LLM Dependency Parsing with Symbolic Rules

Michael Ginn and Alexis Palmer
University of Colorado
michael.ginn@colorado.edu

Abstract

We study whether incorporating symbolic rules can aid large language models in dependency parsing. We consider a paradigm in which LLMs first produce symbolic rules given fully labeled examples, and the rules are then provided in a subsequent call that performs the actual parsing. In addition, we experiment with providing human-created annotation guidelines in-context to the LLMs. We find that while both methods for rule incorporation improve zero-shot performance, the benefit disappears with a few labeled in-context examples.

1 Introduction

Dependency parsing is a classic task in natural language processing, requiring systems to parse complex linguistic structures. Standard approaches to dependency parsing train neural models on large amounts of labeled data (human-created parses), which either cast parsing as a word-by-word classification task (Covington, 2001), a sequence-to-sequence generative task (Li et al., 2018; Lin et al., 2022a), or a graph-based structure prediction task (Dozat and Manning, 2017).

One limitation of neural parsing methods is the requirement for large amounts of labeled data, which is often unavailable for low-resource languages. While Universal Dependencies (de Marneffe et al., 2021) currently includes over 150 languages, many of these languages have less than one thousand labeled tokens worth of data. Cross-lingual transfer has been proposed as a method to overcome these limitations (Guo et al., 2016; Schuster et al., 2019), but faces challenges due to differences in linguistic structure across languages (Ahmad et al., 2019; Wu and Dredze, 2020).

Recent work has investigated the use of large language models (LLMs) as an approach for parsing tasks (Li et al., 2023; Bai et al., 2023; Lin et al., 2023; Blevins et al., 2023; Tian et al., 2024).[1] LLMs are pretrained on huge multilingual corpora, and can potentially leverage cross-lingual information for effective parsing, even on rarer languages. Generally, this research has found that LLMs are effective zero-shot parsers on common languages such as English and Chinese and can also be effective on rare languages through in-context learning. However, in-context learning quickly grows inefficient and expensive with many examples.

We explore an alternate paradigm for dependency parsing on low-resource languages with LLMs. In our system, the LLM first acts as a descriptive linguist, observing labeled examples and producing linguistic rules. Then, the rules are provided in another LLM call where parsing is performed. We consider several techniques for providing context to the LLM during rule generation.

We hypothesized that explicitly producing symbolic rules could help improve the robustness of LLM-based parsing. We find that incorporating the LLM-generated rules offers clear improvements over the zero-shot setting, but worse performance than few-shot prediction. Furthermore, our best LLM-based setting underperforms state-of-the-art approaches across languages. We explore a number of failure cases and suggest future methods that could be used to address them. Our code and full results are available on GitHub.[2]

2 Related Work

Recent work has explored the potential of large language models for syntactic parsing. Some work has finetuned language models on dependency parsing as a sequence-to-sequence task (Hromei et al., 2024) or proposed statistical methods to automatically extract dependencies from language models

[1] While this work largely focuses on constituency parsing, we assume it shares similarities with dependency parsing.

[2] https://github.com/michaelpginn/ai-researcher-project

Proceedings of the 1st Joint Workshop on Large Language Models and Structure Modeling (XLLM 2025), pages 186–196

(Chen et al., 2024). The most similar work explores prompting-based methods for LLM parsing, that do not require training. Lin et al. (2023) evaluates LLMs on zero-shot parsing for English and Chinese. Tian et al. (2024) proposes similar strategies for constituency parsing in English. Ezquerro et al. (2025) benchmarks dependency parsing performance across four languages and many LLMs. However, using LLMs to parse rare languages remains unexplored.

3 Data

We use data from Universal Dependencies (UD) (de Marneffe et al., 2021) for eight languages with differing geographic regions, linguistic features, and resource availability. We use the pre-defined train/eval/test splits from UD when available, and otherwise produce our own splits which are reused across experiments. We also remove examples with non-projective dependencies, which can pose issues for transition-based parsers. We summarize our languages and splits in Table 1.

Language (code)	Train	Dev	Test
Bambara (bam)	697	149	150
Bhojpuri (bho)	220	47	48
Cantonese (yue)	620	133	133
Erzya (myv)	1429	306	307
Kiche (quc)	655	141	141
Komi Zyrian (pcm)	438	94	94
Nigerian Pidgin (pcm)	6352	826	797
Yoruba (yor)	182	39	40

Table 1: Languages used in this study, and the number of train, development, and test instances for each language. All data comes from Universal Dependencies (de Marneffe et al., 2021).

