Refining User Instructions for Better LLM Assistance

Anonymous ACL submission

Abstract

A typical application scenario for generative LLMs is directly interacting with end-users in conversation. However, the distribution of actual user instructions can differ from those in the publicly available datasets, which could negatively influence the user experience. In this paper, we propose a new method to overcome the instruction's difference via regenerating the instruction. We address a specific case of how user instruction can differ: more flaws can exist in their daily expressions. We leverage instruction-tuned LLMs to refine the flawed instruction so they better align with the training distribution. We explored the effectiveness of directly asking the model to refine the instruction and further finetuned a specialized refiner model to enhance the overall performance. Our experiments demonstrate the effectiveness of the proposed method on the open-source model, especially when using a finetuned model as the refiner. The enhancement is achieved without requiring retraining or parameter increasing on the assistant model, highlighting its practicality and potential to bridge the gap between open-source and proprietary LLM assistants.

1 Introduction

The world has witnessed widespread adoption of Large Language Models (LLMs) in real-world applications, where LLMs directly assist with end-users requests in a conversation style. To complete the user's instruction as expected, not only is the model required to be capable of finishing specific tasks, but correctly interpolating the user's instruction is also crucial. However, the user instructions the model met in deployment can differ from the instructions they are trained on in the datasets, leading to a distribution shift. This out-of-distribution (OOD) problem can hurt the performance of machine learning systems, including the performance of LLM(Wang et al., 2023).

For those public applications powered by proprietary LLMs, like ChatGPT, Claude, and Bard, this distribution shift of instructions is more accessible to resolve as the interaction data is typically gathered and can be used to retrain the model to adapt to the actual distribution of the user instructions(OpenAI, 2023b). However, the data collection policy raises many privacy concerns and may not be acceptable for private applications(Verge, 2023).On the other hand, open-sourced models are less likely to be able to collect actual interaction data and be actively retrained, both due to the expense and the difficulty in the data collection, which impacts the performance and user experience, which can hold people back from using open-source LLMs to substitute the reliance on proprietary ones.

In this work, we focus on a situation with more flaws, like grammar mistakes and informal language used in the institution, which is less common in most public datasets, including those used for instruction tuning and human preference alignment. We propose an alternative method to address the flawed user instruction by regenerating the user instruction to resemble the instruction in the train data. We achieved that by leveraging an instruction-tuned LLM to refine the faulty instructions before providing them to a fixed assistant LLM. The approach utilizes the abilities of LLMs to follow given instructions and generate instructions in a style similar to the examples used to train them. Building a pipeline application to enhance the performance of LLM-based applications is common in practice (Schlag et al., 2023). We introduced a specialized refiner model with an instruction-tuned open-source LLM as the base model to enhance the effectiveness further. Overall, the additional refining step is practical to increase the performance of open-source LLMs in our experiments, and the fine-tuned refiner model shows improved performance in practice.

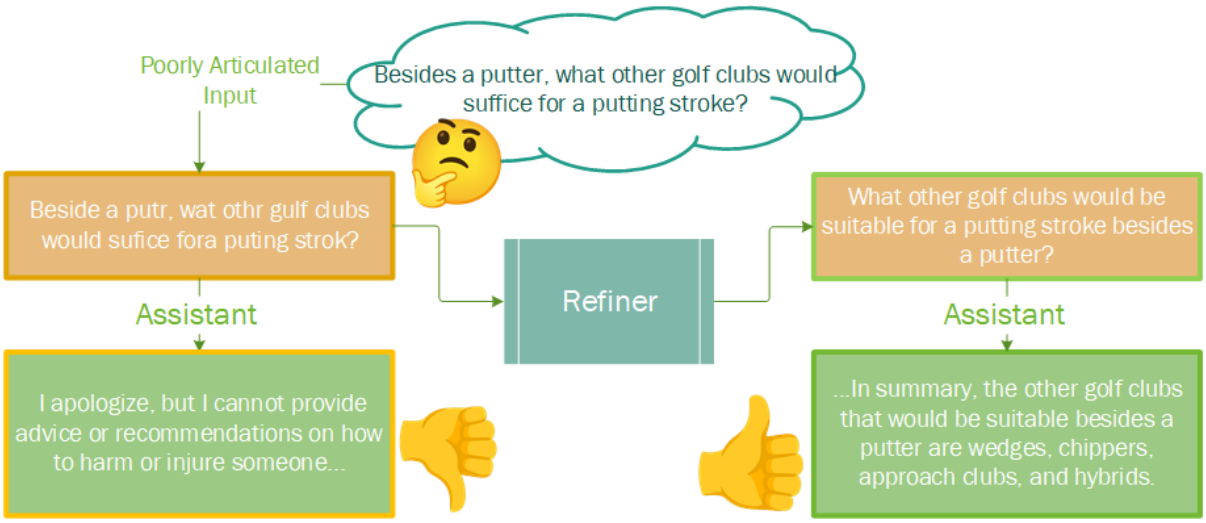

Figure 1: An example from our experiments demonstrates the effectiveness of refining flawed user instructions. Despite the model's ability to precisely restore the user's intention, if the raw user instruction is given directly to the assistant, the model refuses to answer the question, possibly mistaking the misspelled "suffice" for "suffer" or "suffocate."

