Categorial Grammar Supertagging via Large Language Models

Jinman Zhao and Gerald Penn
Department of Computer Science
University of Toronto
Toronto, Canada
jzhao,gpenn@cs.toronto.edu

Abstract

Supertagging is an essential task in Categorical grammar parsing and is crucial for dissecting sentence structures. Our research explores the capacity of Large Language Models (LLMs) in supertagging for both Combinatory Categorial Grammar (CCG) and Lambek Categorial Grammar (LCG). We also present a simple method that significantly boosts LLMs, enabling them to outperform LSTM and encoder-based models and achieve state-of-the-art performance. This advancement highlights LLMs' potential in classification tasks, showcasing their adaptability beyond generative capabilities. Our findings demonstrate the evolving utility of LLMs in natural language processing, particularly in complex tasks like supertagging.

1 Introduction

In natural language processing (NLP), many downstream tasks utilize the syntax representation such as Categorial Grammar (CG) of sentences (Sugimoto and Yanaka, 2022; Tian and Song, 2022). Those syntax trees are typically obtained through a two-step process (Xu et al., 2015; Bhargava and Penn, 2020; Tian et al., 2020; Clark et al., 2018): tag assignment, and tree construction. The step for assigning tags to words is called **supertagging**.

One possible way to improve the performance of supertagging is using LLMs. LLMs (Brown et al., 2020; Touvron et al., 2023; Chowdhery et al., 2023) have experienced many breakthroughs in recent years and have been applied in various NLP tasks (Wan et al., 2023; Wang et al., 2023). However, their application to CG supertagging has received little attention. Indeed, a conventional point of view is that, because most LLMs are decoder-based, they are more suitable for generation tasks rather than classification tasks. Previous studies have shown that LLMs tend to underperform in classification tasks (e.g. Yu et al., 2023). Smaller encoder-based models, like RoBERTa (Liu et al., 2019), operating in supervised settings have demonstrated the ability to match or even surpass the performance of much larger and more resource-intensive decoder-based generative LLMs (Yu et al., 2023). Because CG supertagging is a classification task, modern techniques have only adopted encoder-based models (Tian et al., 2020; Kogkalidis and Moortgat, 2023; Yamaki et al., 2023). The capability of LLM-based supertagging methods has not been fully explored.

In this work, we study the ability of LLMs for CG supertagging. We compare different LLMs and present a novel method for improving their performance at supertagging tasks. Without applying our new method, we find that LLMs are less accurate than standard BiLSTM-based supertaggers. This is because the architecture of encoder/BiLSTM-based models allows them to extract features of context provided by subsequent words, making them more advantageous than LLMs. We find that this disadvantage of LLMs can be addressed by repeating the input sentences. This new method enhances the capability of decoder-based models in supertagging tasks. The main contributions of our works are:

- We evaluate supertagging tasks using different LLMs via in-context learning and fine-tuning and show that LLMs are good CCG/LCG supertagging reasoners.
- We propose a simple finetuning method to make Llama2 finetuning performance State-of-the-art.

2 Related Work

CCG Supertagging Bangalore and Joshi (1999) was the first work to introduce the term "supertag" for LTAG. The original CCG supertagger (Clark, 2002; Clark and Curran, 2007), employed a maximum entropy approach. Starting from Xu et al. (2015), researchers began to apply sequential neural networks to tag each lexicon, also eliminating

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 697–705

the need for POS tags as inputs. From RNNbased (Xu et al., 2015) to LSTM based (Lewis et al., 2016; Bhargava and Penn, 2020; Clark et al., 2018) and transformer based (Tian et al., 2020; Yamaki et al., 2023). Also, some works use graph structure (Kogkalidis and Moortgat, 2023; Prange et al., 2021). Yamaki et al. (2023) is the state-of-the-art formatting CCG as a recursive composition in a vector space. This work is more focused on parsing instead of supertagging, it needs the information of parsing during training while others only require sentences and supertags.

Large Language Models for NLP tasks Recently, there has been a trend toward using LLMs for NLP tasks. For instance, Wan et al. (2023) employs in-context learning on GPT-3 for Relation Extraction(RE). Wang et al. (2023); Xie et al. (2023) adapts LLMs to the Named Entity Recognition (NER) task. LLMs have also been employed for other tasks such as text summarization (Goyal et al., 2023) and sentiment analysis (Sun et al., 2023). However, Pangakis et al. (2023) argue that LLM enhances text annotation but requires human validation due to varying task effectiveness.