4 Experimental Conditions

4.1 Baseline

As a baseline, we simply prompt the LLM to generate dependency parses for a given example. Details about prompts are given in Appendix A. We use a truncated form of the CONLL-U format[3] where each tab-separated line gives an ID, a word, the ID of the word's head, and the dependency relation type. All of our main experiments use GPT-4o[4].

In addition, we use a baseline setting where we specify the list of allowed dependency relation types (obtained by collecting all relation types from the training data). We refer to this setting as LABELS.

4.2 Symbolic Rules

Dependency parsing is a symbolic task equivalent to forming directed edges on a graph.[5] In this study, we seek to understand whether symbolic knowledge can be extracted and leveraged for this task. In particular, symbolic rules and heuristics can be used by LLMs simply by providing the rules in-context, unlike traditional neural parsers. We consider three settings for incorporating rules.

Rule Writing In the RULE WRITING setting, we first provide five labeled examples[6] to the LLM and prompt it to generate rules. The rules are specified by predicting part-of-speech categories of words, and then by writing dependency rules. For a hypothetical English example, the LLM might predict the categories:

```
Det: the
Noun: dog, cat
Verb: chases
Punct: .
```

Then, the LLM writes dependency rules by extracting the dependency relations from provided examples. The predicted rules for the prior example might look like the following:

```
Noun -> Det (det)
Verb -> Noun (nsubj)
Root -> Verb (root)
Verb -> Noun (obj)
Verb -> Punct (punct)
```

Finally, the generated rules are provided as-is in a subsequent prompt to the LLM for performing parsing.

Word Contexts We note that other than the prediction of word categories, these rules are largely just descriptive analysis of observed examples. In the WORD CONTEXTS setting, we eliminate the need for identifying part-of-speech categories, and instead just record the contexts that a word can occur in, consisting of the type and head of a dependency relation pointing to the word. For example, in the previous example, we might have the following contexts, listed as "(head, relation type)":

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

[4] Specifically the GPT-4O-2024-08-06 checkpoint

[5] With some restrictions, of course

[6] We select relevant examples using the method described in subsection 4.3

```
the:
(dog, det)
(cat, det)

dog:
(chases, nsubj)

chases:
(root, root)
...
```

We collect these contexts from the examples in the training dataset. Since some words may occur in a huge number of contexts, we sample up to two contexts for a given word and relation type. These contexts are less generalized than the rules of the prior section, but more accurate, as they do not require predicting part-of-speech categories.

Guidelines Dependency parsing is typically conducted by human annotators, and as such there already exist detailed parsing guidelines for many languages. In theory, these guidelines should be sufficient for someone with a baseline knowledge of the language's vocabulary and syntax to perform accurate parsing. In the GUIDELINES setting, we provide these guidelines directly to the LLM, avoiding the potential error of LLM-based rule extraction. The human guidelines should be highly accurate and relevant to the task at hand, thus we expected this setting to have clear benefits.

We scrape guidelines from the appropriate Universal Dependencies webpage, convert to markdown, and remove links. An excerpt from the processed Kiche guidelines is given in Appendix B.

4.3 In-context examples

Prior research has indicated that providing in-context examples is vital to enabling LLMs to perform tasks in rare languages (Lin et al., 2022b; Cahyawijaya et al., 2024; Ginn et al., 2024). Thus, we compare the four settings described previously (LABELS, RULE WRITING, WORD CONTEXTS, GUIDELINES) across a zero-shot setting, a three-shot setting, and a five-shot setting. This comparison allows us to measure the effects of these strategies compared to the effect of increasing in-context examples.

As in Ginn et al. (2024), we select relevant examples to the target sentence by choosing the sentences in the training set with the highest chrF++ score, computed using the target sentence as the reference. This ensures that the in-context examples have high substring overlap with the target example, and are more likely to be relevant for parsing.

5 Experimental Results

We report our results on the development set, using GPT-4o, in Figure 1. We average UAS and LAS scores over languages; full results are available in our GitHub repo. Generally, we observe a clear trend where providing symbolic information helps in the zero-shot setting, but the benefit decreases with increased in-context examples. At the five-shot setting, any benefits from symbolic knowledge are effectively nullified. Next, we analyze the effects of the various settings. We also perform additional variational experiments on the use of labels and chain-of-thought prompting in Appendix D.

