Step-by-step Instructions and a Simple Tabular Output Format Improve the Dependency Parsing Accuracy of LLMs

Hiroshi Matsuda **Chunpeng Ma**
Megagon Labs, Tokyo,
Recruit Co., Ltd.
{hiroshi_matsuda, ma.chunpeng}@megagon.ai

Masayuki Asahara
National Institute for Japanese
Language and Linguistics
masayu-a@ninjal.ac.jp

Abstract

Recent advances in large language models (LLMs) have enabled impressive performance in various tasks. However, standard prompting often struggles to produce structurally valid and accurate outputs, especially in dependency parsing. We propose a novel step-by-step instruction strategy, where universal part-of-speech tagging precedes the prediction of syntactic heads and dependency labels, and a simplified CoNLL-U like output format, our method achieves state-of-the-art accuracy on Universal Dependencies datasets across 17 languages without hallucination or contamination. We further show that multilingual fine-tuning simultaneously improves cross-language generalization performance. Our results highlight the effectiveness of explicit reasoning steps in LLM-based parsing and offer a scalable, format-consistent alternative to bracket-based approaches.

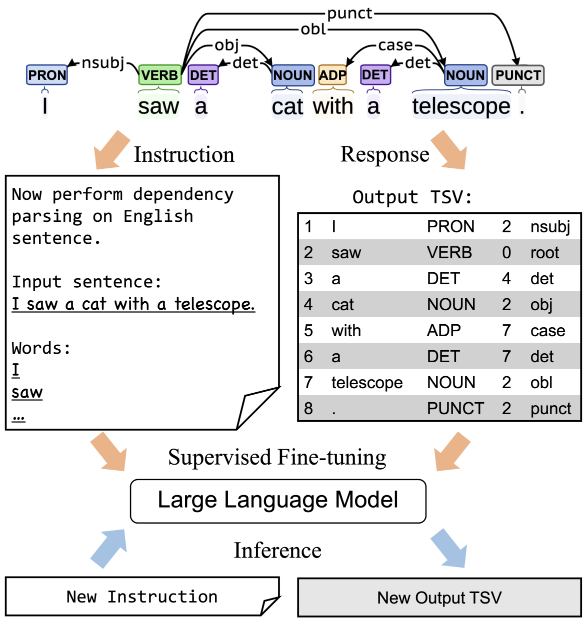


Figure 1: Framework of the proposed method.

1 Introduction

Recent advances in large language models (LLMs) have dramatically reshaped the landscape of natural language processing; however, their potential for syntactic analysis – particularly dependency parsing – remains underexplored. Furthermore, it is desirable to systematically investigate prompting and fine-tuning techniques that enhance the performance of LLM-based dependency parsing.

In this work, we examine how fine-tuned LLMs can be effectively guided to perform accurate dependency parsing using simple, structured instruction prompts. Specifically, we design a single-turn supervised fine-tuning setup where the input sentence is accompanied by a tabular output format based on a minimal subset of the `CoNLL-U`[1], which is the standard format of Universal Dependencies (UD) treebanks (Nivre et al., 2020) as in Figure 1. This table-based representation not only improves format validity and readability, but also facilitates learning non-projective structures.

The results of our preliminary experiments using `UD_English-EWT`[2] are summarized in Table 1. First, we found that performing SFT with a single-step prompt yielded accuracy comparable to or better than that of `UDPipe 2.0` (Straka, 2018). Next, we introduced a step-by-step prompting strategy in a Chain-of-Thought style (Wei et al., 2022). Specifically, we first predict UPOS tags, then syntactic heads and dependency relations. We observed that step-by-step prompts leads to substantial gains in both unlabeled attachment score (UAS) and labeld attachment socre (LAS).

Despite using a very simple prompt, we observed fairly high parsing accuracy, prompting us to investigate the possibility of data contamination (refer Appendix B for details). Based on our analysis, we found no evidence of contamination in

[1] https://universaldependencies.org/format.html

[2] https://universaldependencies.org/treebanks/en_ewt/

Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), pages 11–19
August 26, 2025 ©2025 Association for Computational Linguistics

	Token Recall	UPOS	UAS	LAS
`UDPipe 2.0`	100.0	97.5	93.4	91.5
`gpt-4o-mini-2024-07-18`: Chain-of-Thought Steps				
1 (UPOS+HEAD+DEPREL)	98.5	98.4	93.3	91.4
2 (UPOS; HEAD+DEPREL)	99.1	**98.6**	94.8	93.1
3 (UPOS; HEAD; DEPREL)	**99.6**	**98.6**	**95.1**	**93.5**

Table 1: Preliminary experiment on evaluating Chain-of-Thought effect in `UD_English-EWT r2.15`. We performed all steps within a single-turn prompt. The example prompts are presented in Appendix C.