We present the concept and implementation of our method in Section 2. In Section 3, we empirically show the effectiveness of the refiner approach. We presented our experiment setup and the results of the performance evaluation. Finally, we discuss the related research (Section 4), the conclusion (Section 5), and the limitations (Section 6) of our study in the end.

2 Method

We considered the situation where an instruction-tuned Large Language Model is used as an assistant to directly respond to the user's instruction to satisfy the user's requirements, which is a common scenario in many LLM-based applications. We denote the user's requirement as q. The requirement will be satisfied by an optimal response r^*, which we represent as $r^* = o(q)$.

The system expects users to submit an instruction to represent their intention, denoted as I, which is formatted in a prompt template and provided to an auto-regressive text generation model to generate the response. For simplicity, we treat the formatting step and the generation process as a single function, whose input is the instruction and the output is the response, denoted as $r \sim A(I)$. Here, we use the $\sim$ symbol to represent this as a probabilistic function whose output is a distribution.

The instruction tuning process (Wei et al., 2022) trains the model to follow the user's instruction to generate a response that satisfies the user, so for a sufficiently trained model, we can assume that $A(i) = A(I(q)) \approx O(q)$, which is saying the probability to generate a response that is similar to the optimal response r^* is high enough. However, notice the similarity is dependent on I, which means the distribution shift of the user instruction, for example, the presence of flaws in the instructions, could cause an impact on the generation quality.

Our method can be represented as below:

1. Given the user instruction i, we first ask a refiner model to regenerate the instruction to fix its potential flaws. We denote the regenerated instruction as $i' \sim R(i)$

2. Next, the new instruction is presented to the generation model system to acquire the response: $r \sim A(i')$

With our additional refinement, the new instruction i' can better represent the user's intention so the model can sufficiently understand the user's intention. In other words, the composition $R \cdot I'$ should be closer to I than I' alone.

The refining process is a particular case of the guided regeneration method for tackling the OOD problem. In practice, we can use different types of regeneration instruction to address various distribution shifts. In our implementation, we used an instruction-tuned model as the refiner. As the model has seen instructions generated by I in training, it is unsurprising that the model has acquired

the ability to create instructions that follow a similar distribution. The experiment results also illustrated the effectiveness of the method. We use a prompt to guide the model to refine the instruction. We provide the instruction set we used in our experiment in Section 6. Besides the generation, we further applied some post-processing to the output of the model, which helps to normalize the response for more straightforward experiments and more control over the content, similar to how people extract the response from the generated text of an instruction-tuned LLMs that utilize a template (Wei et al., 2022).

3 Experiments

We evaluated our refining method on a modified version of publicly available instruction tuning datasets. To simulate a distribution shift in the instruction, specifically to increase the presence of flaws, we utilize GPT-3.5 to convert the instruction in the dataset.

We first examine the difference caused by the flaw insertion. The models tested include GPT-3.5[1] and LLaMA 2 7B Chat[2], and the original instructions are from the dolly dataset. We test both models under three settings:

1. Original instructions from the dataset, which we reference as *oracle*;

2. Instructions with inserted flaws, we reference them as *flawed*. We generated them based on the original ones with the method stated above.

3. A refined version of the instruction, converted from the corrupted ones by the same model with instruction to refine the instruction, is referenced as refined.

The responses are gathered and then graded by GPT-3.5 for preference score. Here, we utilize GPT-3.5 to rate the response from the assistant model, following prior work (Zheng et al., 2023). The answer is paired with the original instructions under all three settings to simulate the helpfulness and alignment of the response toward the user's intention.

We show the result of the first experiment in Figure 2, which indicates that GPT-3.5 (annotated as *gpt*) is less affected by the flaws introduced in the conversion (-4.2% in preference score compared to llama's -10.2%), and further refining based on the corrupted version of instructions is unable to increase the grading. In the case of GPT-3.5, the accumulated misalignment is likely to cause a further decrease in the score, as the model is less affected by the quality of the instruction. However, the experiment result on the llama model shows that the refining step is practical on this model, which we further prove in the following experiment.