Effect of Rule Writing We observe a large benefit to both UAS and LAS in the zero-shot setting, and a smaller benefit in the three-shot setting. Recall that the rules were written using five relevant examples. Thus, it is not terribly surprising that performance is similar when using the rules versus using the examples directly. In fact, the similar performance indicates that the LLM can effectively compress the relevant information from the in-context examples into symbolic rules, providing the same benefit with far less text.

While this is less encouraging for improving absolute performance, it could be useful for improving efficiency. For example, an LLM could be used to write rules about many similar groups of sentences ahead of time, and the relevant rules could be selected during inference, drastically reducing the length (and thus, speed and cost) of the parsing prompt.

We perform manual qualitative examination of rules in Bambara. We observe two common failure cases.

First, in some cases the LLM produced overly-specific part-of-speech categories that prevented the correct relation from being predicted. For example, in one case the correct root verb was "ye". However, in the provided examples, "ye" was only ever used as an auxiliary verb, leading the LLM to predict that "ye" was a member of a category named AUX. Because the LLM failed to identify that auxiliary verbs were the same category as verbs, it then failed to predict "ye" as the root of the sentence, instead choosing a random noun as

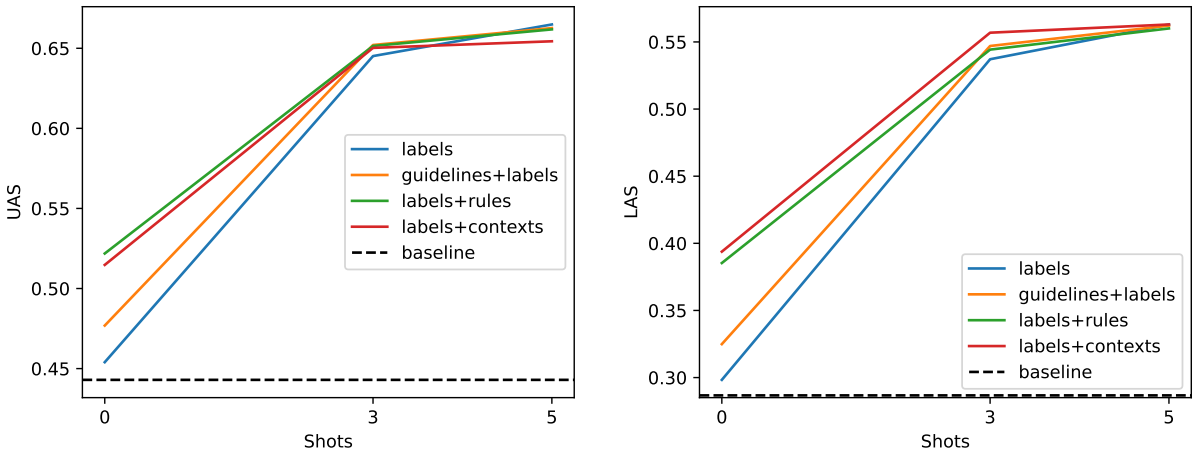

Figure 1: UAS (left) and LAS (right) scores for LLM-based dependency parsing, averaged across eight languages, across different numbers of in-context examples and different strategies for incorporating symbolic knowledge. We find that in the zero-shot setting, symbolic knowledge can provide clear improvements, but the margin disappears with sufficient in-context examples. The BASELINE setting is a zero-shot setting with no information provided at all.

the root.

Second, the LLM sometimes produced correct rules which could produce multiple possible parses for the target sentence. For example, one instance began with the sentence "n na kònò to n bolo!". The LLM correctly identified "n" and "kònò" as a pronoun and noun, respectively, and produced the correct rules PRONOUN -> PARTICLE (CASE) and NOUN -> PARTICLE (CASE). The LLM predicted a case relation from "kònò" to "na", rather than the correct relation from "n" to "na", both of which were allowed under the specified rules. This reveals a limitation of this sort of dependency rule: by not specifying word ordering, ambiguous situations arise with multiple valid parses. We explore one solution to this issue in Appendix C

Effect of Word Contexts The inclusion of word contexts (scraped directly from training data) had comparable performance to the LLM-written rules, with the highest LAS scores of any setting. Because this setting does not require an initial LLM call (unlike the rule writing setting), it drastically reduces the total cost of inference, while meeting or exceeding the performance of the RULE WRITING strategy.

