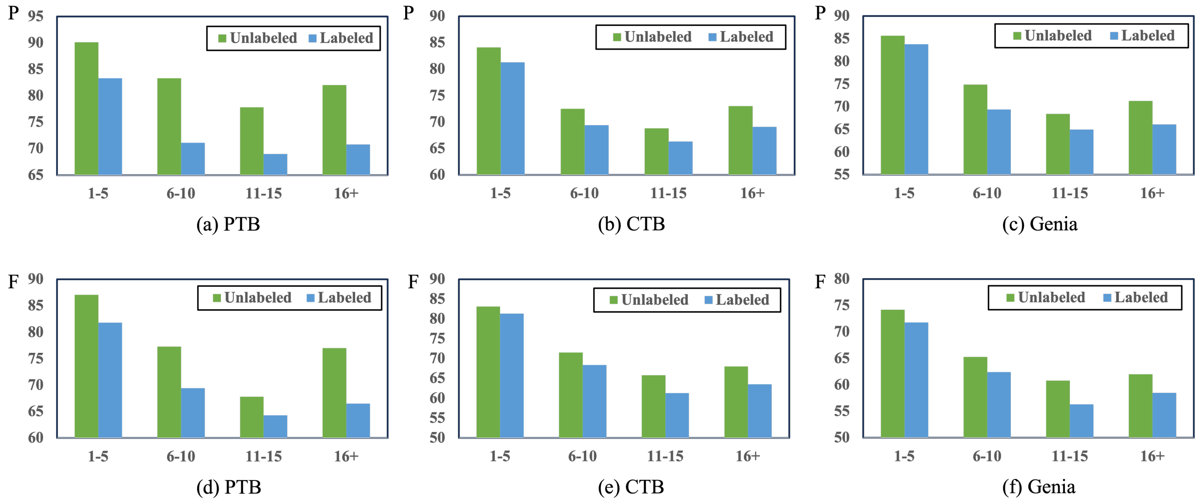
## 3 Adapting LLMs for Parsing

Based on experimental results on parsing and chunking, we hypothesize that LLMs' performance on parsing can be improved by decomposing the parsing task into two steps: first, LLMs are asked to chunk the input sentence; second, LLMs are asked to produce a parse tree based on a list of chunks. To alleviate the effect of error propagation in a multi-step system like this, after the chunking step, we filter out low-quality chunks and send the remaining ones to the parsing step, as illustrated in Figure 4. As LLMs are better at identifying short chunks than the long ones, the filtering step will simply filter by the chunk length. We also propose chain-of-thought (CoT) prompting (Wei et al., 2022) in

[5] Figure 3 shows the F-scores for chunks with length 16+ are higher than chunks with 11-15 words. This is due to the fact that our calculation includes chunks that cover the entire sentences; those chunks tend to have 16+ words and can be identified by LLMs pretty accurately.

| Dataset | Models | Unlabeled |  |  | Labeled |  |  |
|---|---|---|---|---|---|---|---|
|  |  | P | R | F | P | R | F |
| PTB | *BERT-base | 94.43 | 94.35 | 94.88 | 93.20 | 93.47 | 93.33 |
|  | LLaMA-7B | 72.04 | 58.40 | 64.51 | 70.31 | 52.07 | 59.83 |
|  | LLaMA-65B | 79.26 | 62.74 | 71.27 | 75.54 | 58.91 | 67.42 |
|  | GPT-3.5 | 84.87 | 73.35 | 78.70 | 80.42 | 69.68 | 74.67 |
|  | GPT-4 | 88.54 | 78.35 | 83.14 | 84.53 | 73.22 | 78.47 |
| CTB5 | *BERT-base | 92.26 | 92.60 | 92.43 | 90.85 | 91.10 | 90.97 |
|  | LLaMA-7B | 66.53 | 50.32 | 57.30 | 63.41 | 47.89 | 54.56 |
|  | LLaMA-13B | 68.46 | 52.17 | 59.21 | 65.14 | 48.04 | 55.30 |
|  | GPT-3.5 | 79.64 | 65.28 | 71.74 | 75.25 | 61.09 | 67.43 |
|  | GPT-4 | 81.53 | 70.30 | 75.50 | 77.32 | 66.49 | 71.50 |
| Genia | *BERT-base | 87.10 | 87.53 | 87.31 | 83.64 | 83.79 | 83.71 |
|  | LLaMA-7B | 52.76 | 35.78 | 42.64 | 49.24 | 33.90 | 40.15 |
|  | LLaMA-65B | 57.80 | 40.72 | 47.78 | 53.57 | 36.04 | 43.09 |
|  | GPT-3.5 | 79.47 | 58.43 | 67.34 | 77.79 | 55.82 | 64.99 |
|  | GPT-4 | 81.46 | 63.95 | 71.65 | 79.74 | 61.49 | 69.43 |