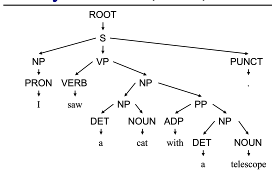
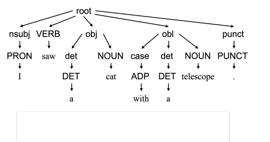
Vinyals et al. (2015)	Hromei et al. (2024)
	
(ROOT♦(S♦ (NP♦PRON♦)$_{NP}$♦ (VP♦VERB♦(NP♦ (NP♦DET♦NOUN♦)$_{NP}$♦ (PP♦ADP♦ (NP♦DET♦NOUN♦)$_{NP}$♦)$_{PP}$♦)$_{NP}$♦)$_{VP}$♦ PUNCT♦)$_{S}$♦)$_{ROOT}$	[root♦[nsubj♦[PRON♦[I♦]]] ♦[VERB♦[saw♦]]♦ [obj♦[det♦[DET♦[a♦]]]♦ [NOUN♦[cat♦]]]♦ [obl♦[case♦[ADP♦[with♦]]] ♦[det♦[DET♦[a♦]]]♦ [NOUN♦[telescope♦]]]♦ [punct♦[PUNCT♦[.♦]]]]

Table 2: Comparison of bracket-based linearization methods. Syntactic elements are separated by "♦".

the prediction of syntactic heads, and dependency relations by the models used in this study for the test set of `UD_English-EWT r2.15`. However, we suspect that the part-of-speech tagging may have been exposed to the models during its pre- and mid-training[3] phases.

2 Related Work

Linearization techniques are essential for both constituency parsing (Vinyals et al., 2015; Ma et al., 2017) and dependency parsing (Li et al., 2018; Hromei et al., 2024) using sequence-to-sequence model with bracket-based representations, illustrated in Table 2.

In generative parsing using bracket-based representations, the tree structure in the output text is often invalid, which is one of the factors that reduces the accuracy of parsing, resulting in additional recovery procedure (Bai et al., 2023), or even redesign the topology of neural networks to ensure the output validity (Dyer et al., 2015; Gómez-Rodríguez and Vilares, 2018).

[3]https://vintagedata.org/blog/posts/what-is-mid-training

3 Approach

In this section, we describe a table-based representation of dependency structures, similar to the `CoNLL-U` format, and explain how to construct instruction prompts for dependency parsing.

3.1 Table-based representation

Recent large language models (LLMs) have significantly improved their ability to output in structured formats such as JSON or CSV, enabling function calling for flexible interaction with external services[4]. This capability facilitates the direct handling of tabular structures such as `CoNLL-U`, potentially allowing LLMs to generate parse results with higher structural validity compared to the bracket-based representations employed in prior studies.

In this work, we adopt a table-based representation that extracts only the essential fields – ID, FORM, UPOS, HEAD, and DEPREL – from the `CoNLL-U` format, as illustrated in the output TSV in Figure 1. A further advantage of the table-based approach is its ability to naturally represent non-projective dependency structures using index-based head references. However, it should be noted that table-based representations can represent circular references and multiple roots. As we demonstrate in the next section, the tabular outputs generated by the LLMs were mostly well-formed, and the validity errors were fairly rare on the `UD_English-EWT` r2.15 test set. Furthermore, the table-based representation offers an advantage in recovery processing, as it can accurately recover word indices and forms as long as the number of records and the field structure are correctly output.

3.2 Step-by-step instruction prompts

We began our preliminary experiments using the simple single-step prompt illustrated in Figure 1. Through iterative refinement, we found that parsing the UPOS tags first, followed by the HEAD and DEPREL fields in a step-by-step manner, led to improved accuracy. Accordingly, the experiments presented in next chapter employ a three-step Chain-of-Thought prompting strategy, processing the elements in the order of UPOS, HEAD, and DEPREL. Representative examples of these prompt templates are provided in the Appendix C.

[4]https://platform.openai.com/docs/guides/function-calling

Models	# of Parameters Trainable vs. Transformer	UD_English-EWT r2.2 Token Recall	UPOS	UAS	LAS	UD_English-EWT r2.15 Token Recall	UPOS	UAS	LAS
Baselines:									
UDPipe 2.0	30.1M† / 168M	-	-	-	-	-	97.5	93.4	91.5
Hexatagger bert-base	222M / 178M	-	-	91.4	88.7	-	-	93.5	91.4
(+ Gold POS)	224M / 178M	-	-	(93.3)	(91.1)	-	-	(93.8)	(92.1)
U-DepP Llama-2-13b-hf*	31.6M / 13.0B	96.3	-	88.9	86.6	95.3	-	92.3	90.1
U-DepP Llama-3.1-8B*	22.0M / 8.05B	98.4	-	90.7	88.4	97.8	-	92.0	90.2
Our methods:									
gpt-4o-mini-2024-07-18	(undisclosed)	99.6	97.7	93.2	91.1	99.5	98.3	94.9	93.3
gpt-4o-2024-08-06	(undisclosed)	99.7	**98.1**	93.7	91.7	**100.0**	98.5	95.2	93.5
gemma-2-2b*	10.4M / 2.62B	99.7	98.0	93.2	91.1	99.9	98.3	94.4	92.7
Qwen2.5-7B*	20.2M / 7.64B	99.5	97.9	93.1	91.0	99.5	98.3	94.7	93.0
Llama-3.1-8B*	21.0M / 8.05B	99.4	97.8	93.4	91.3	**100.0**	98.4	94.8	93.1
gemma-2-9b*	27.0M / 9.27B	**99.8**	**98.1**	**93.8**	**91.9**	**100.0**	**98.6**	**95.5**	**94.1**