A possible interpretation is that the only relevant information extracted by symbolic rules is the contexts in which particular words occur. Generalized symbolic rules over word categories do not seem to be particularly helpful by comparison.

We run paired bootstrap resampling (Koehn, 2004) to test the significance of the improvements of WORD CONTEXTS over the LABELS setting. We report the average significance score for the zero, three, and five-shot settings in Table 2. The results reinforce our qualitative observations.

Shots	Confidence
0	98.0%
3	61.3%
5	43.9%

Table 2: Paired bootstrap resampling score for the WORD CONTEXTS setting versus the LABELS setting, ran with 1000 iterations and test sets of 20 items.

Effect of Guidelines We expected GUIDELINES to be the most effective setting, as they provide the information that was used by human annotators to produce the labeled examples. However, we observe that while the GUIDELINES setting does provide small benefits over LABELS in the zero- and three-shot settings, it underperforms the RULE WRITING and WORD CONTEXTS strategies by a good margin.

There are two possible interpretations of this result. One possibility is that while the guidelines provide sufficient information to perform parsing, the LLM failed to understand and apply this information. This would align closely with the results of Aycock et al. (2025), which finds that LLMs struggle to utilize language reference materials when performing translation with rare languages.

The other possibility is that the guidelines do not actually provide sufficient information to per-

Method	bam	bho	yue	myv	quc	kpv	pcm	yor
mBERT	82.8 / 78.9	69.9 / 61.0	73.3 / 66.4	76.8 / 67.2	84.1 / 77.4	63.3 / 48.1	91.7 / 88.8	62.5 / 51.6
XLM	84.0 / 79.8	72.2 / 62.7	**76.0 / 70.4**	**79.6 / 69.8**	82.3 / 74.7	67.3 / 53.9	**93.7 / 90.9**	64.9 / 53.2
UDPipe 2	**92.4 / 90.2**	**76.9 / 68.0**	75.8 / 70.2	77.6 / 69.1	**88.6 / 84.1**	74.9 / 65.6	93.0 / 89.7	**76.0 / 69.5**
LLMs w/ labels and contexts, 5-shot								
GPT-4o	69.3 / 61.7	67.4 / 55.5	71.8 / 65.1	54.7 / 44.5	81.0 / 72.7	49.5 / 37.7	73.0 / 68.2	54.2 / 45.5
Gemini	76.0 / 70.6	67.9 / 56.3	68.4 / 63.0	77.4 / 69.6	87.7 / 82.9	**75.0 / 65.9**	72.8 / 68.6	59.1 / 48.9
Cmd R+	54.3 / 47.9	60.9 / 52.7	51.9 / 45.8	47.8 / 36.1	69.7 / 61.1	46.8 / 34.3	50.9 / 46.2	41.2 / 32.3
Llama 3.1	41.5 / 34.1	58.9 / 50.4	37.8 / 31.6	35.8 / 25.5	54.6 / 44.6	38.6 / 26.4	38.9 / 32.0	39.5 / 32.2

Table 3: Test set results on various state-of-the-art methods and LLMs using our best method from the preceding section. Scores are reported as UAS / LAS.

form parsing. Certainly, these guidelines do not include complete bilingual dictionaries, so an LLM which cannot translate words in the target language would likely struggle to apply more sophisticated grammar rules. This could be studied in future work by also providing word-by-word translations alongside guidelines. However, it may also be the case that the UD guidelines do not specify all of the information needed for parsing, assuming some knowledge of the grammar of the language.

6 Baseline Comparison

6.1 Baselines

We consider the following baseline models, which represent the common approaches used for dependency parsing and often are around the state-of-the-art, depending on the dataset.

Transition Parser We use a neural transition-based parser following the approach of Covington (2001); Nivre (2003); Jurafsky and Martin (2025). The parser predicts actions to form arcs between words, processing words in the sentence one-by-one and using a stack to retain words until they have been fully processed. In order to potentially benefit from crosslingual transfer, we finetune our classifier using two pretrained multilingual model, mBERT (Devlin et al., 2019) and XLM-RoBERTa (Conneau et al., 2020).

UDPipe We use UDPipe 1 (Straka and Straková, 2017), a pipeline that performs tokenization, lemmatization, tagging, and parsing, with trainable components for each step.[7] We train models using the default hyperparameters.