3 Method

In this section, we will show the methodologies employed to utilize LLMs for supertagging. The focus will be on two primary aspects: In-context learning and fine-tuning a generative LLM.

3.1 In-context Learning for CG Supertagging

For the prompts, we structure them into three main components:

1. **Task instruction**: This aims to leverage the LLM's capabilities for specialized tasks by providing clear instructions, context for understanding, and a directive for the expected output.

2. **Exemplar for few-shot learning:** For few-shot learning, it is essential to provide the LLM with several examples. These examples serve as a guide for the model, demonstrating how to perform the task.

3. **Input sentence that we want to do supertagging.**

Appendix A illustrates the detail of the prompt we use.

3.2 Fine-tuning for CG Supertagging

The specificity of CG supertagging highly relies on the order. For instance, S/NP indicates the necessity for an NP (noun phrase) to the right, whereas S\NP signifies the need for an NP to the left. Given this, the auto-regressive nature of LLMs, which process sentences solely from left to right once, may not be entirely sufficient for understanding the complexity of CG supertagging. From this perspective, we feed the input sequence to the model multiple times to enhance comprehension(note this only increases the length of each input, not the quantity of the test data). We use *"CG supertagging task."* as the instruction. Figure 1 is an example of a single training data. We fine-tuned LLM in a manner akin to traditional language model training, focusing on minimizing the cross-entropy loss.

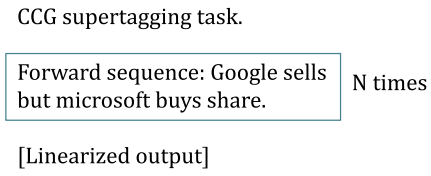

Figure 1: This is a demonstration of the text of each training data of LLM during fine-tuning.

The intuition behind extending the length of each training data by repeating the input text is rooted in the combinatorial nature of CG. In CG, each word's supertag is contextually related to both the preceding and subsequent words. This intricacy makes it essential to provide a model, especially in a decoder-only architecture, with an extended sequential context to adequately capture the relationships between a word and its surrounding words.

By doubling the input length, for example, we create a continuous, extended context that more naturally mirrors the combinatorial interactions inherent in CG.

3.3 Linearization

Overall, we use the sequential text that consists of *{category word}* pairs to represent the supertagging result. We also use two different methods to represent the categories.

Classification Based Supertagging can be regarded as a classification problem, so we have constructed a bijective mapping $ID()$ from categories to integers, using these integers to represent the various classes (i.e., categories). This method sim-

plifies the representation of a wide range of categories by assigning each unique category a specific integer, thereby facilitating easier processing and classification by models. Below is an example with $ID(NP) = 1$ and etc:

{1 Google} {2 sells} {3 but} {1 Microsoft} {2 buys} {1 shares} {4 .}

Generation Based While classification-based text aligns it with traditional classification problems, this approach has a limitation: it cannot generate new classes. Moreover, even if the model produces a new class represented by an integer, the numeric form does not convey what the specific category is. We also experimented with keeping the entire category in string form, as demonstrated in the following example:

{NP Google} {(S\NP)/NP sells} {conj but} {NP Microsoft} {(S\NP)/NP buys} {NP shares} {. .}

For ICL, we only feed generation-based exemplars because CG usually contains a massive number of categories(more than 1000). This can lead to difficulties for LLMs in distinguishing between the various classes.

Due to the space limitation, we put the comparison of two linearizations in Appendix F.0.1.

4 Result

4.1 Experimental Setup

Model Selection We use the following classical open and closed models:

- GPT-3.5 (Ouyang et al., 2022).
- GPT-4[1].
- Llama2 (Touvron et al., 2023) 7B/13B.

Dataset Selection For CCG we use CCGbank (Hockenmaier and Steedman, 2007)[2] and rebank (Honnibal et al., 2010) for evaluation. For LCG, we use LCGbank (Bhargava et al., 2024).

Statistics of CCGbank/CCGrebank/LCGbank are demonstrated in Appendix B.

Tuning Method and Hyperparameter We use LoRA (Hu et al., 2021) for tuning our model. Hyperparameters are listed in Appendix C. The trainable parameter of LoRA is much smaller than full fine-tuning.

[1] https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo

[2] https://catalog.ldc.upenn.edu/LDC2005T13

	Prompt	Acc	ReAcc
GPT-3.5	zero-shot	24.6	25.4
	five-shot	36.0	38.4
GPT-4	zero-shot	37.2	41.8
	five-shot	**53.2**	**57.3**

Table 1: ICL results for CCGbank via different LLMs. All numbers are in percentage.