Table 5: The performance (in unlabeled and labeled F1 scores) of the chunking task. Models tuned on the training data are marked by "*".



(a) PTB (b) CTB (c) Genia

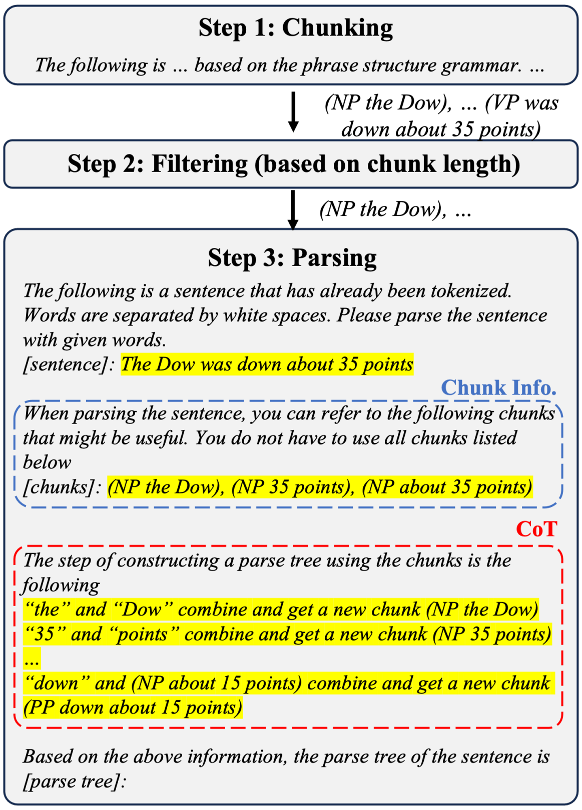
(d) PTB (e) CTB (f) Genia

Figure 3: The precision (a)-(c) and F1 scores (d)-(f) of GPT-4 produced chunks with respect to the chunk length. X-axis is the range of chunk length, and y-axis is the F1 scores of the chunks of the length in that range.

the third step for further enhancement.

### 3.1 Three-step Parsing with LLMs

The details of our parsing procedure are illustrated as follows. In Step 1, we prompt LLMs (e.g., using the one presented in Table 4) to annotate chunks in various word-based lengths following the five-shot setting. In Step 2, we filter out chunks whose word-based length is larger than a threshold. We set that threshold to be 5, based on the results from Figure 3. In Step 3, we utilize the chunks as knowledge to enhance the performance of natural language processing models, which is demonstrated to be effective in existing studies (Song et al., 2017; Zhou and Zhao, 2019; Zhang et al., 2019a,b; Yang et al., 2019; Tian et al., 2020a; Qin et al., 2021; Ouyang et al., 2022; Song, 2022). Specifically, we add the chunks to the prompt as conditions to instruct LLMs for parsing. In this step, we do not want LLMs to treat the given chunks as hard constraints when producing the parse trees; thus, we ask LLMs to refer to the chunks without having to use all of them. This way, LLMs are allowed to ignore


**Step 1: Chunking**

*The following is ... based on the phrase structure grammar. ...*

*(NP the Dow), ... (VP was down about 35 points)*

**Step 2: Filtering (based on chunk length)**

*(NP the Dow), ...*

**Step 3: Parsing**

*The following is a sentence that has already been tokenized. Words are separated by white spaces. Please parse the sentence with given words.*
*[sentence]: The Dow was down about 35 points*

**Chunk Info.**

*When parsing the sentence, you can refer to the following chunks that might be useful. You do not have to use all chunks listed below*
*[chunks]: (NP the Dow), (NP 35 points), (NP about 35 points)*