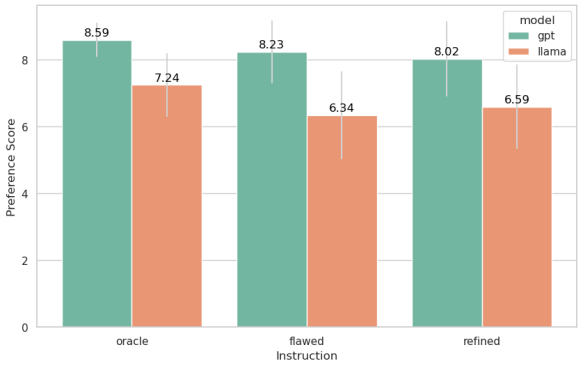

Figure 2: The quality of the instruction has a more significant impact on LLaMA than the proprietary model GPT-3.5. We provide the standard error as the error bar's length.

We also conduct further experiments to evaluate the effectiveness of the refining step with LLaMA 2 7B Chat as the assistant model. Here we collected instructions from two different datasets, noted as *dolly*[3] and *gpt4all*[4]. Both are filtered as previously described, then 1000 samples are drawn and used in our experiment. We first processed the instructions with GPT-3.5 to insert more flaws, which served as the baseline in this experiment. Then three models are used to refine the instruction, GPT-3.5 (referenced as *gpt* in Figure 3), LLaMA 2 7B Chat, (llama), then a finetuned refiner (*refiner*). The refiner model is finetuned based on LLaMA 2 7B Chat. The dataset used for training is a distinct set of the dataset *dolly*. We use 10,000 samples in the finetuning process. For each instance, we format the flawed version of the instruction and the instruction refined by GPT-3.5 in the refining template (we further format the result with the chat template used by the LLaMA 2 7B Chat model). The process is done using the TRANSFORMERS(Wolf et al., 2020), PEFT(Mangrulkar et al., 2022), and

[1]The model used is GPT-3.5-TURBO-0613. API is provided by OpenAI.

[2]We downloaded the model from Hugging Face.

[3]Downloaded from databricks/databricks-dolly-15k

[4]Downloaded from nomic-ai/gpt4all-j-prompt-generations

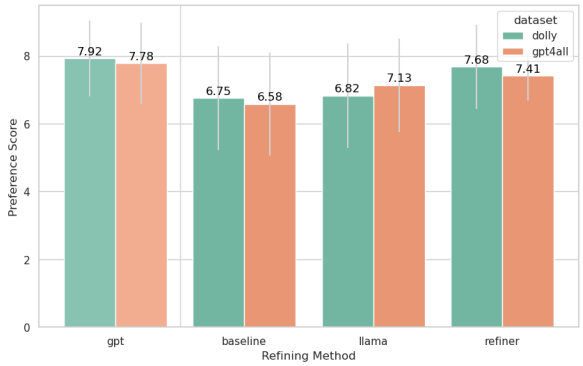

Figure 3: Above is a comparison of the effectiveness of different refining methods. We also show the result of GPT-3.5 as the refiner for comparison. The non-refined version instructions serve as the baseline in this experiment.

TRL(von Werra et al., 2020) library.

We pair the *oracle* instructions with the responses corresponding to all four versions of processed instructions. Then, we utilize GPT-3.5 with the method stated above in the first experiment to get the preference score. We show the result in Figure 3. As a result, with the assistant model fixed, GPT-3.5's ability to perform the refining is still more potent than the open-source models. However, after finetuning, the gap is effectively narrowed (the difference is reduced by 78.0% on the *dolly* dataset and 42.6% on the *gpt4all* dataset). Overall, the experiments demonstrate the potential of the refining step to increase performance on various flawed instructions, and the specialized refiner is significantly more capable of doing so without changing the parameter size and structure.

4 Related Work

Previous research widely explored the problem of out-of-distribution (OOD) in various NLP fields, including the detection(Lang et al., 2023), performance evaluation(Teney et al., 2023), and the method to approach better generalization(Yang et al., 2023). Multiple studies also described the distribution shift between LLM's training and inference. For example, (Ren et al., 2023) tried to detect when the instruction did not follow the training distribution. (Kirk et al., 2023) compared the effectiveness of SFT and RLHF method for adapting pretrained LLMs to new inputs. We considered the flawed user instructions and the instructions used for training as an OOD problem and used an additional refining step to resolve it.