Table 3: Evaluation of various models in UD_English-EWT r2.2 and r2.15. Best scores are highlighted in **bold**. The scores for UDPipe 2.0 are taken from its official documentation. The scores for Hexatagger and U-DepPLLaMA are the results of our reproduction experiments. The scores in the row (+ Gold POS) are provided for reference, as they use gold POS tags. The LoRA-SFT models are marked by "*". "†" indicates that the value is estimated from the size of distributed model archive.

Dataset	Baselines: UDPipe 2.0 UPOS	UAS	LAS	Hexatagger(+Gold POS) UAS	LAS	Ours: Monolingual: gemma-2-2b* UPOS	UAS	LAS	Qwen2.5-7B* UPOS	UAS	LAS	Llama-3.1-8B* UPOS	UAS	LAS	gemma-2-9b* UPOS	UAS	LAS	Ours: Multilingual: gemma-2-9b* UPOS	UAS	LAS
ar_padt	97.0	88.1	83.7	86.8 (87.9)	81.8 (83.9)	95.4	87.7	83.1	96.1	88.7	84.3	**97.1**	**90.0**	**85.5**	96.2	89.8	85.4	96.2	89.9	**85.7**
bg_btb	99.3	95.3	92.6	94.7 (95.4)	92.0 (92.4)	99.3	95.3	92.9	99.2	95.2	92.7	99.1	95.7	93.2	**99.5**	**96.9**	**94.7**	**99.5**	**97.0**	**94.8**
ca_ancora	99.2	94.9	93.4	94.8 (95.1)	93.1 (93.7)	98.9	94.5	92.9	99.0	94.7	93.1	99.1	95.1	93.6	**99.3**	**95.7**	**94.5**	**99.3**	**95.7**	94.3
cs_pdt◇	99.4	95.7	94.3	94.3 (94.7)	92.5 (92.9)	99.3	95.1	93.6	99.3	95.2	93.8	99.3	95.5	94.2	99.4	95.7	**94.4**	99.3	94.9	93.3
de_gsd	97.1	89.2	85.5	87.9 (90.2)	83.8 (86.1)	97.0	89.1	85.3	97.1	89.7	85.9	97.1	89.7	86.1	**97.4**	**90.2**	**86.5**	**97.4**	**90.3**	**86.7**
en_ewt	97.5	93.4	91.5	93.5 (93.8)	91.4 (92.1)	98.3	94.4	92.7	98.3	94.7	93.0	98.4	94.8	93.1	**98.6**	**95.5**	**94.1**	**98.6**	**95.5**	**94.1**
es_ancora	99.1	94.0	92.4	93.4 (94.6)	91.6 (92.9)	99.1	93.7	92.0	99.1	94.1	92.6	99.1	94.2	92.7	99.1	**94.6**	**93.2**	**99.2**	**95.1**	**93.7**
fr_gsd	98.5	95.0	93.2	94.4 (96.0)	93.3 (94.6)	98.6	95.6	94.0	98.7	95.9	94.4	98.5	95.6	93.9	98.7	**96.7**	**95.3**	**98.8**	**96.7**	95.2
it_isdt	**98.8**	95.1	93.4	95.7 (96.5)	94.0 (95.2)	98.6	95.6	94.0	98.4	94.8	93.2	98.7	95.8	94.4	98.7	**96.0**	**94.6**	**98.8**	**96.4**	**94.9**
ja_gsd	98.6	95.1	94.2	94.6 (95.2)	93.3 (94.6)	98.7	94.4	93.1	98.8	94.8	93.6	98.5	94.8	93.6	**99.0**	**95.7**	**94.9**	98.9	95.5	94.5
ko_gsd	96.7	88.9	85.2	87.4 (87.9)	83.6 (85.1)	96.5	87.4	83.9	96.8	89.3	86.5	96.9	88.8	85.9	**97.2**	**90.3**	**87.3**	**97.2**	**90.6**	**87.6**
nl_alpino	98.2	94.9	92.9	93.3 (94.1)	90.6 (91.5)	98.1	94.0	91.3	97.4	94.0	91.6	97.3	93.9	91.6	**98.4**	**95.7**	**93.9**	**98.5**	95.6	93.7
no_bokmaal	98.6	94.7	93.4	**95.9** (96.5)	94.3 (94.8)	98.7	94.7	93.5	98.5	94.2	92.9	98.7	95.0	93.8	**98.8**	95.7	**94.8**	98.7	95.5	94.6
ro_rrt	98.1	92.6	89.1	91.9 (92.9)	87.9 (89.0)	98.1	92.5	88.9	97.9	92.7	89.2	97.7	93.1	89.7	98.1	**94.4**	**91.3**	**98.4**	94.3	**91.3**
ru_syntagrus	98.8	94.5	92.3	94.1 (95.1)	91.7 (92.7)	98.8	95.2	93.1	98.9	95.3	93.4	98.9	95.5	93.5	**99.0**	**95.8**	**93.8**	98.6	95.0	92.7
sl_ssj	98.8	94.5	92.9	94.0 (94.8)	92.3 (93.2)	98.5	94.0	92.4	98.7	94.1	92.4	98.9	94.7	93.2	**99.0**	**95.9**	**94.6**	**99.0**	**95.9**	**94.7**
zh_gsdsimp	95.8	86.7	83.6	87.4 (90.1)	84.3 (87.9)	96.4	86.5	83.6	96.5	87.2	84.4	96.8	87.7	84.7	**97.5**	**89.3**	**86.9**	97.3	**89.7**	**87.3**