4.2 In-context Learning Result

We have observed that LLMs do indeed possess a certain capacity for supertagging inference. Table 1 summarizes the performance of various LLMs. For accuracy, we employ an absolute accuracy metric, which reflects instances where the category generated by the LLM matches the ground truth exactly. We are interested in exploring whether LLMs can capture the syntax of sentences. Thus, we introduce a relative accuracy(ReAcc) metric(for ICL only). In calculating relative accuracy, we disregard the contents within "[]"; for example, we treat "NP[nb]/N" and "NP/N" as equivalent. This approach allows us to evaluate the models' understanding of syntactic structures. Although exact matches are ideal, a high degree of similarity in category prediction—especially in the context of CCG's extensive(more than 1000 class) system—still demonstrates significant supertag capability.

Additionally, in the ICL, GPT-4 significantly outperforms other models. Llama2 series models generated output either unreadable or nonsense. Appendix E contains the ICL output of Llama. So we did not put Llama2 results here. GPT-4 appears to possess a certain level of supertagging capability, achieving a 57.3% ReAcc rate in a 5-shot setting, which indicates its ability to capture the structure of a sentence. For the details of output cleaning, please refer to Appendix D.

4.3 Fine-tuning Result

4.3.1 Main Result

We demonstrate the main results in Table 2, where all Llama2-based models repeat the forward sequence 5 times and the output is generation-based. For LoRA, we set rank=256. We found that our method outperformed all previous supertaggers across these banks.

For CCGbank, the best case is Llama2-13B + LoRA. Fine-tuning this configuration required roughly 15 GPU hours on 2xA100s. Our word-

Model	Acc	LF
Clark et al. (2018)	96.1	-
Bhargava and Penn (2020)	96.00	90.9
Liu et al. (2021)	96.05	90.87
Prange et al. (2021)	96.09	-
Prange et al. (2021)	96.22	90.91
Tian et al. (2020)	96.25	90.58
Kogkalidis and Moortgat (2023)	96.29	-
Yamaki et al. (2023)	96.6	**92.12**
Llama2 7B + Full	96.1	-
Llama2 7B + LoRA	96.36	-
Llama2 13B + LoRA	**96.64**	92.05

Model	Acc
Kogkalidis et al. (2019)	90.68
Prange et al. (2021)	94.83
Kogkalidis and Moortgat (2023)	95.07
Yamaki et al. (2023)†	95.48
Llama2 13B + LoRA	**95.5**

Model	Acc	Parsable
Kogkalidis and Moortgat (2023)†	95.81	92.47
Llama2 13B + LoRA	**96.25**	**99.17**

Table 2: Fine-tuning results for CCGbank (top), CCGrebank (middle), and LCGbank (bottom). All numbers are in percentage. † are our replication.

level accuracy archives state-of-the-art. We have also presented results for parsing (LF) using our supertags with the C&C parser(Clark, 2015). Yamaki et al. (2023) focuses more on parsing rather than supertagging and includes learning derivations during training, achieves higher LF scores than our work. However, our parsing performance has exceeded all other prior works, confirming the robustness of the supertags generated by our method.

For LCG, its derivations are slightly different from those of CCG, and there is no corresponding annotation in the LCGbank. Therefore, we compare our method to the second-best supertagger (Kogkalidis and Moortgat, 2023) on the CCGBank and rebank. We employ the parser from Zhao and Penn (2024) for parsing performance. The results show a significant improvement in word-level supertag accuracy. From a parsing perspective, 92.47% of the sentences tagged by Kogkalidis and Moortgat (2023) are parsable, whereas 99.17% of the output from our tagger is parsable. These results further confirm our model's ability to learn syntax.

Furthermore, we found that LoRA's outcomes were superior to those of full fine-tuning. The accuracy for Llama2-7B full fine-tuning on CCGbank is 96.1%, which is worse than the performance of Llama2-7B + LoRA.

4.3.2 Impact of Repeating Times

In this experiment, we selected the setting of rank 8 + generation-based output for CCGbank supertagging. We chose to showcase this combination because, in this setup, the model initially used only one forward sequence, both Llama2-7B and Llama2-13B's performance was even lower than the classical BiLSTM with ELMo word embedding (the baseline in Bhargava and Penn (2020)) model. However, simply by increasing the number of repetitions of the forward sequence, Llama2-13B's performance surpassed that of the BERT-based model (Tian et al., 2020). This improvement underscores the importance of repeating input in LLM text analysis. It suggests that the repetition of input sequences can help the model better understand and analyze the text, leading to more accurate and reliable results. See Figure 2 for the detailed performance.

