Branching Out: Exploration of Chinese Dependency Parsing with Fine-tuned Large Language Models

He Zhou, Emmanuele Chersoni, Yu-Yin Hsu
Department of Chinese and Bilingual Studies
The Hong Kong Polytechnic University
{he.zhou, emmanuele.chersoni, yu-yin.hsu}@polyu.edu.hk

Abstract

In this paper, we investigate the effectiveness of large language models (LLMs) for Chinese dependency parsing through fine-tuning. We explore how different dependency representations impact parsing performance when fine-tuning the Chinese Llama-3 model.

Our results demonstrate that while the Stanford typed dependency tuple representation yields the highest number of valid dependency trees, converting dependency structure into a lexical centered tree produces parses of significantly higher quality despite generating fewer valid structures. The results further show that fine-tuning enhances LLMs' capability to handle longer dependencies to some extent, though challenges remain. Additionally, we evaluate the effectiveness of DeepSeek in correcting LLM-generated dependency structures, finding that it is effective for fixing index errors and cyclicity issues but still suffers from tokenization mismatches.

Our analysis across dependency distances and relations reveals that fine-tuned LLMs outperform traditional parsers in specific syntactic structures while struggling with others. These findings contribute to the research on leveraging LLMs for syntactic analysis tasks.

1 Introduction

Natural language processing (NLP) has been profoundly transformed with the advent of large language models (LLMs), which enable a wide range of NLP downstream tasks, such as question answering and machine translation, to achieve remarkable performance. Despite the advancements of LLMs, fundamental NLP tasks–such as morphological, syntactic and semantic analysis–remain crucial for understanding the structure and meaning of languages. However, LLMs have significantly transformed how these tasks are conducted.

Syntactic parsing is challenging to LLMs (Tian et al., 2024). In NLP, syntactic analysis has two widely used formalisms in constituency grammar and dependency grammar, respectively. Preliminary exploration of using LLMs for syntactic parsing demonstrated that LLMs possess limited capability in generating full parses of good quality (Bai et al., 2023; Lin et al., 2023). On the other hand, the potential of LLMs in parsing still remains largely unexplored.

The conventional approach to syntactic parsing involves training statistical parsing models on treebank data containing a large number of well-annotated trees, which are subsequently deployed to predict the hierarchical syntactic structures of newly input sentences. In contrast, prompting LLMs for parsing shifts the paradigm from sentence-to-structure prediction to sequence-to-sequence generation.

In this paper, we aim to investigate the capability of LLMs in Chinese dependency parsing. Our preliminary experiments, which implement zero-shot and five-shot prompting LLM for dependency parsing, yielded limited performance. Although LLMs are trained on vast amounts of data, they are not explicitly exposed to syntactic trees or structured knowledge of syntactic rules and annotation conventions. Consequently, simply prompting LLMs to perform syntactic parsing may yield subpar results, as LLMs are prone to hallucinating syntactic structures.

Therefore, we start with fine-tuning LLM on treebank data, hypothesizing that fine-tuning will improve LLMs' capabilities to generate dependency structures that adhere to the annotation conventions. A dependency tree is a hierarchical structure where each word, except the root, is connected to one head word, forming a tree-like structure. Guiding LLMs for parsing via prompting involves instructing the LLM to generate a sequence that repre-

Proceedings of Recent Advances in Natural Language Processing,pages 1437–1445
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-166

sents the syntactic structure of an input sentence. However, the encoding methods of a dependency structure into a text sequence and the relationship between tree representations and LLMs' parsing capabilities remain unclear. Additionally, as long dependencies have been challenging for statistical syntactic parsers (Candito and Seddah, 2012), we are also interested in whether large language models can effectively handle long dependencies. It is also conceivable that LLMs may generate dependency structures that are not well-formatted. To address this issue, we leverage LLMs to correct the problematic formats based on specific instructions. Moreover, we also compare parses of LLMs and traditional neural dependency parsers in terms of dependency distance and dependency relations. We specifically aim to investigate whether LLMs demonstrate superior performance in handling particular syntactic structures.

