Llmbda: Large Language Model Based logic Deduction Assistant

Liam Scalzulli
McGill University
liam.scalzulli@mail.mcgill.ca

Jeff Zhang
McGill University
jeff.zhang2@mail.mcgill.ca

Sam Zhang
McGill University
sam.zhang@mail.mcgill.ca

Abstract

In this study, we evaluated the performance of large language models (LLMs) like Llama 2, GPT-3.5 Turbo, and GPT-4 in semantic parsing tasks, transforming semi-formal natural language into formal propositional logic based on McGill University's course requisites. Our analysis showed that while non-fine-tuned LLMs struggled with following rules, fine-tuned models performed well, suggesting their capability for understanding logical semantics.

1 Introduction

Since late 2022, large language models (LLMs) like ChatGPT have gained popularity due to their ability to perform diverse natural language processing tasks with human-like proficiency. These models, typically structured as decoder-only transformers, efficiently process text into token embeddings that capture both syntactic and semantic nuances. This architecture enables effective language generation and has supported the recent growth in model sizes (Zi et al., 2023). LLMs are particularly user-friendly due to their natural language instruct-ability, leading to widespread usage in research and industry for various tasks.

Semantic parsing, which involves converting natural language with logical elements into structured expressions like first-order logic (FOL), may benefit from the use of LLMs, as these models efficiently handle long-range dependencies and provide expressive representations, crucial for tasks like Q&A systems, information extraction, and machine translation.

This paper explores whether LLMs, through extensive training corpora, can understand semantics and convert semi-formal natural language (closely resembling logic but not fully structured) into structured propositional logic. We examine the model's accuracy in this conversion and investigate the effect of different alignment methods on model performance.

`COMP 202 and (COMP 250 or COMP 206) or permission of instructor` = COMP 202 $\wedge$ (COMP 250 $\vee$ COMP 206)

Figure 1: A semi-formal natural language and its formal propositional logic representation

2 Related Work

Logic-related semantic parsing is not a new task in NLP, and many forms of textual meaning representation have been explored. Such as first-order logic, lambda calculus, graphs, and programming languages (Kamath and Das, 2019).

Earlier systems for semantic parsing that showed some success were rule-based, applying a set of rules to a sentence's parse tree to generate the meaning representation. An example of

this can be seen in (Woods et al., 1972), where natural language is converted into a database language to execute queries. However, rule-based systems require hand engineering and extensive linguistic knowledge to develop the rule set and thus are difficult to generalize.

Statistical methods later became preferred as semantic information could be learned via supervised learning. (Zettlemoyer and Collins, 2012) implemented a learning algorithm similar to conditional random fields, training on sentences labeled with their lambda calculus representation. This model learns a probabilistic grammar, mapping lexical items to lambda calculus expressions. These can be composed to obtain the sentence's most likely semantic parse.

With the recent surge in LLM research, a new focus in semantic parsing has emerged in code generation tasks (Luo et al., 2023), in which LLMs have shown positive performance. In our research, we borrow a similar idea and explore the use of LLMs for logical semantic parsing, employing a unique dataset and focusing on the less-explored task of parsing propositional logic.

3 Method

3.1 Data Selection & Augmentation

Our dataset is created from McGill University's eCalendar. We selected $\approx$ 600 course requisites with diverse structures, which were manually labeled. To increase the sample size, we performed data augmentation by substituting course codes in the input strings and labels, creating new sentences with identical structures and trees, but different course codes.

3.2 List Representation of Logic

We present a structure for representing propositional logic, utilizing Python's list format for ease of parsing and correctness verification.

```
['&', 'COMP 202', ['|', 'COMP 250', 'COMP 206']] ✓
['|', [. . .], 'permission of instructor'] ✗
```

Figure 2: Non course code related requisites should be ignored, requisite from Figure 1

This structure is a recursive syntax tree: leaves represent individual course codes as strings, and non-leaves are lists. The first element of each non-leaf list is a logical operator, either "&" for "and" or "|" for "or," followed by its child nodes, and every non-leaf must have at least two children. The structure excludes any non-course code-related requirements. (Figure 2)

4 Experiment

We evaluated the performance of several well-known LLMs on our task, including OpenAI's GPT-3.5 Turbo and GPT-4 (the engines behind ChatGPT), Google's Bard (powered by Gemini Pro), and Meta's open-sourced Llama 2 and Llama2-chat. Due to hardware limitations, only the 7B parameter version of Llama 2 was tested.

4.1 Prompt Engineering

For prompt-instruct enabled models[1], we test out its task-specific performance using prompt engineering techniques. Specifically, for each sample in testing, we start a brand-new session with the models and inject the system prompt before asking it to formalize a single input. (Figure 3) For all models, we evaluate the performance with zero-shot learning (no examples) and few-shot learning (including the same set of 5 training examples in the prompt).