Table 4: Evaluation results on various UD r2.15 datasets. For each language, best scores among the baselines and our monolingual models are shown in **bold**, with ties and second-best scores underlined. Additionally, scores from our multilingual model that outperform the baselines and monolingual models are also highlighted. The scores for UDPipe 2.0 are taken from its official documentation. The scores for Hexatagger are the results of our reproduction experiments. The scores in the brackets are provided for reference, as they use gold POS tags. LoRA-SFT models are marked by "*". "◇" indicates use of a language-specific pre-trained model in UDPipe 2.0.

4 Experiments

We conducted both supervised fine-tuning (SFT) with Low-Rank Adaptation (LoRA) (Hu et al., 2022) and inference experiments for open LLMs on a high-performance cloud service[5,6]. For OpenAI models, SFT was performed via the official web console[7]. We explored SFT hyper-parameters[8] on the UD_English-EWT r2.15 development set and applied them to all experiments. We used simple TSV recovery process only restores the ID and FORM on a row-by-row basis.

[5]Experiments were conducted on a Google Cloud A2 Ultra instance with 8 × NVIDIA A100 GPUs (80GB each), 96-core Intel Xeon CPUs @ 2.20GHz, 1,360GB RAM, and 5TB of SSD storage. The software environment included: Ubuntu 22.04, CUDA 12.1, Python 3.11.9, PyTorch 2.5.1, Transformers 4.49.0, TRL 0.15.2, PEFT 0.14.0, OpenAI 1.68.2, Unsloth 2025.3.18, and vLLM 0.7.2.

[6]The implementation used in the experiments is available on GitHub. https://github.com/megagonlabs/llmpp

[7]https://platform.openai.com/docs/guides/fine-tuning. The cost of fine-tuning the en_ewt-r2.15 train set for 2 epochs was about $52 for gpt-4o-mini and about $430 for gpt-4o.

[8]**Open LLMs:** num_epochs=3, max_seq_length=8192, lr=3e-4, lr_scheduler=cosine_with_min_lr, min_lr=0.1, LoRA: r=8, dropout=0.05, target_modules="all-linear" (embedding layers excluded). **OpenAI:** num_epochs=2, max_seq_length=8192, lr=default.

4.1 Dataset

We mainly used Universal Dependencies treebanks r2.15. For UD_English-EWT (en_ewt), we also used r2.2 for comparison with baseline methods.

For monolingual SFT. We used datasets for the following 17 languages to evaluate the parsing accuracy for each language: ar_padt, bg_btb, ca_ancora, cs_pdt, de_gsd, en_ewt, es_ancora, fr_gsd, it_isdt, ja_gsd, ko_gsd, nl_alpino, no_bokmaal, ro_rrt, ru_syntagrus, sl_ssj, and zh_gsdsimp. Statistics for each dataset are provided in the Appendix A.

For multilingual SFT. To train a multilingual parsing model, we constructed a new dataset by gathering training sets from the datasets for the 17 languages above. To reduce training time and costs, we downsampled cs_pdt and ru_syntagrus by 17% to balance them with other language datasets. The final training data consisted of 182,255 sentences and 3,889,494 tokens, which was used to train a multilingual model (denoted as 17_multi below). Additionally, we evaluated the following 10 language datasets not included in the multilingual training data: el_gdt, he_htb, hi_hdtb, hu_szeged, id_gsd, pt_gsd, sv_talbanken, tr_imst, vi_vtb, and zh_gsd.

4.2 Baseline methods

We compared our method against three strong baselines: UDPipe 2.0[9] (Straka, 2018), Hexatagger[10] (Amini et al., 2023), and U-DepPLLaMA[11] (Hromei et al., 2024). The reported scores for UDPipe 2.0 were taken from its official documentation, while the results for Hexatagger and U-DepPLLaMA were reproduced in our environment using their publicly available implementations[12]. For Hexatagger, we report the accuracy under the setting that does not use gold POS tags (the accuracy when using gold POS tags is also provided as a reference).