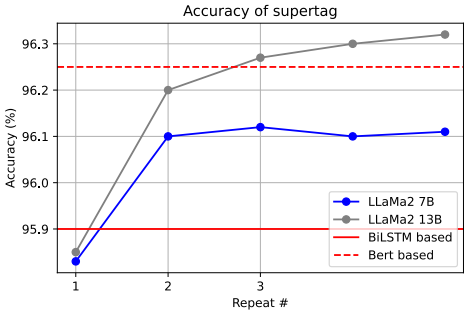

Figure 2: Demonstration of the impact of the number of repeating.

For all other factors that influence the model's performance, we put analysis in Appendix F.

5 Conclusion

We examine LLM's ability to perform supertagging tasks. We find that providing repeating input to LLMs can boost their performance. Our results show that generative models may also be applied to capture the syntax of natural languages. Future works may re-evaluate the capability of decoder-based auto-regressive models on other classification problems.

Acknowledge

We would like to express our gratitude to Aditya Bhargava for his invaluable contributions to the replication of the models in prior research.

Limitation

Due to the limitation of GPU resources, our research did not explore the potential of larger models. This decision inevitably limits the breadth of our experimentation and potentially the ceiling of accuracy we might have achieved. Employing models with a greater number of trainable parameters and adopting advanced tuning strategies could very likely enhance model performance. However, such approaches demand significantly more computational resources. The trade-off between the potential gains in accuracy and the substantial increase in required resources presents a critical consideration. This limitation underscores a broader challenge within the field: balancing the pursuit of state-of-the-art performance with the practicalities of computational resource availability and environmental impact.

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Aditya Bhargava, Timothy A. D. Fowler, and Gerald Penn. 2024. Lcgbank: A corpus of syntactic analyses based on proof nets. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, Turin, Italy.

Aditya Bhargava and Gerald Penn. 2020. Supertagging with CCG primitives. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 194–204, Online. Association for Computational Linguistics.

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

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

Stephen Clark. 2002. Supertagging for Combinatory Categorial Grammar. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pages 19–24, Universitá di Venezia. Association for Computational Linguistics.

Stephen Clark. 2015. The java version of the c&c parser: Version 0.95.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2023. News summarization and evaluation in the era of gpt-3.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Matthew Honnibal, James R. Curran, and Johan Bos. 2010. Rebanking CCGbank for improved NP interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215, Uppsala, Sweden. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Konstantinos Kogkalidis and Michael Moortgat. 2023. Geometry-aware supertagging with heterogeneous dynamic convolutions. In *Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD)*, pages 107–119, Gothenburg, Sweden. Association for Computational Linguistics.

Konstantinos Kogkalidis, Michael Moortgat, and Tejaswini Deoskar. 2019. Constructive type-logical supertagging with self-attention networks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 113–123, Florence, Italy. Association for Computational Linguistics.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, San Diego, California. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yufang Liu, Tao Ji, Yuanbin Wu, and Man Lan. 2021. Generating ccg categories. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13443–13451.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Nicholas Pangakis, Samuel Wolken, and Neil Fasching. 2023. Automated annotation with generative ai requires validation.

Jakob Prange, Nathan Schneider, and Vivek Srikumar. 2021. Supertagging the Long Tail with Tree-Structured Decoding of Complex Categories. *Transactions of the Association for Computational Linguistics*, 9:243–260.

Tomoki Sugimoto and Hitomi Yanaka. 2022. Compositional semantics and inference system for temporal order based on Japanese CCG. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 104–114, Dublin, Ireland. Association for Computational Linguistics.

Xiaofei Sun, Xiaoya Li, Shengyu Zhang, Shuhe Wang, Fei Wu, Jiwei Li, Tianwei Zhang, and Guoyin Wang. 2023. Sentiment analysis through llm negotiations.

Yuanhe Tian and Yan Song. 2022. Combinatory grammar tells underlying relevance among entities. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5780–5786, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuanhe Tian, Yan Song, and Fei Xia. 2020. Supertagging Combinatory Categorial Grammar with attentive graph convolutional networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6037–6044, Online. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3534–3547, Singapore. Association for Computational Linguistics.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023. Gpt-ner: Named entity recognition via large language models.