This paper is organized as follows. In Section 2, we describe the experimental settings for fine-tuning large language models and neural dependency parser training. Then, we explain conversion of dependency structures into three types of representations in Section 3. Section 4 will present and discuss experimental results. Previous works will be covered in Section 5, and the paper will be concluded in Section 6.

2 Experimental Settings

2.1 Data

In this experiment, we use the Chinese GSDSIMP treebank[1] within the Universal Dependencies project (De Marneffe et al., 2021). To evaluate the performance of sentences with long dependencies, assuming that longer sentences are more likely to contain long dependencies (Gibson, 1998; Gibson et al., 2000), we measure sentence lengths by the number of tokens (words) in each sentence and divide each set of data into four subsets, shown in Table 1. Specifically, we set the thresholds for the split at 20, 30 and 40 tokens.

2.2 Large Language Models

We use the LLAMA-3-CHINESE-8B-INSTRUCT[2] v3 model (Cui et al., 2023) for fine-tuning. The model is specifically optimized for the Chinese language by continual pre-training on large-scale Chinese data on Meta Llama-3 (AI@Meta, 2024), and is fine-tuned with selected instruction data to further enhance Chinese semantic and instruction understanding capabilities.

When correcting ill-formatted dependency structures, we utilized the open-source model DEEPSEEK-V3 (DeepSeek-AI, 2024). DeepSeek-V3 is built on a Mixture-of-Experts (MoE) architecture with 671 billion parameters with 37 billion activated for each token. It is pre-trained on diverse and high-quality multilingual corpora containing 14.8 trillion tokens. Following pre-training, it undergoes supervised fine-tuning and reinforcement learning to fully harness its capabilities.

2.3 The Deep Biaffine Parser

We also trained a traditional dependency parser using the implementation of the deep biaffine parser (Dozat and Manning, 2017) by Zhang et al. (2020)[3], which is a neural graph-based dependency parser. The parser uses a biaffine classifier and replaces MLP-based attention with biaffine attention, which allows for more relevant information to be retained before being used in the biaffine classifier.

The parser is trained using the default hyperparameters of the original base parser (refer to the Github repository[4]). We use the Chinese RoBERTa model CHINESE-ROBERTA-WWM-EXT (Cui et al., 2020) to encode each input word. A scaler mixture of the last four layers is passed through a linear layer to produce the embeddings.

2.4 Evaluation

When guiding LLMs to generate dependency structures for input sentences via prompt-based approaches, the generated outputs may occasionally violate the formal constraints of well-formed dependency trees, such as single-rootedness and acyclicity. We thus implement a validation step prior to accuracy computation based on the properties of valid dependency structures (see Section 3.1.2), and report the number of valid dependency trees as one of the metrics.

For the evaluation of the valid dependency trees, we utilize the CoNLL 2018 Shared Task Scorer (Zeman et al., 2018)[5]. Our evaluation is based on main

[1] https://github.com/UniversalDependencies/UD_Chinese-GSDSimp/tree/master

[2] https://huggingface.co/hfl/llama-3-chinese-8b-instruct-v3

[3] https://github.com/yzhangcs/parser

[4] https://github.com/hzh1-cl/chinese_dep_parsing_ft_llm

[5] https://universaldependencies.org/conll18/evaluation.html

Data	Sent #	Sent # by token number (t)			
		$t \leqslant 20$	$20 < t \leqslant 30$	$30 < t \leqslant 40$	$40 \leqslant t$
Train	3,997	1,801	1,192	585	419
Dev	500	201	162	86	32
Test	500	232	150	77	41

Table 1: Statistics of the Chinese GSDSIMP treebank

dependency types as the subtypes are removed prior to parser training and LLM fine-tuning. For example, the subtype poss within the possessive nominal modifier nmod:poss is removed, leaving only the main type nmod. We report the unlabeled attachment score (UAS) and labeled attachment score (LAS). UAS is the percentage of words that get assigned to the correct head, and LAS is the percentage of words that that get assigned to both the correct head and dependency label. Both scores are macro-averaged. We additionally analyze accuracies based on dependency labels.