4.3 Evaluation of various models

We conducted a comparative evaluation of baselines and our SFT method with various LLMs including gpt-4o-mini[13], gpt-4o[14], gemma-2-2b[15], Qwen2.5-7B[16], Llama-3.1-8B[17], and gemma-2-9b[18] using UD_English-EWT.

Results are summarized in Table 3. Overall, gemma-2-9b achieved the highest performance, followed closely by gpt-4o. Beyond Table 3, circular references were rare, with only 3 cases found in the output of Qwen2.5-7B, and no multiple roots found in the output of either model on the test set. These results highlight the favorable cost-performance trade-off of open LLMs, leading us to exclude OpenAI models from the subsequent experiments.

From the perspective of model parameter size, the pre-trained LLMs used in this experiment contain 2.6 to 9.3 billion parameters, which is several tens of times larger than the bert-base models used in the baselines. However, the numbers of trainable LoRA parameters are relatively small, ranging from 10 to 27 million. This suggests that LoRA-based SFT effectively leverages the capabilities of large, fixed-weight networks for dependency parsing tasks. Moreover, the parsing accuracy appears to depend on the number of pre-training parameters, given a certain number of trainable parameters.

4.4 Evaluation in 17 languages

Monolingual SFT. We evaluated the proposed method in 17 UD languages to assess its monolingual performance. Table 4 shows the detailed results for each language.

The proposed method achieved the highest LAS in all 17 languages, and the highest UAS in 16, except Norwegian, indicating its overall effectiveness. Among the open LLMs, gemma-2-9b demonstrated consistently strong performance, ranking first in 16 languages with the sole exception of Arabic. Due to lower tokenization efficiency in ar_padt compared to other languages, the LLMs occasionally failed to output the complete analysis results within the available context length, particularly for long sentences. However, the Llama-3.1 tokenizer was approximately 20% more efficient at tokenizing Arabic text than the gemma-2 and Qwen2.5 tokenizers, which contributing to higher accuracy. This indicates a trade-off between efficiency and accuracy: as the number of Chain-of-Thought steps increases, the allowable input sentence length becomes more constrained by the maximum context length of the LLMs.

[9] https://ufal.mff.cuni.cz/udpipe/2/models

[10] https://github.com/rycolab/parsing-as-tagging

[11] https://github.com/crux82/u-deppllama

[12] The publicly available implementation of U-DepPLLaMA uses the precision as the accuracy, but we followed the UD convention and used F1-measure as the accuracy.

[13] https://platform.openai.com/docs/models/gpt-4o-mini

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

[15] https://huggingface.co/google/gemma-2-2b

[16] https://huggingface.co/Qwen/Qwen2.5-7B

[17] https://huggingface.co/meta-llama/Llama-3.1-8B

[18] https://huggingface.co/google/gemma-2-9b

Dataset	gemma-2-9b 17-multi		
	UPOS	UAS	LAS
el_gdt	93.1	92.4	86.3
he_htb	90.1	83.3	70.0
hi_hdtb	76.9	72.9	55.2
hu_szeged	87.1	85.7	75.1
id_gsd	86.5	82.4	66.6
pt_gsd	92.2	86.4	78.0
sv_talbanken	92.9	89.6	82.1
tr_imst	79.2	67.9	53.5
vi_vtb	81.8	72.3	57.7
zh_gsd	97.2	89.5	87.2

Table 5: Evaluation results of our multilingual model on UD r2.15 datasets not used for training.

Multilingual SFT. An additional advantage of the proposed method is its compatibility with multilingual training. The `gemma-2-9b 17-multi` model achieved comparable or higher accuracy than its monolingual counterparts, except in Czech and Russian, likely due to the down-sampling.

Table 5 shows the evaluation results on 10 languages not included in the training data for `17_multi`. Among these, Greek and Swedish exhibited relatively high performance, indicating successful generalization from typologically or linguistically related languages. This highlights the model's ability to generalize across languages, a key strength of our method.

4.5 Analysis

Error analysis. We conducted an error analysis on Simplified Chinese, which showed the lowest UAS in monolingual evaluation. Errors were primarily concentrated in nouns (27.8%), verbs (24.8%), and punctuation marks (16.1%) for `gemma-2-9b`. Most of these errors occurred in sentences containing multiple independent clauses—a structure more frequent in Chinese than in many other languages. Due to the structural parallelism among these clauses, an output that differs from the gold annotation is not necessarily incorrect.

Figure 2 illustrates an example that includes noun, verb, and punctuation errors, highlighting the challenge of analyzing paratactic structures with minimal syntactic markers.

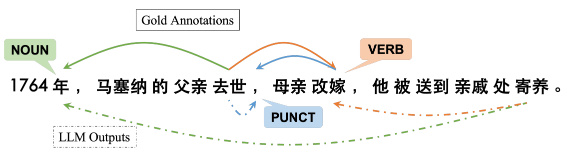


Figure 2: An example illustrating common errors for Chinese dependency parsing.