**CoT**

*The step of constructing a parse tree using the chunks is the following*
*"the" and "Dow" combine and get a new chunk (NP the Dow)*
*"35" and "points" combine and get a new chunk (NP 35 points)*
*...*
*"down" and (NP about 15 points) combine and get a new chunk (PP down about 15 points)*

*Based on the above information, the parse tree of the sentence is [parse tree]:*

Figure 4: Workflow and example prompts used in our three-step approach for parsing. We first use the prompt (e.g., the one in Table 4) to extract chunks; then filter the chunks based on their length (this example filters out chunks whose length is higher than 3); and finally perform parsing. The yellow background highlights the example; the blue dashed box presents the prompt to leverage the chunks from the second step; the red dashed box shows the CoT prompt that illustrates the instructions for final parsing.

the errors that could exist in the provided chunks, alleviating the impact of error propagation.

### 3.2 CoT for Parsing

In the aforementioned third step, to provide more detailed instructions for LLMs to leverage chunks for parsing, we perform CoT prompting. Specifically, to provide CoT demonstration, we automatically obtain step-by-step instructions from gold-standard parse trees through the following process. For each internal node $N$ in the parse tree, we find all its children nodes $C_1 \cdots C_M$ and the corresponding text spans $s_1 \cdots s_M$. Then we add "$s_1, s_2, \cdots, s_M$ *are combined into a new text span* $s$" (where $s$ is the concatenation of $s_1 \cdots s_M$) into the demonstration, which serves as a part of the prompt. This process instructs LLMs to leverage chunk information when processing each test instance. In practice, $s_1, s_2, \cdots, s_M$ include single words; we only add the case where $s_1, s_2, \cdots, s_M$ appear in the filtered chunks from Step 2.

Let us use the example in Figure 4 to demonstrate the process. Here, the provided chunks are "*(NP the Dow)*", "*(NP 35 points)*", and "*(NP the 35 points)*". In the gold-standard parse tree, the PP node that covers "*down about 15 points*" has two children, namely, the single word "*down*" and an NP phrase "*about 35 points*". Since the phrase "*about 35 points*" appears in the chunk list, we combine "*down*" and "*(NP about 35 points)*" and get a new chunk "*(PP down about 15 points)*" in the CoT part. In contrast, for the VP node that covers "*was down about 15 points*", we do not include it in CoT since one of its children, the phrase "*down about 15 points*" does not appear in the chunk list.

## 4 Analysis

### 4.1 Main Results

We evaluate our approach on the benchmark datasets with five-shot learning, and the results are in Table 6. For Step 3, we show the results with or without CoT. There are two main observations. First, CoT improves system performance across the board, which confirms the benefits of providing step-by-step guidance on how chunks should be used during the parsing step.

Second, compared with the results of directly using LLMs for parsing in Table 3, our approach, with or without CoT, achieves significant improvements on all datasets for all four LLMs. Particularly, the improvements on recall are remarkable, which indicates that our approach is able to prompt LLMs to correctly identify more chunks and include them in parse trees. A possible explanation for the improvement is that, by separating chunking from parsing, the LLMs in Step 1 are able to focus on the shallow parsing, which they are good at. Thus, this step allows more correct chunks to be identified. In the parsing step, LLMs pay more attention to combining these chunks to construct a valid parse tree. In addition, we alleviate the risk of the error propagation by filtering out long chunks and asking LLMs to treat the given chunks as soft constraints. All these factors may contribute to the better performance of the 3-step approach over the 1-step direct parsing approach.

Furthermore, as shown in Figure 2, on average, the parse trees generated by our approach are deeper than the ones generated by the 1-step approach and the curves for our approach are similar