3 Experiments

3.1 Fine-tuning CHINESE LLAMA-3

We adapt Chinese Llama-3 for dependency parsing through supervised instruction fine-tuning using the Parameter-Efficient Fine-Tuning (Mangrulkar et al., 2022, PEFT) approach. This approach integrates lightweight adapters into specific layers of the pre-trained base model, targeting transformer blocks where task-specific adjustments yield the most benefits. During fine-tuning, only the adapters are updated, while the majority of the model's parameters remain frozen to preserve general linguistic knowledge. The fine-tuning used a single GPU card. The hyper-parameters for fine-tuning are listed on the Github repository [6].

3.1.1 Data: Dependencies Conversion

The data used for fine-tuning consists of an instruction, tokenized sentences as inputs and their corresponding dependency structures as outputs, as shown in Table 2. The <*input*> field puts a tokenized sentence, and the tokenization follows the gold standard, as this ensures that the generated dependency structures could be properly evaluated against the gold standard[7].

[6] https://github.com/hzh1-cl/chinese_dep_parsing_ft_llm/blob/main/appendix.pdf

[7] In our preliminary experiments with raw sentences, we observed that, in most cases, the tokenization by LLMs did not align with the gold standard.

"instruction": *You are a dependency parser for Chinese. Given a tokenized Chinese sentence, for each word, identify its head and the dependency relation between them. Do not modify the tokenization of the input sentence. Please parse the sentence with the given words.*
"input": <*sentence*>
"output": <*parse*>

Table 2: Instruction template for fine-tuning

Dependency structures are conventionally represented as CoNLL-U format. In a CoNLL-U formatted sentence, each token is represented on one line, consisting of 10 fields: token ID, Form, Lemma, universal Part-Of-Speech, extended Part-Of-Speech, Morphological features, Head of the current word, universal dependency relation to the head, enhanced dependencies and other needed information. To evaluate how dependency representations impact LLMs' performance, we propose three formats for linearizing syntactic dependencies in the fine-tuning dataset. The linearized parse is put in the <*parse*> field. This tests the assumption that Chinese Llama-3 will exhibit varying performance depending on how dependency relationships are encoded. We demonstrate the conversion of the three formats using an example sentence shown in Figure 1.

The first type of representation is a simplified CoNLL-U format (hereinafter referred to as CONLLU), which linearizes dependency trees. For each token in a sentence, we extract the token's index and word form, along with the index and word form of its head token, as well as the dependency relation between them. The five pieces of information for a tab-separated entry, and all the entries within a sentence dependency structure are separated by line breaks, as exemplified in Table 3.

We also utilize the Stanford typed dependencies representation (De Marneffe and Manning, 2008, hereinafter referred to as SD). The Stanford typed dependencies representation was designed to pro-

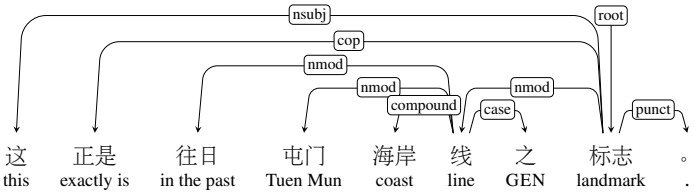


Figure 1: Dependency tree for “这正是往日屯门海岸线之标志。”(This was exactly the landmark of the Tuen Mun coastline in the past.) GEN denotes the genitive case.

1	这	8	标志	nsubj
2	正是	8	标志	cop
3	往日	6	线	nmod
4	屯门	6	线	nmod
5	海岸	6	线	compound
6	线	8	标志	nmod
7	之	6	线	case
8	标志	0	ROOT	root
9	。	8	标志	punct

Table 3: Simplified CoNLL-U representation of the example sentence Fig.1.

vide a simple description of the grammatical relationships in a sentence that can easily be understood and effectively used. It represents all sentence relationships uniformly as typed dependency relations, that is, as triples of a relation between pairs of words. Table 4 presents the example sentence of Fig. 1 in the Stanford typed dependencies format. For instance, nsubj(标志-8, 这-1) denotes “the nominal subject of 这 (index 1) is 标志 (index 8)”. We assume that this representation can be easily understood by LLMs as well.