6.2 Results

We report results in Table 3. While the various settings for LLM inference were similar in the five-shot setting, we select the WORD CONTEXTS setting to compare, as it performed best in the zero- and three-shot settings. We run this setting with the following LLMs:

- GPT-4o, as in the development experiments (OpenAI, 2024)
- Gemini 2.0 Flash (Gemini Team, 2024)
- Command R+[8], a 104B parameter model specifically designed for low-resource multilingual tasks
- Llama 3.1 7b (Dubey et al., 2024), using the 8-bit quantization and the MLX (Hannun et al., 2023) checkpoint

We observe that for most languages, the LLM-based method underperforms or matches traditional neural SOTA methods. Of the four models tested, Gemini performs best on average. While the paradigms studied here can certainly improve performance over the zero-shot setting, they are not sufficient to beat the best prior approaches.

7 Conclusion

We studied methods for performing dependency parsing on low-resource languages with large language models (LLMs) that incorporate (symbolic) rules. We compared using LLM-written rules, extracting contexts that words appear in, and providing human-readable annotation guidelines. Overall, we found that these methods provide benefits in the zero-shot setting, but with sufficient in-context examples, their benefit was minimal. We evaluated several LLMs against state-of-the-art baselines, finding that the LLMs were unable to beat the best prior models.

[7] We chose not to use UDPipe 2 as it proved impossible to replicate the necessary development environment

[8] https://docs.cohere.com/docs/command-r-plus

8 Acknowledgements

The original idea for this work was produced by the AI ideation system of Si et al. (2024), and this paper originated from research conducted in the associated meta-study.

References

Wasi Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. 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 2440–2452, Minneapolis, Minnesota. Association for Computational Linguistics.

Seth Aycock, David Stap, Di Wu, Christof Monz, and Khalil Sima'an. 2025. Can LLMs really learn to translate a low-resource language from one grammar book? In *The Thirteenth International Conference on Learning Representations*.

Xuefeng Bai, Jialong Wu, Yulong Chen, Zhongqing Wang, and Yue Zhang. 2023. Constituency parsing using llms. *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. Association for Computational Linguistics.

Samuel Cahyawijaya, Holy Lovenia, and Pascale Fung. 2024. LLMs are few-shot in-context low-resource language learners. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 405–433, Mexico City, Mexico. Association for Computational Linguistics.

Junjie Chen, Xiangheng He, and Yusuke Miyao. 2024. Language model based unsupervised dependency parsing with conditional mutual information and grammatical constraints. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6355–6366, Mexico City, Mexico. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Michael A Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, volume 1. Athens, GA.

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

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, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and alia. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ana Ezquerro, Carlos Gómez-Rodríguez, and David Vilares. 2025. Better benchmarking llms for zero-shot dependency parsing. *arXiv preprint arXiv:2502.20866*.

Gemini Team. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Michael Ginn, Mans Hulden, and Alexis Palmer. 2024. Can we teach language models to gloss endangered languages? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5861–5876, Miami, Florida, USA. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *AAAI Conference on Artificial Intelligence*.

Awni Hannun, Jagrit Digani, Angelos Katharopoulos, and Ronan Collobert. 2023. MLX: Efficient and flexible machine learning on apple silicon.

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

Daniel Jurafsky and James H. Martin. 2025. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*,

3rd edition. Online manuscript released January 12, 2025.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

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

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.

Boda Lin, Zijun Yao, Jiaxin Shi, Shulin Cao, Binghao Tang, Si Li, Yong Luo, Juanzi Li, and Lei Hou. 2022a. Dependency parsing via sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7339–7353, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Boda Lin, Xinyi Zhou, Binghao Tang, Xiaocheng Gong, and Si Li. 2023. Chatgpt is a potential zero-shot dependency parser. *Preprint*, arXiv:2310.16654.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. 2022b. Few-shot learning with multilingual generative language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9019–9052, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160, Nancy, France.

OpenAI. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. 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 1599–1613, Minneapolis, Minnesota. Association for Computational Linguistics.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2024. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. *Preprint*, arXiv:2409.04109.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Yuanhe Tian, Fei Xia, and Yan Song. 2024. Large language models are no longer shallow parsers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7131–7142, Bangkok, Thailand. Association for Computational Linguistics.