Utilizing LLM's capability to enhance the LLM system's overall performance by introducing more steps is also common in practice. Some work attempts to directly optimize the responses, for example, (Welleck et al., 2022), (Liu et al., 2023), and (Lightman et al., 2023). Some optimize the prompt like us, (Cheng et al., 2023) proposed using prompt-optimization as an alternative to the RLHF training process by directly reflecting the difference in the responses into the prompt. (Weston and Sukhbaatar, 2023) remove unrelated information from the prompt to reduce the negative effect on the accuracy of the assistant's response. One concurrent work (Deng et al., 2023) performed refining to the instruction similar to ours and observed an increment in performance of GPT-4 on several custom benchmarks. Our work focused on improving the performance of smaller open-source model assistant systems and further showed that finetuning can increase the system's performance.

5 Conclusion

We present a refining-based instruction regenerating step to tackle the OOD challenge in the application scenarios where the end users can produce instructions with more flaws. Our method effectively increased the performance of the open-sourced LLM assistant model on flawed user instructions. The experiments conducted on the simulated dataset demonstrated the effectiveness of the refining step and how we can further finetune the model to perform such refining to enhance the effect.

6 Limitation

We cannot perform the preference grading with state-of-the-art LLMs such as GPT-4 (OpenAI, 2023a), which could affect the precision of the grading, according to our reference (Zheng et al., 2023). We generated our flawed user instructions with LLM, and the distribution of flaws in actual user input can differ. Nevertheless, we argue that the introduction of flaws, whether from user input or intentional injection, serves as an instance of OOD, so the effectiveness of our method still holds. The amount of instructions tested is also limited, and the language and length of the instructions are restricted, so the result may require more testing under a different setting. Further, the open-source model used in this work is limited to LLaMA 2 7B Chat, which simultaneously serves as the assistant model and the instruction refiner. With a different

model, especially a model different in parameter sizes, the ability to overcome the flaws in the user instruction and refine a user instruction can vary. By utilizing our method, there will be an extra cost in the inference time, which can lead to more expense and environmental impact, but the effect is limited. Considering the situation where the improved response satisfied the users, so no second request is required, we can potentially save more cost.

References

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2023. Black-box prompt optimization: Aligning large language models without model training.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2023. Understanding the effects of rlhf on llm generalisation and diversity.

Hao Lang, Yinhe Zheng, Yixuan Li, Jian Sun, Fei Huang, and Yongbin Li. 2023. A survey on out-of-distribution detection in nlp.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step.

Zeyi Liu, Arpit Bahety, and Shuran Song. 2023. Reflect: Summarizing robot experiences for failure explanation and correction.

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.

OpenAI. 2023a. Gpt-4 technical report.

OpenAI. 2023b. Openai privacy policy. https://openai.com/policies/privacy-policy.

Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J. Liu. 2023. Out-of-distribution detection and selective generation for conditional language models.

Imanol Schlag, Sainbayar Sukhbaatar, Asli Celikyilmaz, Wen tau Yih, Jason Weston, Jürgen Schmidhuber, and Xian Li. 2023. Large language model programs.

Damien Teney, Yong Lin, Seong Joon Oh, and Ehsan Abbasnejad. 2023. Id and ood performance are sometimes inversely correlated on real-world datasets.

The Verge. 2023. Openai's regulatory troubles are just beginning - the verge. https://www.theverge.com/2023/5/5/23709833/openai-chatgpt-gdpr-ai-regulation-europe-eu-italy.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, Binxin Jiao, Yue Zhang, and Xing Xie. 2023. On the robustness of chatgpt: An adversarial and out-of-distribution perspective.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct.

Jason Weston and Sainbayar Sukhbaatar. 2023. System 2 attention (is something you might need too).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Linyi Yang, Yaoxiao Song, Xuan Ren, Chenyang Lyu, Yidong Wang, Lingqiao Liu, Jindong Wang, Jennifer Foster, and Yue Zhang. 2023. Out-of-distribution generalization in text classification: Past, present, and future.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

A Prompts Used in Experiments

In this section, we provide the prompt used in our experiments to guide the behavior of LLMs. Our experiment is conducted on instruction-tuned LLMs, and the following prompt texts are used in the system prompt in the interaction with the model.

Convert Original Instruction to Flawed version

Below is an instruction describing a user request, paired with the related context. Coarsen the instruction to introduce grammar and spelling errors to make it almost incomprehensible. Your response should only contain the coarsened instructions.

Refine the Flawed instructions

Below is a problematic user request, paired with the further related context. Revise the instruction to improve its clarity and comprehensibility, but don't alter the context. Your response should only contain the refined instructions.

Follow User Instruction

You will be provided with a question and an optional context. Answer the question based on the context.

Grade the Response

Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Begin your evaluation by providing a short explanation. Be as objective as possible. After providing your explanation, please rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]"