nsubj(标志-8, 这-1)
cop(标志-8, 正是-2)
nmod(线-6, 往日-3)
nmod(线-6, 屯门-4)
compound(线-6, 海岸-5)
nmod(标志-8, 线-6)
case(线-6, 之-7)
root(ROOT-0, 标志-8)
punct(标志-8, 。-9)

Table 4: Stanford typed dependency representation of the example sentence Figure 1

For the third representation, we convert the dependency structure of a sentence into a lexical centered tree structure (Croce et al., 2011; Hromei et al., 2024, hereinafter referred to as LCT). In a tree, the nodes correspond to words, and edges express syntactic relationships, each labeled with a specific dependency type. Figure 2 portrays the syntactic parse tree structure of the same example sentence from Figure 1. In this tree, non-terminal nodes are words with the root positioning at the top, while terminal nodes are dependency relations. The plain bracket representation of the dependency tree in Figure 2 correspond to Table 5. We hypothesize that the tree structure representation can more effectively preserve hierarchical information, although it may pose challenges to LLMs.

[标志-8 [这-1 [nsubj]] [正是-2 [cop]] [线-6 [往日-3 [nmod]] [屯门-4 [nmod]] [海岸-5 [compound]] [nmod] [之-7 [case]]] [root] [。-9 [punct]]]

Table 5: Bracket representation of Figure 2

3.1.2 Output Validation

After LLMs generate a parse, we validate the output by ensuring adherence to three formal properties required for a well-formed dependency tree. First, each token in the sentence must be assigned a unique and consecutive numerical index in the parse structure, ensuring linear ordering of all words. Second, except for the ROOT token, no token may have a head index identical to its own index, which prevents self-referential dependencies. Third, the structure must satisfy acyclicity, that is, no pair of tokens in the sentence can form a mutual dependency cycle. If a parse conforms to the three criteria, it is considered as a valid dependency structure.

However, to guarantee that the generated parse can be directly compared with the gold standard annotation, a fourth criterion becomes essential: the alignment of tokenization between the LLMs’ output and the gold standard data. This alignment ensures that token boundaries match precisely, avoiding evaluation failures caused by tokenization discrepancies. Consequently, in our experiments, the number of valid trees is equivalent to the number of LLMs’ parses that meet all four criteria.

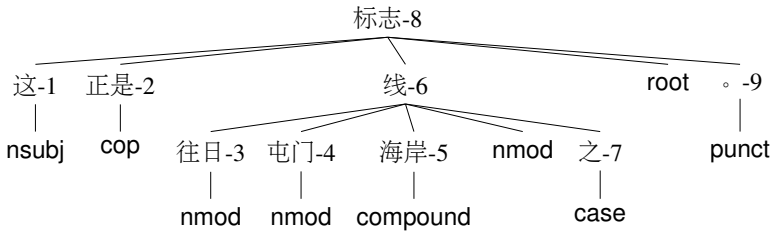


Figure 2: Lexical centered tree structure for the example sentence of Figure 1

3.2 Correct Parsing Errors with DeepSeek-V3

For the parses that do not pass validation, we leverage DeepSeek-V3 with prompting to correct the ill-formed dependency structures and tokenization misalignments (the prompt can be found in the Github repository[8]). Subsequently, the revised parses undergo a validation process as described in Section 3.1.2.

4 Results and Discussion

4.1 How does the fine-tuned Chinese Llama-3 perform on different syntactic dependency representations?

Table 6 presents the parsing results of the three representations with the fine-tuned Chinese Llama-3, including the number of valid trees we obtained and the attachment scores. We also compare these results against the biaffine parser. First, we observed a pronounced sentence length effect: the fine-tuned LLM exhibits better parsing performance on sentences containing less than 30 words but struggles parsing long sentences. In particular, when the dependency structures are in the LCT format, the LLM generated no valid trees containing more than 30 words. The observed pattern correlates with the sentence length distribution of the training data used for fine-tuning (see Table 1).