| Dataset | Models | P | R | F1 |
|---|---|---|---|---|
| PTB | LLaMA-7B | 33.58 | 25.74 | 29.14 |
| | LLaMA-7B + CoT | 39.39 | 32.85 | 35.82 |
| | LLaMA-65B | 49.04 | 38.10 | 42.88 |
| | LLaMA-65B + CoT | 55.46 | 48.06 | 51.50 |
| | GPT-3.5 | 75.09 | 60.31 | 76.00 |
| | GPT-3.5 + CoT | 84.86 | 72.48 | 78.18 |
| | GPT-4 | 86.27 | 73.36 | 79.29 |
| | GPT-4 + CoT | **89.04** | **77.74** | **83.00** |
| CTB5 | LLaMA-7B | 31.39 | 24.15 | 27.30 |
| | LLaMA-7B + CoT | 37.09 | 30.16 | 33.27 |
| | LLaMA-13B | 37.56 | 28.49 | 32.40 |
| | LLaMA-13B + CoT | 44.53 | 36.41 | 40.06 |
| | GPT-3.5 | 79.50 | 70.84 | 74.92 |
| | GPT-3.5 + CoT | 83.83 | 73.11 | 78.10 |
| | GPT-4 | 85.14 | 74.05 | 79.21 |
| | GPT-4 + CoT | **87.53** | **76.64** | **81.72** |
| Genia | LLaMA-7B | 26.59 | 19.47 | 22.47 |
| | LLaMA-7B + CoT | 28.95 | 22.04 | 25.03 |
| | LLaMA-65B | 40.11 | 32.04 | 35.62 |
| | LLaMA-65B + CoT | 44.39 | 35.82 | 39.64 |
| | GPT-3.5 | 73.58 | 63.20 | 67.99 |
| | GPT-3.5 + CoT | 74.95 | 66.23 | 70.32 |
| | GPT-4 | 76.70 | 66.65 | 71.32 |
| | GPT-4 + CoT | **79.27** | **68.80** | **73.66** |

Table 6: The results of our approach with different LLMs on PTB, CTB5, and Genia. "+CoT" means that the CoT is used in our approach. Note that the corresponding baseline results are reported in Table 3.

| Dataset | Models | UC | UCC |
|---|---|---|---|
| PTB | LLaMA-65B | 82.38 | 85.37 |
| | GPT-4 | 89.92 | 92.02 |
| CTB5 | LLaMA-13B | 76.36 | 79.78 |
| | GPT-4 | 87.64 | 89.89 |
| Genia | LLaMA-65B | 78.26 | 80.31 |
| | GPT-4 | 85.70 | 87.83 |

Table 7: Comparison of chunk usage in Step 3 by different models. "UC" and "UCC" are the percentages of used and used correct chunks, as defined in Section 4.2.

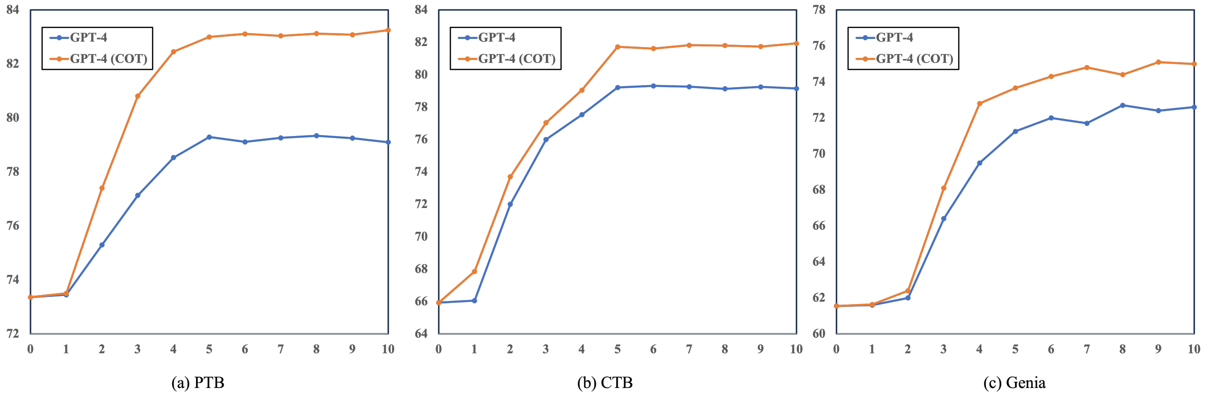
| Dataset | Models | P | R | F |
|---|---|---|---|---|
| PTB | LLaMA-65B | 48.20 | 36.87 | 41.78 |
| | GPT-4 | 85.21 | 70.46 | 77.13 |
| CTB5 | LLaMA-13B | 37.01 | 28.11 | 31.95 |
| | GPT-4 | 78.39 | 69.01 | 73.40 |
| Genia | LLaMA-65B | 39.82 | 31.67 | 35.28 |
| | GPT-4 | 76.42 | 66.29 | 70.99 |

Table 8: Performance of LLMs without CoT when they are explicitly prompted to use all the chunks from step 2 for parsing, which is compared to the results of our approach in Table 6 that does not use all chunks.

to the ones based on gold-standard parse trees. This indicates that LLMs are no longer shallow parsers, when used appropriately.