Performance in other tasks. An LLM fine-tuned for dependency parsing clearly performs worse on other tasks, even if the base model has been instruction-tuned. This performance degradation in general tasks may be mitigated or even reversed by fine-tuning the model on the dependency parsing task simultaneously with other instruction-tuning datasets (Asada and Miwa, 2025); however, experimental verification remains a future challenge.

4.6 Unimplemented UD tasks

Tokenization. In the early stages of this work, we evaluated LLM-based word segmentation by inserting a word segmentation step at the beginning of step-by-step instructions. However, particularly for Japanese, the segmentation accuracy was significantly lower than that of commonly used morphological analyzers. To address this issue, full-parameter LLM training, including the word embedding layer, on large-scale training data would be necessary. However, the associated cost could be several orders of magnitude higher than that of LoRA-SFT, which is employed in this study. Thus, an efficient method for training word segmentation criteria tailored to LLMs is still required.

Lemmatization. Lemmatization has traditionally relied on dictionaries and heuristic rules; however, end-to-end approaches have recently gained traction (Qi et al., 2020). LLMs may also be capable of effectively selecting the appropriate normalized form from a range of synonymous expressions or character variants by leveraging the knowledge acquired through large-scale pre-training, although this remains to be empirically validated.

Morphological features. The Universal Features[19] inventories over 200 lexical and inflectional features designed to classify word properties. Decoder-based classifiers offer significant advantages for simultaneously classifying this large number of features, whereas using generative models such as LLMs is relatively inefficient.

5 Conclusions

We proposed a novel step-by-step prompting strategy for LLM-based dependency parsing using a simple tabular format, achieving improved output validity and parsing accuracy across 17 languages. Multilingual SFT often outperformed monolingual models and generalized well to unseen languages.

[19] https://universaldependencies.org/u/feat/

Acknowledgments

This work was conducted as part of a collaborative research project between Recruit Co., Ltd. and the National Institute for Japanese Language and Linguistics. We are grateful to all those involved in the management and support of this project. We would also like to express our sincere gratitude to Yuji Matsumoto of RIKEN AIP for his valuable advice from the early stages of this research. Finally, we thank the anonymous reviewers for their constructive and detailed comments.

References

Afra Amini, Tianyu Liu, and Ryan Cotterell. 2023. Hexatagging: Projective dependency parsing as tagging. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1453–1464, Toronto, Canada. Association for Computational Linguistics.

Masaki Asada and Makoto Miwa. 2025. Improving relation extraction by sequence-to-sequence-based dependency parsing pre-training. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7099–7105, Abu Dhabi, UAE. Association for Computational Linguistics.

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

Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2025. Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57(6):1–39.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324, Brussels, Belgium. Association for Computational Linguistics.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Chunpeng Ma, Lemao Liu, Akihiro Tamura, Tiejun Zhao, and Eiichiro Sumita. 2017. Deterministic attention for sequence-to-sequence constituent parsing. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3237–3243. AAAI Press.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043. European Language Resources Association.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108. Association for Computational Linguistics.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Oriol Vinyals, Łukasz 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, page 2773–2781. Curran Associates, Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22.

A Dataset Statistics

Statistics for the Universal Dependencies treebanks used in the experiments are shown in Table 6.

Language	Dataset	Train number of sentences	Train number of tokens	Dev number of sentences	Dev number of tokens	Test number of sentences	Test number of tokens
UD r2.2:							
English	en_ewt	12,543	204,585	2,002	25,148	2,077	25,096
UD r2.15:							
Arabic	ar_padt	6,075	223,881	909	30,239	680	28,264
Bulgarian	bg_btb	8,907	124,336	1,115	16,089	1,116	15,724
Catalan	ca_ancora	13,123	429,578	1,709	58,073	1,846	59,610
Czech	cs_pdt	68,491	1,173,285	9,270	159,283	10,146	173,918
German	de_gsd	13,814	263,791	799	12,480	977	16,498
English	en_ewt	12,544	204,579	2,001	25,149	2,077	25,094
Spanish	es_ancora	14,287	453,039	1,654	53,476	1,721	53,622
French	fr_gsd	14,450	354,652	1,476	35,721	416	10,018
Italian	it_isdt	13,121	276,014	564	11,907	482	10,417
Japanese	ja_gsd	7,050	168,333	507	12,287	543	13,034
Korean	ko_gsd	4,400	56,687	950	11,958	989	11,677
Dutch	nl_alpino	12,289	186,027	718	11,541	596	11,046
Norwegian	no_bokmaal	15,696	243,886	2,409	36,369	1,939	29,966
Romanian	ro_rrt	8,043	185,125	752	17,073	729	16,324
Russian	ru_syntagrus	69,630	1,204,640	8,906	153,325	8,800	157,718
Slovenian	sl_ssj	10,903	215,155	1,250	26,500	1,282	25,442
Simplified Chinese	zh_gsdsimp	3,997	98,616	500	12,663	500	12,012

Table 6: Statistics of Universal Dependencies treebanks used in SFT experiments.