Across the three formats, the SD representation achieved the highest yield (295 valid trees), outperforming LCT (194 trees) and CONLLU (153 trees). This conforms to the assumption in Section 3.1.1, where we posit that the simplicity of the SD representation could enhance the LLM's understanding. The CONLLU format, which linearizes the syntactic dependencies by concatenating word and head information, poses challenges for the LLM. The flattened representation obscures the hierarchical relationships, limiting the model's capability to infer tree-structured dependencies. In contrast, the LCT format explicitly encodes the syntactic hierarchies, better preserving structural information. However, its increased complexity also poses significant challenges to the model.

In comparing the UAS and LAS, the fine-tuned Chinese Llama-3 surpasses the biaffine parser on both CONLLU and LCT formatted dependency trees. For the CONLLU formatted trees, the LLM achieves 1.99 points higher in UAS (81.56 vs. 79.57) and 1.68 points higher in LAS (73.61 vs. 71.93). The performance gap widens for the LCT formatted trees, where the LLM outperforms the biaffine parser for 3.62 points in UAS (86.12 vs. 82.50) and 8.03 points in LAS (82.47 vs. 74.44). These results suggest that, while the LLM does not always produce well-formatted trees, it does possess the capability for syntactic dependency analysis. Although the LLM generated the highest yield of well-formatted trees with the SD representation, it significantly underperformed the biaffine parser, with a margin of 8.67 points in UAS (70.95 vs. 79.62) and 9.60 points in LAS (61.72 vs. 71.32).

4.2 Can Deepseek-V3 correct erroneous dependency structures?

For the outputs that did not pass the validation, we feed them into DeepSeek-V3 (DeepSeek-AI, 2024) with prompt (which can be found at the Github repository[9]), and request DeepSeek to identify the errors and offer corrections. Table 7 presents the evaluation results for the three types of representations. Despite not being fine-tuned for dependency parsing, DeepSeek managed to correct 117 out of 306 erroneous LCT-formatted dependencies, 64 out of 205 SD-formatted trees, and 58 out of 336 CONLLU-formatted trees. We further take a closer look at sentences that DeepSeek does not correct, finding that the majority of those sentences have

[8] https://github.com/hzh1-cl/chinese_dep_parsing_ft_llm/blob/main/appendix.pdf

[9] https://github.com/hzh1-cl/chinese_dep_parsing_ft_llm/blob/main/appendix.pdf

# Tokens		$t \leqslant 20$	$20 < t \leqslant 30$	$30 < t \leqslant 40$	$40 \leqslant t$	Total
# Gold Sentences		232	150	77	41	500
CONLLU	# *valid*	120	27	5	1	153
	UAS	84.75	77.91	70.00	72.09	**81.56**
	LAS	76.94	68.40	63.53	60.47	**73.61**
biaffine$_{conllu}$	UAS	81.82	78.37	67.65	55.81	79.57
	LAS	74.82	69.33	58.82	48.84	71.93
SD	# *valid*	203	79	13	0	295
	UAS	79.00	61.93	56.82	-	70.95
	LAS	69.30	53.04	48.86	-	61.72
biaffine$_{sd}$	UAS	82.34	76.24	76.14	-	**79.62**
	LAS	74.52	67.04	68.64	-	**71.32**
LCT	# *valid*	155	39	0	0	194
	UAS	86.91	84.26	-	-	**86.12**
	LAS	83.18	80.78	-	-	**82.47**
biaffine$_{lct}$	UAS	83.73	79.59	-	-	82.50
	LAS	75.48	71.99	-	-	74.44

Table 6: Parsing performance comparison between dependency parses generated by the fine-tuned Chinese Llama-3 and the biaffine parser, where # *valid* refers to the number of valid output, and *biaffine*$_X$ denotes the biaffine parser's performance on the validated sentences in format X.

token mismatch problems. On one hand, this suggests that the fine-tuned Chinese Llama-3 does not strictly follow the instruction of not modifying the input tokenization. On the other hand, it also shows that DeepSeek is able to correct format errors, such as index errors or cyclicity issues, but it cannot effectively address token mismatches, especially for the CONLLU formatted trees.