## 4.2 Effect of Chunking

In Step 3, LLMs are given a chunk list and are instructed to refer to the list but do not have to use all the chunks in the list when producing the parse tree. We want to know how many chunks in the list are actually used by LLMs. Let $A$ be the set of chunks that appear in the parse tree produced by an LLM. Let $B$ be the set of chunks provided to the LLM in Step 3's prompt. Let $B1$ be the set of chunks in $B$ that appear in the gold-standard parse tree for the sentence. We compute two metrics: the first one, UC (the percentage of "Used Chunks"), is the percentage of the provided chunks that are used by the LLM; that is, $UC =\mid A \cap B \mid / \mid B \mid$. The second metric, UCC (percentage of "Used Correct Chunks"), is the percentage of the gold chunks that are used by the LLM; i.e., $UCC =\mid$



(a) PTB

(b) CTB

(c) Genia

Figure 5: Parsing performance (in labeled F-1 score) of GPT-4 with or without CoT on the test sets of the three treebanks. The X-axis shows the value of the chunk length threshold, which is used to filter out chunks in Step 2. When x=0, we show the performance of the baseline system (i.e., 1-step parsing) which does not use chunks.

$A \cap B1 \mid / \mid B1 \mid$. Table 7 show the results of two representative LLMs; namely, LLaMA and GPT-4 (one open-source and one closed-source LLM). The table indicates that in Step 3 LLMs use the majority of chunks provided to them, and the percentage of chunks used is even higher among the gold chunks in the provided chunk list.

What would happen if we require LLMs to use all the chunks provided to them? The results under the no-CoT setting are in Table 8, which are worse than the corresponding rows in Table 6. In other words, forcing LLMs to use all the chunks hurts the parsing performance; therefore, the provided chunk list should indeed be treated as a reference, not a hard constraint.

Finally, we analyze the effect of the chunk length threshold used in Step 2. Figure 5 shows the parsing performance with different threshold values. The figure shows that the system performance improves a lot when the threshold value increases from 1 all the way to 5; afterwards, parsing performance quickly plateaus as bigger chunks identified in Step 1 are more error-prone and thus offer limited help.