Tingyu Xie, Qi Li, Jian Zhang, Yan Zhang, Zuozhu Liu, and Hongwei Wang. 2023. Empirical study of zero-shot NER with ChatGPT. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7935–7956, Singapore. Association for Computational Linguistics.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. CCG supertagging with a recurrent neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 250–255, Beijing, China. Association for Computational Linguistics.

Ryosuke Yamaki, Tadahiro Taniguchi, and Daichi Mochihashi. 2023. Holographic CCG parsing. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 262–276, Toronto, Canada. Association for Computational Linguistics.

Hao Yu, Zachary Yang, Kellin Pelrine, Jean Francois Godbout, and Reihaneh Rabbany. 2023. Open, closed, or small language models for text classification?

Jinman Zhao and Gerald Penn. 2024. A generative model for lambek categorial sequents. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, Turin, Italy.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

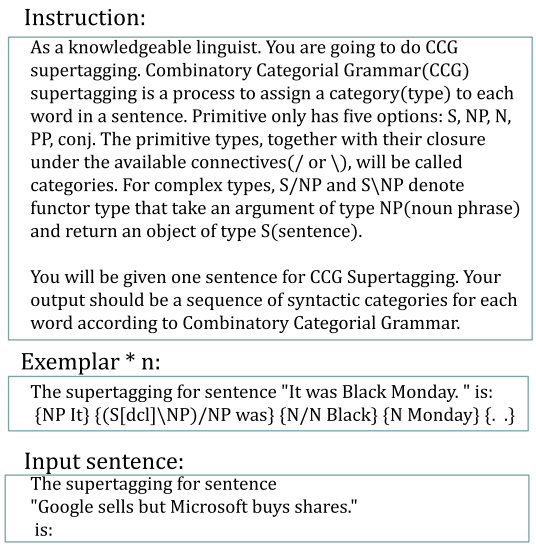

Figure 3: This is a demonstration of an in-context learning prompt. The prompt is divided into three key components: instruction, exemplar and input sentence.

A In-Context Learning Example

See Figure 3 for ICL CCG.

B CG Bank Statistics

See Table 3a for CCGbank, Table 3b for rebank and Table 3c for LCGbank.

C Hyperparameter

We tune Llama2 on a platform provided by LLaMa-Factory (Zheng et al., 2024), which uses PyTorch and the Hugging Face library. For inference, the temperature was set to 0 to enhance the stability of the output. The learning rate was set at 3e-5, with the max new tokens being 1024, and the learning rate scheduler type is cosine. LoRA targets are q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj and down_proj. Use the valid set for tuning purposes.

D Output Cleaning

The sequences generated by LLM during ICL may not always be in the desired format due to the following several reasons:

- Tokenizer inconsistencies: For instance, some tokens in the CCGbank that consist of two consecutive words might be interpreted as a single word by the LLM's tokenizer.

	Train	Valid	Test
Token #	929552	45422	55371
frequent(≥100)	919946	44973	54825
uncommon(≥10)	7549	354	442
rare(≥1)	2057	66	82
unseen(=0)	0	29	22
Sentence #	39604	1913	2407
Category #	1286	394	435

(a) Statistics of CCGbank.

	Train	Valid	Test
Token #	943204	46201	56395
frequent(≥100)	931037	45614	55698
uncommon(≥10)	9640	459	563
rare(≥1)	2527	97	107
unseen(=0)	0	31	27
Sentence #	39604	1913	2407
Category #	1574	478	538

(b) Statistics of CCG rebank.

	Train	Valid	Test
Token #	1054937	45656	55641
frequent(≥100)	1046549	45339	55252
uncommon(≥10)	6607	246	296
rare(≥1)	1781	64	82
unseen(=0)	0	7	11
Sentence #	44871	1921	2416
Category #	1086	296	327

(c) Statistics of LCGbank.

Table 3: Statistics of various categorial grammar banks, split by train/valid/test set. Note that for frequent/uncommon/rare/unseen tokens, it is determined by the number of occurrences(in train) of the token's category not the number of occurrences of the token itself.

- Addition or omission of words: While these modifications might not change the semantics, they change the structure of the sentence.
- Word modifications: Alterations such as tense changes, typo correction, switching from plural to singular form, or changing from uppercase to lowercase.