B Contamination Verification

A major concern in LLM-based evaluation is the contamination of testing data (Shokri et al., 2017; Das et al., 2025). To address this, we employed two diagnostics: (1) observing learning curves on `UD_English-EWT r2.15` to detect unusually high initial performance, and (2) comparing fine-tuning results using training-only vs. training + test data. Evaluation results for contamination verification are presented below.

Learning curves. Prior to the analysis, the learning curves of token recall (Figure 3) show that `gpt-4o-mini` is able to generate outputs with correct formats in very early stage, while other models need to be trained, and the learning curves of token recall after recovery (Figure 4) indicates our simple recovery algorithm works effectively.

For the learning curves of UPOS recall (Figure 5), the similarity between Figure 4 and Figure 5 suggests that the UPOS tagging task is one of the abilities that has been acquired in advance in these LLMs, which is also indicated by the high initial accuracy of the precision-based learning curves of UPOS in Figure 8.

In contrast, the gradual learning curves for HEAD and DEPREL identification (Figures 6 and 7) indicate the necessity of SFT for learning the knowledge for dependency parsing.

Overall, we conclude that the tested LLMs do not exhibit potential contamination in syntactic head and relational label identification tasks, despite possible prior exposure to the UPOS tagging task.

Effect of additional training on test set. In Table 7, when the testing data was included in training, all models naturally achieved certain accuracy gains (>+0.7 for UPOS, >+2.3 for LAS). This indicates a low possibility of contamination for the test set of `UD_English-EWT r2.15`.

	UPOS train	UPOS train + test (± diff)	LAS train	LAS train + test (± diff)
`gpt-4o-mini-2024-07-18`	98.3	99.3 (+1.0)	93.2	97.1 (+3.9)
`Qwen2.5-7B`	97.1	98.5 (+1.4)	88.7	93.6 (+4.9)
`Llama-3.1-8B`	97.2	98.7 (+1.5)	88.9	94.1 (+5.2)
`gemma-2-9b`	98.0	98.7 (+0.7)	91.7	94.0 (+2.3)

Table 7: Difference of UPOS and LAS scores between SFT on training data only and training + test data for `UD_English-EWT r2.15`.

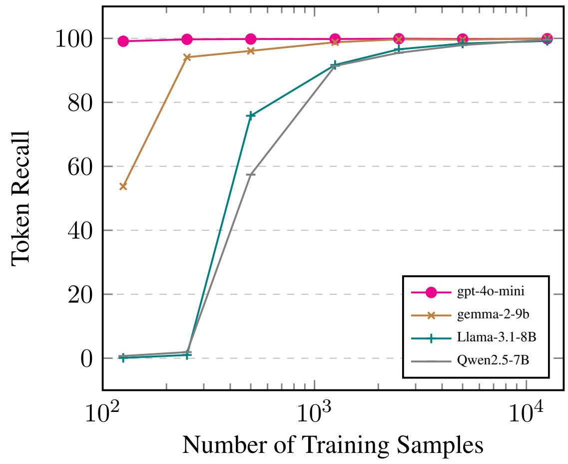


Figure 3: Learning curve - token recall.

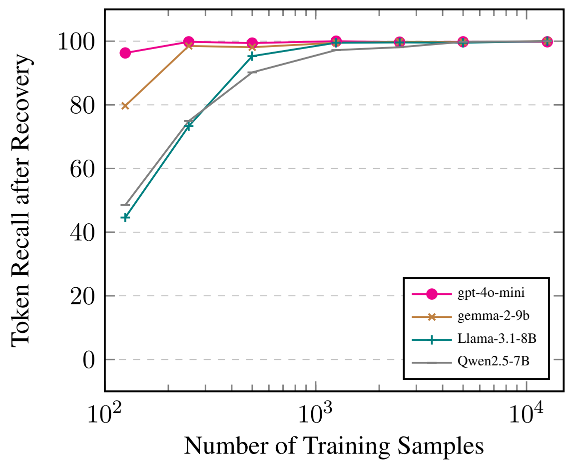


Figure 4: Learning curve - token recall after recovery.

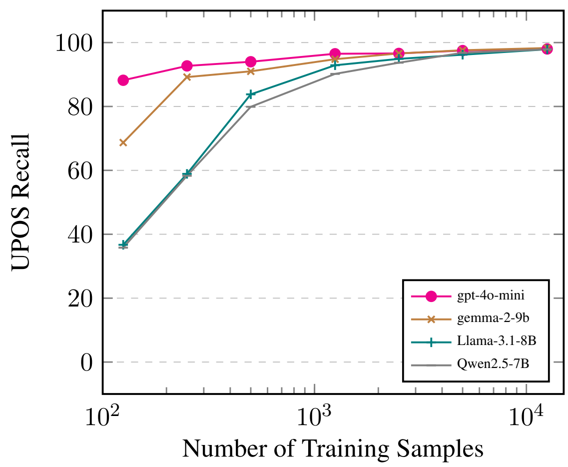


Figure 5: Learning curve - UPOS recall.