### 4.3 Case Study

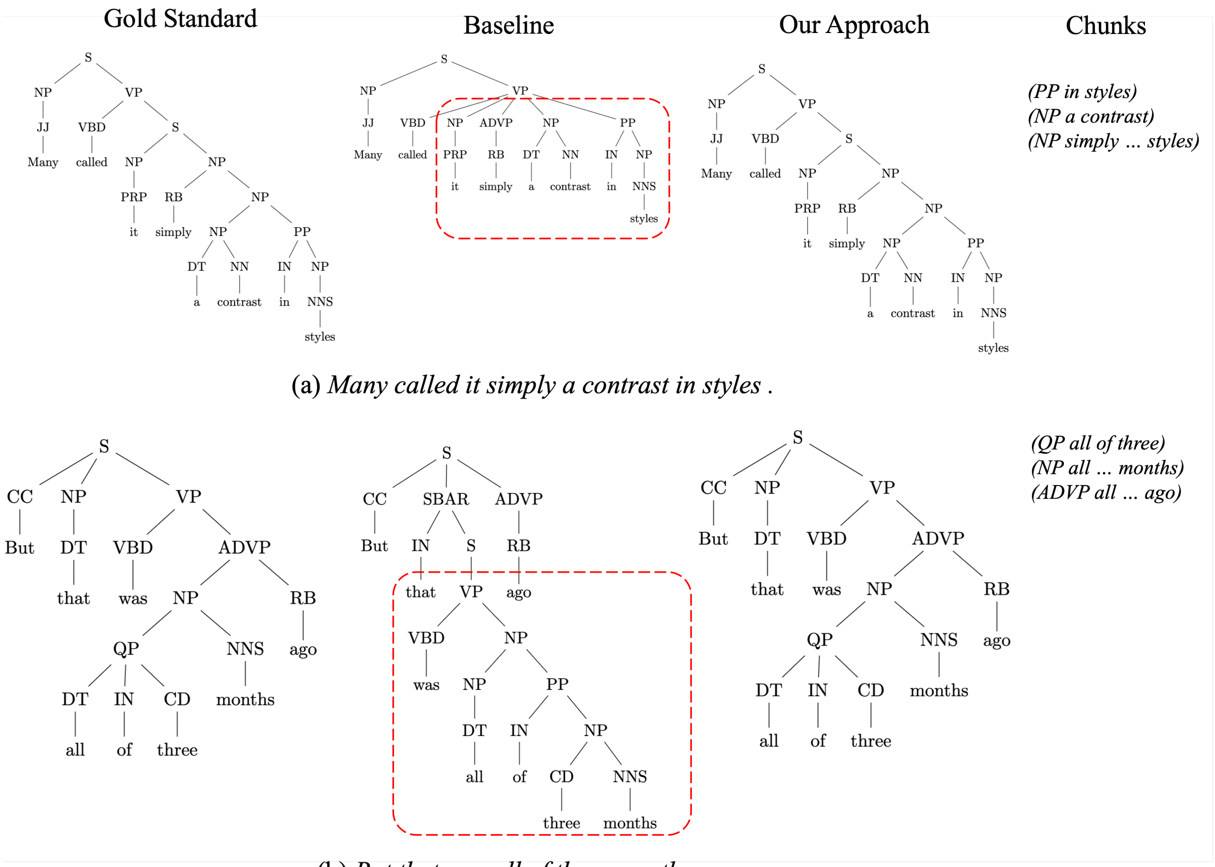
As previously shown, our 3-step approach outperforms the baseline system where LLMs produce the parse tree in one step ("the GPT-4 baseline"). To better understand where the gain comes from, we look at parse trees produced by these approaches and two examples are in Figure 6. In the first example, in the gold-standard parse tree, the VP "*called it simply a contrast in styles*" has a deep syntactic structure, where as the GPT-4 baseline simply combines all the words together to construct a flat VP. In our system, GPT-4 identifies chunks such as "*(PP in styles)*" and "*NP simply a contrast in styles*" in Step 1. When they are provided to GPT-4 in Step 3, GPT-4 is able to produce a parse tree with much deeper structure.

In the second example, the GPT-4 baseline fails to combine the word "*ago*" and the NP phrase "*all of three months*" to form an ADVP. In our approach, GPT-4 produces chunks such as "*all of three months ago*" in Step 1. When they are passed to GPT-4 in Step 3, GPT-4 is able to build the correct parse tree.

## 5 Related Work

Constituency parsing is a fundamental NLP task that is generally performed via chart- or transition-based approaches (Collins, 1997; Sagae and Lavie, 2005; Glaysher and Moldovan, 2006; Song and Kit, 2009; Dyer et al., 2016; Liu and Zhang, 2017; Stern et al., 2017; Kitaev and Klein, 2018; Suzuki et al., 2018; Fried et al., 2019; Zhou and Zhao, 2019; Tian et al., 2020c; Yang and Deng, 2020; Zhang et al., 2020; Nguyen et al., 2021; Yang et al., 2022; Gu et al., 2022), where extra resources are leveraged to improve the parsing performance (Mrini et al., 2019; Maveli and Cohen, 2021; Yang et al., 2022; He and Choi, 2023; Shayegh et al., 2023).

Recently, LLMs have achieved remarkable success in many NLP tasks, which motivates researchers to apply them to parsing. Existing approaches that leverage LLMs for parsing generally follow the sequence-to-sequence settings (Vinyals et al., 2015; Suzuki et al., 2018), where the goal of LLMs is to produce the parse tree in the bracket format. The LLM-based approaches fine-tune LLMs on the training data or design prompts to instruct



(a) *Many called it simply a contrast in styles .*

(b) *But that was all of three months ago .*

Figure 6: Two examples from the test set of PTB. The first three columns show the gold-standard parse tree; the parse tree generated by the GPT-4 baseline and our approach, respectively; the last column lists some chunks produced in Step 1 of our approach. The red boxes mark the parts in the generated parse tree that do not match the gold standard. The parse trees by our approach for these sentences happen to be identical to the gold-standard trees.