4.2 Supervised Fine-Tuning

For models with options for fine-tuning[2], we perform supervised fine-tuning for alignment

[1] Llama-2-chat, GPT 3.5 Turbo, GPT 4, Gemini Pro
[2] GPT3.5, Llama2

Figure 3: Example of a task following a system prompt.

on a relatively small and high-quality dataset (Zhou et al., 2023). For GPT models, we fine-tune with 375 samples as per OpenAI's official recommendation. For Llama2, we fine-tune on an augmented dataset of $\approx$ 3300 samples locally, using Parameter-Efficient Fine-Tuning (Mangrulkar et al., 2022) and 4-bit quantization of the parameters.

4.3 Evaluation

Evaluation methods should be informative and provide insights into the model's capability to formalize input into logical statements. These requirements lead to challenges when designing a benchmarking system, which we try to address with our evaluation metrics.

1. Two logical statements with different trees can be equivalent. Only comparing trees is not a good way of measuring accuracy.

2. Only comparing the equivalency of the model output and the label does not give an indication of how close the model's answer was to the correct label. A model that outputs only incorrect trees can have similar accuracy to a model that outputs almost correct trees.

4.3.1 Well-formedness

The well-formedness of the tree is an indicator function of whether the tree is syntactically valid. The tree is well-formed if it can be parsed as a nested list in Python, and respects all rules defined in 3.2. Trees that do not follow the rules are considered invalid and cannot evaluated by further metrics.

4.3.2 Tree similarity

We treat the human-labeled syntax tree for each sample as the golden standard. For each prediction, we compute the tree similarity based on the tree edit distance between two trees. The tree edit distance between two trees T_1 and T_2 is defined as the minimum number of edit operations required to make $T_1 = T_2$, where valid edit operations are: inserting a node, deleting a node, and relabeling a node (Henderson; Zhang and Shasha, 1989; Bille, 2005). This benchmarks how well a model was able to partially formalize the logic behind a natural language given an incorrect model output.

4.3.3 Accuracy based on logical equivalency

We compute the model accuracy based on the logical equivalency between the human-labeled tree and the model-labeled tree. For every sample, we compare the truth table of both trees, with course codes being boolean variables. Two trees with the same truth table are considered a correct prediction.

5 Results

In addition to the overall performance, we also bin the test samples by their tree's complexity for evaluation. A tree's complexity C_T scales with its number of leaves l_T and its height h_T, it is defined as:

$$C_T = l_T \times (h_T - 1) \tag{1}$$

The threshold for each bin is:

Alignment	Model	Accuracy				Mean Tree Distance				% Well-Formed			
		Overall	Trivial	Medium	Hard	Overall	Trivial	Medium	Hard	Overall	Trivial	Medium	Hard
zero-shot	GPT-3.5-turbo	15.13%	9.20%	37.50%	1.25%	0.48	0.04	0.42	5.44	22.27%	10.34%	50.00%	6.88 %
	GPT-4	52.52%	49.43%	68.75%	68.76%	0.27	0.05	0.23	2.88	58.82%	52.30%	77.08%	75.00%
	Gemini Pro	15.55%	11.49%	33.33%	6.25%	0.44	0.02	0.5	4.75	23.95%	12.64%	58.33%	43.75%
few-shot	Llama2-chat-7B	6.30%	5.74%	10.42%	0.00%	0.16	0.02	0.27	1.31	11.34%	6.90%	27.08%	12.50%
	GPT-3.5-turbo	78.99%	87.36%	66.67%	25.00%	0.29	0.03	0.52	2.38	88.24%	89.66%	89.58%	68.75%
	GPT-4	90.76%	93.68%	87.50%	**68.75%**	0.28	0.05	0.75	1.31	95.80%	95.98%	97.91%	87.50%
	Gemini Pro	76.47%	85.63%	58.33%	31.25%	0.5	0.10	0.54	4.69	86.55%	88.51%	77.08%	93.75%
fine-tuned	Llama2-7B	61.34%	71.84%	38.58%	12.50 %	0.59	0.12	0.77	5.19	77.74%	75.86%	83.33%	81.25%
	GPT-3.5-turbo	**93.70%**	**97.70%**	**89.58%**	62.50 %	0.34	0.04	0.13	4.31	99.16%	99.16%	99.43%	97.92%

Table 1: Overall results for each model.

$$\begin{cases} \text{Trivial} & 0 \leq C_T \leq 4 \\ \text{Medium} & 4 \leq C_T \leq 15 \\ \text{Hard} & \text{otherwise} \end{cases} \quad (2)$$

In our experiments, we used Llama 2 Chat for prompt-based few-shot instruction tests, as the base Llama 2 model lacks chat instruction capabilities. Additionally, we did not record zero-shot results for Llama2-chat, as it could not be verbally instructed to produce the required output format, likely due to limitations inherent in its smaller model size.