The number of valid sentences after correction also reveals a clear sentence length effect: shorter sentences are less challenging for the LLM. In addition, when comparing the attachment scores of the DeepSeek outputs to those of the biaffine parser, the LLM achieved only subpar performance, with substantial gaps. This is understandable, since DeepSeek is not fine-tuned for dependency parsing, thus it does not possess sufficient capabilities to accurately identify correct head and dependency relations. Despite this limitation, DeepSeek still proves to be a very effective tool in examining, identifying and correcting format errors.

4.3 Performance measured by dependency distance and dependency relations

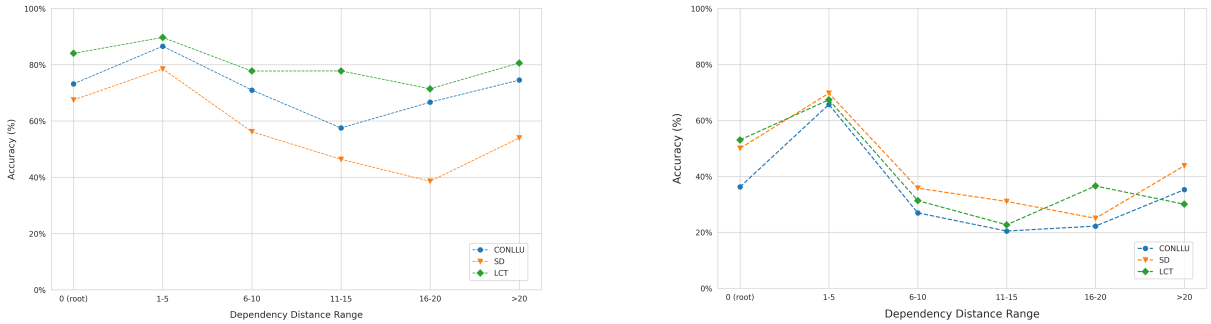
Dependency Distance We take a closer look at the accuracies by dependency distance for the parses obtained from the fine-tuned Chinese Llama-3 and the revised parses obtained from DeepSeek, as shown in Figure 3. In Figure 3a, we observe that as the dependency distance increases, the model faces greater challenges, leading to a decline in accuracy. However, it is interesting to note that the accuracy begins to improve when the dependency distance exceeds 20. We also find a similar pattern on DeepSeek, which is used directly for error correction without fine-tuning, as seen in Figure 3b. Moreover, while both models struggle with long dependency distances, we notice that DeepSeek endures a sharp drop in accuracy when the dependency distance is greater than 5. This indicates that the LLM is able to enhance its ability to handle longer dependencies through fine-tuning. Also, from Figure 3a, we find that fine-tuning the LLM with the lexical centered tree representation have a better performance in all dependency distances.

Dependency Relations We also investigate the performance by dependency relations via comparing the fine-tuned Chinese Llama-3 and the biaffine dependency parser, and plot comparison graphs for the three types of representation, shown in Figure 4. As discussed in Section 4.1, fine-tuning using the CONLLU and LCT formatted dependency trees outperforms the fine-tuning on the SD format. Regarding dependency relations, when fine-tuned on the CONLLU formatted dependency trees, the LLM outperforms the biaffine parser in most of the dependency relations, with exceptions involving relations such as obj ('direct object'), nummod ('numeral modifier') and root, as seen in Figure 4a.