LLMs to generate valid parse trees (Bai et al., 2023; Li et al., 2023). For example, Bai et al. (2023) directly train LLaMA (Touvron et al., 2023a) on the training set of PTB and show inferior performance than the conventional chart- and transition-based approaches that leverage small pre-trained language models such as BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019). They also perform zero-shot and few-shot parsing but the results are unsatisfactory, which confirms the ineffectiveness of LLMs for parsing. There are studies that select appropriate training data (Li et al., 2023) to further improve constituency parsing with LLM.

In this study, we propose to break parsing into three steps: chunking, filtering chunks, and parsing with chunks, and demonstrate this 3-step approach improve system performance significantly.

## 6 Conclusion

In this paper, we apply LLMs to parsing and observe that the current state-of-the-art LLMs are shallow parsers in that they are effective in performing simple syntactic tasks such as chunking but are ineffective at conducting full constituency parsing due to their auto-regressive nature of text modeling approach. Given this, we propose to break parsing into three steps: chunking, filtering, and parsing with a chunk list. Experiments on English, Chinese, and domain benchmark datasets show that the 3-step approach enables LLMs to produce deeper and better parse trees and thus they are no longer shallow parsers.

This study demonstrates the benefits of decomposing a complex task (e.g., parsing) into subtasks and the importance of including additional mechanisms (e.g., filtering chunks in Step 2 and treating given chunks as soft constraints in Step 3) to alleviate the risk of the error propagation. We plan to apply this idea to other NLP tasks in the future.

## Acknowledgements

This work is supported by the National Key Research and Development Program of China under the grant (2023YFC3303800).

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Xuefeng Bai, Jialong Wu, Yulong Chen, Zhongqing Wang, and Yue Zhang. 2023. Constituency Parsing using LLMs. *arXiv preprint arXiv:2310.19462*.

Terra Blevins, Hila Gonen, and Luke Zettlemoyer. 2023. Prompting Language Models for Linguistic Structure. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6649–6663, Toronto, Canada.

Dongfeng Cai, Yonghua Hu, Xuelei Miao, and Yan Song. 2009. Dependency Grammar Based English Subject-Verb Agreement Evaluation. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*, pages 63–71, Hong Kong.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. *GitHub repository*.

Michael Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain.

Yiming Cui, Ziqing Yang, and Xin Yao. 2023. Efficient and Effective Text Encoding for Chinese LLaMA and Alpaca. *arXiv preprint arXiv:2304.08177*.

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

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent Neural Network Grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California.

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.

Elliot Glaysher and Dan Moldovan. 2006. Speeding Up Full Syntactic Parsing by Leveraging Partial Parsing Decisions. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 295–300, Sydney, Australia.

Xiaotao Gu, Yikang Shen, Jiaming Shen, Jingbo Shang, and Jiawei Han. 2022. Phrase-aware Unsupervised Constituency Parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6406–6415, Dublin, Ireland.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging Pretrained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. *arXiv preprint arXiv:2305.14909*.

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.

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.

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.

Boda Lin, Xinyi Zhou, Binghao Tang, Xiaocheng Gong, and Si Li. 2023. ChatGPT is a Potential Zero-Shot Dependency Parser. *arXiv preprint arXiv:2310.16654*.

Jiangming Liu and Yue Zhang. 2017. In-Order Transition-based Constituent Parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Kyle Mahowald, Anna A Ivanova, Idan A Blank, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. 2023. Dissociating language and thought in large language models: a cognitive perspective. *arXiv preprint arXiv:2301.06627*.

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.

Nickil Maveli and Shay B Cohen. 2021. Co-training an unsupervised constituency parser with weak supervision. *arXiv preprint arXiv:2110.02283*.

R. Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. 2023. How Much Do Language Models Copy From Their Training

Data? Evaluating Linguistic Novelty in Text Generation Using RAVEN. *Transactions of the Association for Computational Linguistics*, 11:652–670.

Khalil Mrini, Franck Dernoncourt, Trung Bui, Walter Chang, and Ndapa Nakashole. 2019. Rethinking Self-Attention: An Interpretable Self-Attentive Encoder-Decoder Parser. *arXiv preprint arXiv:1911.03875*.