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 *Advances in Neural Information Processing Systems*.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

A Prompts

A.1 Parsing

The base prompt used across experiments is given below.

```
You are predicting the
   dependency parse for a
   sentence in $language.

You will be given a sentence
   word-by-word, with each word
    on a new line. Below is an
   example in English:

1       The
2       dog
3       chases
4       the
5       cat
6       .

You are to predict the
   dependency parse for this
   sentence. For each token,
   you should predict the
   following:
```

```
1. The index of the token's
   head according to its
   dependency relation, or "0"
   if it is the root
2. The type of dependency
   relation

You should output the
   dependency parse using the
   original format, with two
   additional columns (
   separated by tabs) for the
   head and relation type. For
   the example above, you
   should produce the following
   :

1       The     2       det
2       dog     3       nsubj
3       chases  0       root
4       the     5       det
5       cat     3       obj
6       .       3       punct

Do not output any additional
   text. Only produce the
   dependency parse following
   the above format.

Please gloss the following
   example in $language:

$target_example
```

For any settings with the label list included, we add the following:

```
The allowed dependency
   relations are the following:
    $label_list
```

For settings with few-shot examples, we add the following:

```
Below are some fully glossed
   examples in $language.

$examples
```

A.2 Rule Writing

The base prompt for writing rules is:

```
You are writing dependency
   grammar rules given a small
   number of examples. You will
    be provided with parsed
   sentences written word-by-
   word with each word on a new
    line. An example in English
    is given below:

1       The     2       det
2       dog     3       nsubj
3       chases  0       root
4       the     5       det
5       cat     3       obj
6       .       3       punct

The first column is the ID of
   the word. The third column
   is the ID of the head for
   the word. The fourth column
   is the type of dependency
   relation. From this sentence
   , you should first infer
   categories for each of the
   words and output them.
   Please output "Categories:"
   followed by your inferred
   categories, as in the
   following example:

Categories:
Det: the
Noun: dog, cat
Verb: chases
Punct: .

You should omit duplicate words
   , and words may belong to
   multiple categories.

Then, write dependency grammar
   rules using the convention "
   Head -> Dependent (relation
   type)", based on the
   observed rules in the data.
   Print all of the rules,
   seeking to find a minimal
   set of rules that explains
   the data, and starting with
   "Rules:". Do not repeat
   rules. The rules from the
   previous example are given
   below.
```

```
Rules:

Noun -> Det (det)
Verb -> Noun (nsubj)
Root -> Verb (root)
Verb -> Noun (obj)
Verb -> Punct (punct)

You are writing rules for
   $language. Please use the
   following examples to
   produce the analysis, making
    sure to include both the
   Categories and Rules
   sections.

$examples
```

B Example Guidelines

An excerpt from the Kiche guidelines is given below.

```
#### Nouns

- Most nouns are not inflected
   for number, although animate
    nouns can be, in this case
   they are annotated with `
   Number=Plur`.
- There is a subset of nouns
   used relationally, these are
    called relational nouns and
    are used where adpositions
   would be used in other
   languages.
  - They are marked with the
     feature `[NounType]()=
     Relat`.
  - The lemmas are: _ech_, _uk'
     _, _umal_, _wach_, _ib'_,
     _onojel_, _wi'_, _pam_,
     _ij_, _xe'_, _xo'l_,
     _tukel_, _tzalaj_, _naqaj_
     .
  - Relational nouns are also
     used for:
    - Reflexive, _ib'_
    - Introducing the agent in
       a passive, _umal_

#### Verbs

- Transitive verbs have
   polypersonal agreement which
    is indicated through
   layered features `Person[obj
   ]`, `Number[obj]`, `Person[
   subj]`, `Number[subj]`.
- Finite verbs have `Aspect`
   but no `Tense`.
  - The imperfective or
     incompletive is annotated
     with `Aspect=Imp`.
  - The perfective or
     completive is annotated
     with `Aspect=Perf`.
- Incorporated movement is
   indicated through the
   feature `Movement`:
  - Movement away from is
     marked with `Movement=Abl
     `, this is the morph _\-e
     '-_
  - Movement towards is marked
     with `Movement=Lat`, this
     is the morph _\-l-_
- There are two principle
   valency changing processes:
   Passive and antipassive.
   Both produce verbs with only
    set B agreement.
  - In the passive, annotated
     with `Voice=Pass`, the
     object is promoted to
     subject and the subject is
      demoted to oblique.
  - In the antipassive,
     annotated with `Voice=
     Antip`, the subject
     agreement is maintained
     and the object is demoted
     to oblique.
...
```