We use the longest common subsequence (LCS) algorithms solved by dynamic programming. The process begins by identifying the LCS between the prediction and the ground truth. Subsequently, we only consider the categories for the words within this sequence that are correctly predicted when calculating accuracy. This accuracy is then divided by the total token count. Example outputs are listed in Table 4. To address these issues, we use the longest common subsequence (LCS) algorithms solved by dynamic programming. The process begins by identifying the LCS between the prediction and the ground truth. Subsequently, we only consider the categories for the words within this sequence that are correctly predicted when calculating accuracy. This accuracy is then divided by the total token count, for CCG, as illustrated in Table 3a, which in this context is 55,371.

This metric impacts accuracy to some extent, especially in the ICL, where we've found that the tokenizer is different than CCGbank.

Ground truth	Generated output
{S/S No}	{NP[nb]/N No}
{, ,}	{, ,}
{NP it}	{NP it}
{(S[dcl]\NP)/NP was}	~~{NP/N is}~~
{(S\NP)\(S\NP) n't}	{NP n't}
{N/N Black}	{N Black}
{N Monday}	~~{N monday}~~
{..}	{..}

Table 4: Demo for computing accuracy after LCS. The total token is 8, correct predictions are {, ,}, {NP it},{. .}. So accuracy is 3/8.

E Llama Output

Table 5 demonstrates the output of Llama2-7B with input sentence *'Big investment banks refused to step up to the plate to support the beleaguered floor traders by buying big blocks of stock, traders say.'*.

Table 6 demonstrates the output of Llama2-7B with input sentence *'The 49 stock specialist firms on the Big Board floor – the buyers and sellers of last resort who were criticized after the 1987 crash – once again could n't handle the selling pressure .'*.

Table 7 demonstrates the output of Llama2-13B with input sentence *'Big investment banks refused to step up to the plate to support the beleaguered floor traders by buying big blocks of stock , traders say .'*.

F Impact of other factors

F.0.1 Impact of Linearization

Table 8 shows that generation-based has higher accuracy. This experiment is under the setting that tuning Llama2-7B with rank=2 and without repeating the input.

From Table 8, we can see that there isn't a significant difference between the two methods of linearization; however, generation-based is slightly better.

F.0.2 Impact of Rank

Figure 4 the impact of rank during tuning with feeding input twice. We can see that rank and accuracy are positively correlated, with accuracy converging at a rank of 256.

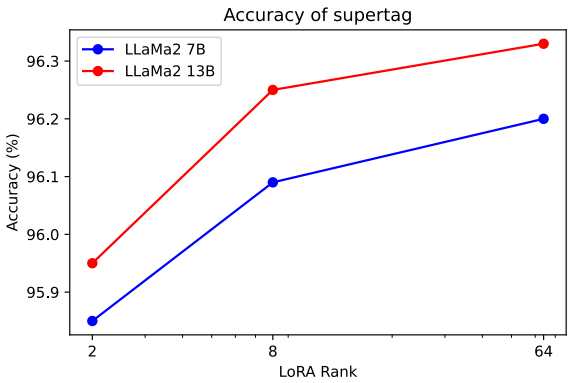

Figure 4

F.0.3 Impact of Beam

Figure 5 demonstrates how the beam influences the accuracy. While the beam increases, accuracy grows.

Generated output
1. Big - N
2. investment - N
3. banks - N
4. refused - N/N
5. step - N
6. up - N
7. to - N
8. plate - N
9. support - N
10. beleaguer - N
...

Table 5: Llama2 output1.

Generated output
1. The - S
2. stock - N
3. specialist - N
4. firms - N
5. on - P
6. the - S
7. Big - N
8. Board - N
...

Table 6: Llama2 output2.

Generated output
1. "Big" - N
2. "investment" - N
3. "banks" - NP
4. "refused" - conj
5. "to" - prep
6. "step" - V
7. "up" - adv
8. "to" - prep
...

Table 7: Llama2 output3. With non-CCG primitives.

Linearization	Acc(%)
Classification based	95.79
Generation based	95.83

Table 8: Result for different linearization methods.

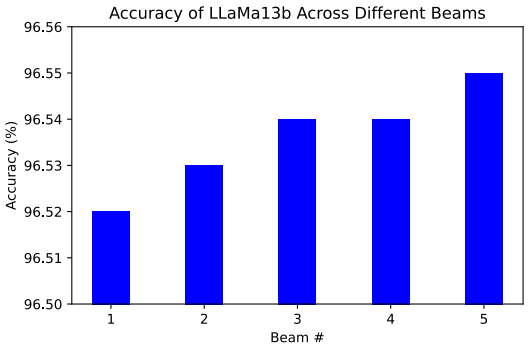

Figure 5: Impact of beam.