Alberto Muñoz-Ortiz, Carlos Gómez-Rodríguez, and David Vilares. 2023. Contrasting linguistic patterns in human and llm-generated text. *arXiv preprint arXiv:2308.09067*.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq Joty, and Xiaoli Li. 2021. A Conditional Splitting Framework for Efficient Constituency Parsing. 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 5795–5807, Online.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training Language Models to Follow Instructions with Human Feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Han Qin, Yuanhe Tian, and Yan Song. 2021. Relation Extraction with Word Graphs from N-grams. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2860–2868, Online and Punta Cana, Dominican Republic.

Leonardo Ranaldi and Fabio Massimo Zanzotto. 2023. Empowering multi-step reasoning across languages via tree-of-thoughts. *arXiv preprint arXiv:2311.08097*.

Kenji Sagae and Alon Lavie. 2005. A Classifier-Based Parser with Linear Run-Time Complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132, Vancouver, British Columbia.

Behzad Shayegh, Yanshuai Cao, Xiaodan Zhu, Jackie CK Cheung, and Lili Mou. 2023. Ensemble Distillation for Unsupervised Constituency Parsing. *arXiv preprint arXiv:2310.01717*.

Yan Song. 2022. Chinese Couplet Generation with Syntactic Information. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6436–6446, Gyeongju, Republic of Korea.

Yan Song and Chunyu Kit. 2009. PCFG Parsing with CRF Tagging for Head Recognition. *Proceedings of CIPS-ParsEval*, pages 133–137.

Yan Song, Chia-Jung Lee, and Fei Xia. 2017. Learning Word Representations with Regularization from Prior Knowledge. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 143–152, Vancouver, Canada.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A Minimal Span-Based Neural Constituency Parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada.

Jun Suzuki, Sho Takase, Hidetaka Kamigaito, Makoto Morishita, and Masaaki Nagata. 2018. An Empirical Study of Building a Strong Baseline for Constituency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 612–618, Melbourne, Australia.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. *GitHub repository*.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005. Syntax Annotation for the GENIA Corpus. In *Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts*.

Yuanhe Tian, Yan Song, Xiang Ao, Fei Xia, Xiaojun Quan, Tong Zhang, and Yonggang Wang. 2020a. Joint Chinese Word Segmentation and Part-of-speech Tagging via Two-way Attentions of Auto-analyzed Knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8286–8296, Online.

Yuanhe Tian, Yan Song, and Fei Xia. 2020b. Supertagging Combinatory Categorial Grammar with Attentive Graph Convolutional Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6037–6044.

Yuanhe Tian, Yan Song, and Fei Xia. 2022. Enhancing Structure-aware Encoder with Extremely Limited Data for Graph-based Dependency Parsing. In *Proceedings of the 29th International Conference on Computational Linguistics*.

Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. 2020c. Improving Constituency Parsing with Span Attention. In *Findings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. LLaMA 2: Open Foundation and Fine-tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Karthik Valmeekam, Sarath Sreedharan, Matthew Marquez, Alberto Olmo, and Subbarao Kambhampati. 2023. On the planning abilities of large language models (a critical investigation with a proposed benchmark). *arXiv preprint arXiv:2302.06706*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a Foreign Language. In *Advances in Neural Information Processing Systems 28*, pages 2773–2781.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks. *Advances in Neural Information Processing Systems*, 33:21687–21698.

Sen Yang, Leyang Cui, Ruoxi Ning, Di Wu, and Yue Zhang. 2022. Challenges to Open-Domain Constituency Parsing. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 112–127, Dublin, Ireland.

Songlin Yang and Kewei Tu. 2023. Don't Parse, Choose Spans! Continuous and Discontinuous Constituency Parsing via Autoregressive Span Selection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8420–8433, Toronto, Canada.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32*, pages 5753–5763.

Hongming Zhang, Yan Song, and Yangqiu Song. 2019a. Incorporating Context and External Knowledge for Pronoun Coreference Resolution. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 872–881, Minneapolis, Minnesota.

Hongming Zhang, Yan Song, Yangqiu Song, and Dong Yu. 2019b. Knowledge-aware Pronoun Coreference Resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 867–876, Florence, Italy.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and accurate neural CRF constituency parsing. *arXiv preprint arXiv:2008.03736*.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. *arXiv preprint arXiv:2305.14078*.

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.