From Table 1, we see that zero-shot prompted models generally perform poorly. The few-shot prompted models performed better as the model's number of parameters increased, leading to an extremely large range of results from ac curacies of 6.30% on Llama2-7B to 90.76% on GPT-4. Prompt-engineered Gemini Pro performed similarly to GPT-3.5 Turbo. Overall, the best performer is the fine-tuned GPT-3.5, closely followed by few-shot prompted GPT-4.

6 Discussion

The task was challenging for most non-fine-tuned LLMs, especially in enforcing output rules through prompt engineering alone. Chat-instruct models struggled to follow instructions, such as only outputting trees or ignoring irrelevant requirements, despite being indicated in the prompt. Interestingly, zero-shot prompted models performed better on medium examples than trivial ones, as they struggled to output well-formed trees for trivial examples. This was remedied with few-shot prompting. Fine-tuned models showed more promise in outputting well-formed trees, benefiting from weight adjustments by training on many fine-tuning examples, rather than mere context changes. Smaller fine-tuned models (e.g. GPT-3.5) outperformed their chat-instruct counterparts, even those with larger sizes (e.g. GPT-4). This suggests limitations in the model's ability to align with complex tasks within a small context window.

Alignment techniques were crucial for both methods. For chat-instruct models, few-shot learning with a mix of instructions and examples proved effective for very large models, aligning with existing prompt engineering consensus (Garcia et al., 2023; Liu et al., 2023). In contrast, fine-tuned models did not require examples in the prompt, but the larger and more diverse set of training data significantly improved their performance.

Despite said challenges, some models successfully parsed many test samples. In particular, with fine-tuning, both GPT-3.5 and Llama 2 showed large improvements from their few-shot settings. Fine-tuned GPT-3.5 even surpassed few-shot prompted GPT-4. Few-shot prompted GPT models also showed good accuracy. These

results support our hypothesis about the capability of LLMs to understand formal logic.

7 Limitations

Our methodology has limitations, notably in the tree similarity measure used for evaluating model outputs. This measure does not perfectly represent correctness, as it does not consider alternative valid logical representations. Additionally, malformed trees are not considered in the calculation, leading to larger tree edit distances for models that generate well-formed but incorrect trees. Consequently, the tree similarity metric should still be viewed as a secondary indicator and with caution.

The scope of our semantic parsing experiment is narrow, focusing mainly on short inputs with only two logical operators. This limited scope does not fully capture the complexity of natural language, and may not comprehensively represent the LLMs' capabilities in more complex semantic parsing tasks.

Lastly, hardware limitations confined our tests to the 7B parameter Llama, leaving a large gap in the parameter range up to GPT-3.5. This unexplored range prevents a conclusive understanding of how model size correlates with performance.

8 Conclusion

In this study, we evaluated the ability of various large language models, including open-source ones like Llama 2 and closed-source models such as GPT and Gemini, on the task of parsing semi-formal natural language into propositional logic. Our analysis compared different alignment methods: zero-shot and few-shot prompting, and supervised fine-tuning. The high performance of fine-tuned models suggests LLMs can understand logical semantics with appropriate training data. However, chat-instruct and general-purpose LLMs suffer from inconsistent performance on this task.

From the results, a model's parameter count is immensely intertwined with its understanding of semantics. However, experimentation with a larger range of numbers of parameters is needed to understand this relationship.

Potential future research direction in this domain could include experiments with other tasks, such as parsing fully natural language, generating FOL, and with a wider range of models such as different LLM sizes and non-LLM language models. Finally, given the potential ability of LLMs to understand semantics, further engineering research on the downstream application of LLMs in this domain may also lead to advancements in NLP at an industrial level.

References

Philip Bille. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1):217–239.

Xavier Garcia, Yamini Bansal, Colin Cherry, George Foster, Maxim Krikun, Fangxiaoyu Feng, Melvin Johnson, and Orhan Firat. 2023. The unreasonable effectiveness of few-shot learning for machine translation.

Tim Henderson. Zhang-shasha: Tree edit distance in python. https://github.com/timtadh/zhang-shasha.

Aishwarya Kamath and Rajarshi Das. 2019. A survey on semantic parsing.

Xin Liu, Daniel McDuff, Geza Kovacs, Isaac Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, and Shwetak Patel. 2023. Large language models are few-shot health learners.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct.

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

W. Woods, Ronald Kaplan, and Bonnie Webber. 1972. The lunar sciences natural language information system.

Luke S. Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *CoRR*, abs/1207.1420.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18:1245–1262.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment.

Wenjie Zi, Layla El Asri, and Simon Prince. 2023. A high-level overview of large language models. https://www.borealisai.com/research-blogs/a-high-level-overview-of-large-language-models.