Similarly, Figure 4c demonstrates that the LLM also outperforms the biaffine parser on LCT formatted trees for most relations, except for a few relations such as parataxis ('run-on sentences') and

# Tokens		$t \leqslant 20$	$20 < t \leqslant 30$	$30 < t \leqslant 40$	$40 \leqslant t$	Total
# Gold Sentences		232	150	77	41	500
CONLLU	# *error*	112	121	71	32	336
	# *valid*	27	17	12	2	58
	UAS	52.39	52.02	50.87	69.57	53.01
	LAS	42.07	41.57	42.18	65.22	43.56
biaffine$_{conllu}$	UAS	79.60	77.20	71.71	63.04	**75.25**
	LAS	71.28	67.46	63.03	60.87	**66.79**
SD	# *error*	29	71	64	41	205
	# *valid*	16	27	15	6	64
	UAS	66.67	56.95	60.80	53.62	58.92
	LAS	62.02	47.09	48.70	46.38	49.71
biaffine$_{sd}$	UAS	82.56	78.03	74.05	71.74	**76.53**
	LAS	72.48	68.16	63.67	66.30	**67.19**
LCT	# *error*	77	111	77	41	306
	# *valid*	49	43	18	7	117
	UAS	61.68	52.86	56.03	49.19	55.68
	LAS	55.47	45.08	46.78	40.78	49.79
biaffine$_{lct}$	UAS	79.93	78.09	75.37	73.46	**77.54**
	LAS	73.72	68.52	65.79	64.08	**68.96**

Table 7: Parsing performance comparison between dependency parses revised by prompting DeepSeek-V3 and the biaffine parser, where # *error* refers to the number of erroneous trees that need correction, # *valid* refers to the number of valid output, and *biaffine*$_X$ denotes the biaffine parser's performance on the validated sentences in format X.



(a) Fine-tuned Chinese Llama-3

(b) DeepSeek-V3

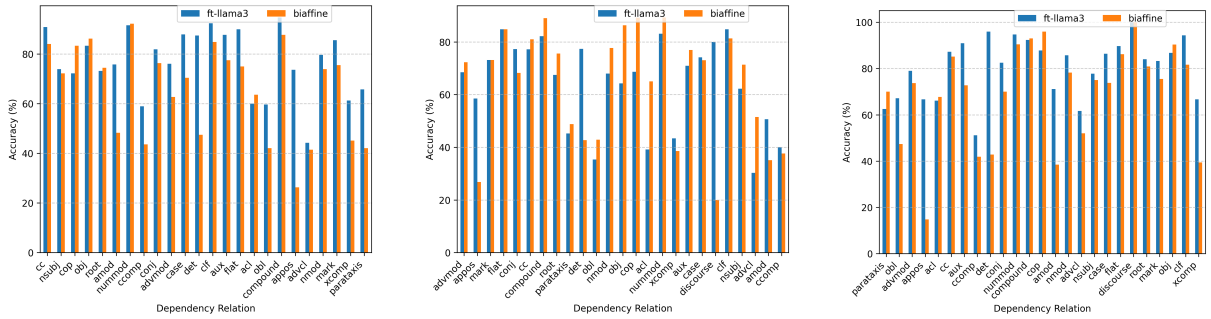
Figure 3: Accuracies computed by dependency distance, where '0 (root)' refers to the accuracy of identifying root of a sentence.

obj. For the SD format, the LLM performs poorly on most relations but significantly outperforms the biaffine parser on relations such as appos, det, discourse and amod. This suggests that the parsing ability of the fine-tuned LLM is influenced by the representation of dependency structures, and the SD representation limits the model's ability to learn complex syntactic structures effectively.

Although different types of dependency structure representations have a different impact on parsing performance, we observe that the fine-tuned LLM consistently outperforms the biaffine parser in certain dependency relations, including nominal phrase relations such as det ('determiner'), appos ('appositional phrase') and amod ('adjectival modifier'), as well as clausal structures such as ccomp ('clausal complement') and xcomp ('open clausal complement'). However, the LLM is not good at handling specific sentence structures, such as cop ('copula'), which refers to the nominal predicate sentences', and acl ('clausal modifier of noun'), which specifically refers to relative clauses.

5 Related Work

Traditionally, dependency parsing is data-driven, based on training machine learning models on a dependency treebank to parse a given input sentence. The data-driven parsing approach can be either transition-based (Nivre, 2003; Chen and Manning,



(a) CONLLU format (b) SD format (c) LCT format

Figure 4: Accuracy comparison computed by dependency relations between the fine-tuned Chinese Llama-3 and the biaffine parser in three types of tree presentation.