C Directional Rule Writing

In section 5, we identified an issue where multiple rules could apply to an example, and there was no way to disambiguate which rule to use. One trivial solution is to also specify an ordering between the head and dependent. We experiment with this idea, prompting the LLM to produce rules of the form:

```
Noun -> Det (det, left)
Verb -> Noun (nsubj, left)
```

```
Root -> Verb (root, none)
Verb -> Noun (obj, right)
```

In addition to the relation label, the LLM simply labels whether the dependent occurs to the left or right of the head. We report results for this variation of rule writing on the development set in Table 4. We observe very small improvements in most languages.

For the preceding example, the LLM now correctly identifies the rule PRONOUN -> PARTICLE (CASE, RIGHT) which should resolve the ambiguity. Unfortunately, the LLM now predicts the incorrect relation "nsubj", with no clear reason why (as this does not follow from the rules). Evidently, the inclusion of directions in the rules is not a clear benefit, but introduces other forms of error.

D Variational Experiments

Effect of Labels All of our main settings include a list of allowed relation labels in the prompt. While it is intuitive why this would be beneficial, we also provide empirical validation. In Figure 2, we report the results of zero-shot prediction with and without labels, across the baseline setting and the setting with guidelines included. We see a small improvement from including labels in not only the Labeled Attachment Score (LAS), but also the Unlabeled Attachment Score (UAS). As providing labels is inexpensive, we use this setting for all main experiments.

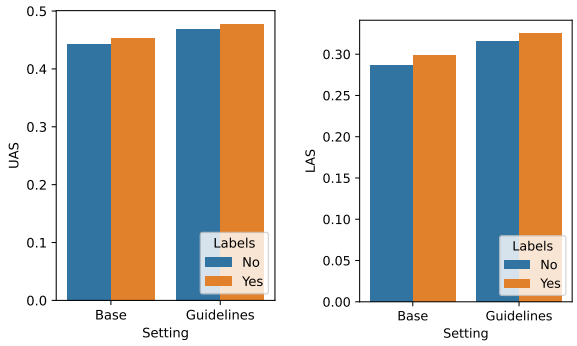

Figure 2: UAS (left) and LAS (right) scores for the zero-shot setting, comparing results when the list of allowed relation labels is included in the prompt versus when it is omitted. In both the base and GUIDELINES setting, we see a small improvement from including labels.

Effect of Chain-of-Thought Another applicable technique is chain-of-thought (CoT) prompting, where the LLM is prompted to produce step-by-step explanations of its thought process (Wei et al., 2022). CoT has proven effective on multistep reasoning problems, and thus is a good fit for the task of understanding and applying the information from in-context examples and in-context rules. We add CoT to the base five-shot setting as well as the five-shot setting with guidelines. We report these results in Figure 3. Unfortunately, adding CoT seems to worsen performance in both settings.

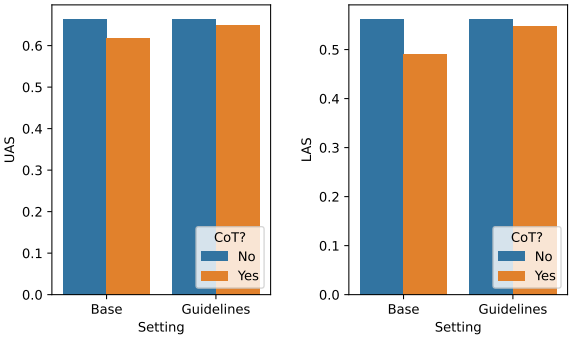

Figure 3: UAS (left) and LAS (right) dev set scores for the five-shot setting, evaluating the effect of adding chain-of-thought prompting. In both the base and GUIDELINES setting, we see a clear detriment from CoT.

Rules	bam	bho	yue	myv	quc	kpv	pcm	yor	mean
Base	61.7	53.4	59.6	**39.9**	72.7	38.3	71.7	50.7	56.0
+ order	**63.5**	**53.6**	**59.7**	38.0	**73.1**	**38.5**	**72.1**	**52.2**	**56.3**

Table 4: LAS scores across languages for the RULE WRITING setting. In the base setting, rules were written as "Head -> Dependent (relation type)" without any notion of word order. In the + ORDER setting, rules additionally included whether the dependent was to the left or right of the head.