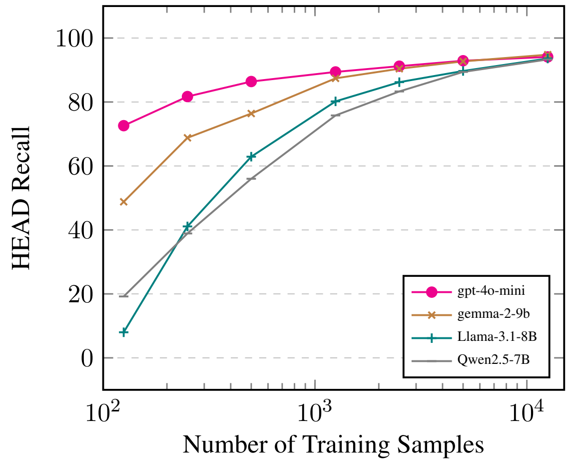


Figure 6: Learning curve - HEAD recall.

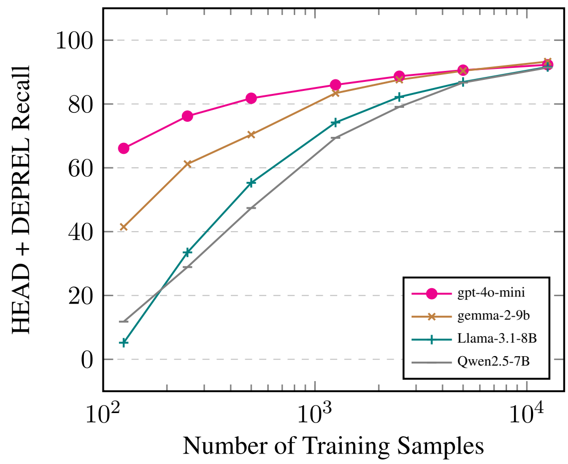


Figure 7: Learning curve - HEAD+DEPREL recall.

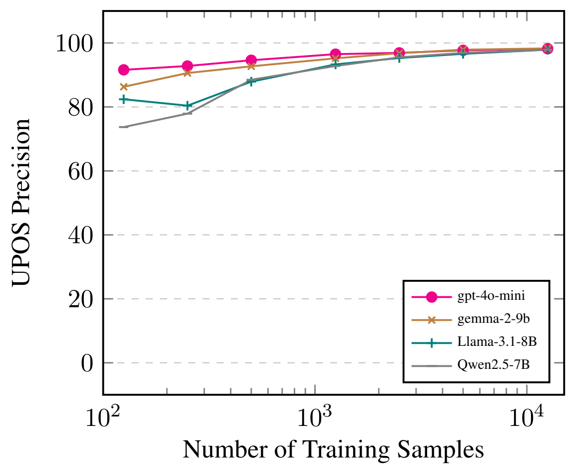


Figure 8: Learning curve - UPOS precision.

C Prompt Examples

The examples of Chain-of-Thought instruction prompts used in the experiments are shown in Figure 9 and Figure 10.

```
[system]
You are an English linguist and specialize in
English dependency analysis based on Universal
Dependencies.

[user]
We will now perform dependency parsing on English
sentence. After splitting the input sentence into
words as shown below, create a TSV with five fields:
word index from 1 to 6 + word +  the Universal
Dependencies part of speech + the dependent word
index + the Universal Dependencies relation.
However, for the word that is the main predicate of
the sentence, the dependent word index should be 0.

input sentence:
I'm just speculating now.

words:
I
'm
just
speculating
now
.

[assistant]
1 I  PRON 4 nsubj
2 'm  AUX 4 aux
3 just  ADV 4 advmod
4 speculating  VERB 0 root
5 now ADV 4 advmod
6 .  PUNCT 4 punct
```

Figure 9: Example for 1-step parsing instruction.

```
[system]
You are an English linguist and specialize in
English dependency analysis based on Universal
Dependencies.

[user]
We will now perform dependency parsing on English
sentence. After splitting the input sentence into
words as shown below, execute following three
tasks:

- Task 1
Create a TSV with three fields: word index from 1
to 6 + word + part of speech.

- Task 2
Add a field for the dependent word indexes to
each row to the output of Task 1. However, for
the word that is the main predicate of the
sentence, the dependent word index should be 0.

- Task 3
Add a field for the Universal Dependencies
relation labels to the output of Task 2.

input sentence:
I'm just speculating now.

words:
I
'm
just
speculating
now
.

[assistant]
- Task 1
1 I  PRON
2 'm  AUX
3 just  ADV
4 speculating  VERB
5 now ADV
6 .  PUNCT

- Task 2
1 I  PRON 4
2 'm  AUX 4
3 just  ADV 4
4 speculating  VERB 0
5 now ADV 4
6 .  PUNCT 4

- Task 3
1 I  PRON 4 nsubj
2 'm  AUX 4 aux
3 just  ADV 4 advmod
4 speculating  VERB 0 root
5 now ADV 4 advmod
6 .  PUNCT 4 punct
```

Figure 10: Example for 3-step parsing instruction.