2014) or graph-based (Chen et al., 2012; Dozat and Manning, 2017). When incorporating transformer-based encoder models (such as BERT, Devlin et al. (2019)), words are first encoded using the word embeddings extracted from the models and then passed as input to the parser. However, generative large language models are rarely employed for dependency parsing due to the complexity of the parsing task.

Lin et al. (2023) utilized ChatGPT for dependency parsing. They created prompts to instruct the LLM to parse 12 languages. The results showed that ChatGPT underperformed traditional parsing algorithms on all the languages involved with a large gap. Hromei et al. (2024) introduced U-DepPLLaMA, which shifted traditional dependency parsing as a sequence-to-sequence generation task, interpreting and transform syntax into bracketed structures. They evaluated 50 datasets in 26 languages, showing competitive performance with traditional parsing algorithms.

In addition to dependency parsing, previous studies also employed LLMs for constituency parsing. Bai et al. (2023) employed three linearization strategies to convert trees into symbol sequences and conducted parsing experiments using a series of LLMs with zero-shot, few-shot and full-training settings. Results demonstrated that LLMs enhanced the performance of sequence-based methods and can yield competitive results compared to chart-based and transition-based parsers. However, LLMs did not achieve equally good performance on few-shot and cross-domain scenarios. Tian et al. (2024) investigated the effectiveness of directly using LLMs for constituency parsing. They observed that LLMs are shallow parsers, since they are effective in chunking but ineffective in full parsing. They also proposed to decompose parsing into three steps: chunking, filtering and parsing with chunks, and indicated that the three-step approach can produce better parse trees.

6 Conclusion

In this paper, we investigated how different dependency representations impact parsing performance when fine-tuning large language models for Chinese dependency parsing. Results comparing with the traditional dependency parsers indicate that using the lexical centered tree representation helps the fine-tuned LLM obtain higher quality parses, despite not producing the highest number of valid trees. When using the Stanford dependency format, the LLM generates the highest number of well-formatted trees, though these trees exhibit lower accuracy. We also utilized DeepSeek for correcting erroneous parses, finding that it is able to fix index errors or cyclicity issues, but cannot solve tokenization mismatches. Furthermore, analyzing from dependency distance and dependency relations, the fine-tuned LLM enhances its capability to handle longer dependencies, although this remains challenging. Our future work will focus on generating more valid trees and incorporate more features into fine-tuning to investigate whether it will further improve parsing performance.

Acknowledgements

We are grateful to the anonymous reviewers for their valuable feedback. This research was supported by the Postdoc Matching Fund (Project code: 1-W28X) from the Hong Kong Polytechnic University.

References

AI@Meta. 2024. Llama 3 model card.

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

Marie Candito and Djamé Seddah. 2012. Effectively long-distance dependencies in french: annotation and parsing evaluation. In *TLT 11-The 11th International Workshop on Treebanks and Linguistic Theories*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Wenliang Chen, Min Zhang, and Haizhou Li. 2012. Utilizing dependency language models for graph-based dependency parsing models. In *Proceedings of ACL*.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046.

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

Yiming Cui, Ziqing Yang, and Xin Yao. 2023. Efficient and effective text encoding for chinese llama and alpaca. *arXiv preprint arXiv:2304.08177*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

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

DeepSeek-AI. 2024. Deepseek-v3 technical report.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, page 4171–4186, Minneapolis, MN.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68(1):1–76.

Edward Gibson et al. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. *Image, language, brain*, 2000:95–126.

Claudiu Daniel Hromei, Danilo Croce, and Roberto Basili. 2024. U-deppllama: Universal dependency parsing via auto-regressive large language models. *IJCoL. Italian Journal of Computational Linguistics*, 10(10, 1).

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*.

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

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the eighth international conference on parsing technologies*, pages 149–160.

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.

Daniel Zeman, Jan Hajic, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual parsing from raw text to universal dependencies*, pages 1–21.

Yu Zhang, Zhenghua Li, and Zhang Min. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of ACL*, pages 3295–3305.