Rewriting the input prompt by an LLM to improve output quality

Sina Hashemi
sinaHashemi@ut.ac.ir

Vahid Moradi
moradilak@ut.ac.ir

1 Problem statement

In recent years, large language models (LLMs) have demonstrated impressive capabilities across a wide range of tasks, from question answering to generating code and assisting in scientific research. However, the quality of their output is often highly dependent on the quality of the input prompts. Slight variations in wording, structure, or clarity can lead to vastly different responses. This creates a significant challenge: users may not always know how to formulate the most effective prompt to get accurate and relevant results.

Recent advances in prompting techniques (e.g. chain-of-thought prompting) have shown that how a question is presented to an LLM can significantly affect its ability to reason through the answer. Even so, users may not always know how to craft the best prompt for a given task. **This research proposes an automated solution: use one LLM to rewrite a user's input prompt into a potentially more effective prompt**, then feed this rewritten prompt to another LLM (or the same LLM) to generate the final answer. The intuition is that the rewriting LLM can serve as a knowledgeable "intermediary," interpreting the user's intent and adding clarity or structure that helps the answering LLM perform better. This approach leverages the generative and interpretive abilities of LLMs themselves to improve prompting, rather than relying solely on human engineering of prompts.

We will focus on tasks that are known to be challenging for LLMs without careful prompts, such as multi-step reasoning problems, mathematical word problems, and implicit reasoning questions. In these cases, a prompt rewriter might, for example, break a complex question into simpler parts, fill in assumed context, or otherwise reformulate the query in a way that makes the solution more reachable by the answering model. A key question we will address is how to instruct the rewriting LLM to produce an improved prompt without changing the original query's intent or introducing unwanted bias. We will also explore how much improvement can be gained with prompt rewriting when using various types of LLMs as the main respondent.

The problem of the effect of rewriting input prompts on the performance and accuracy of models is to be worked on. Our motivation is to reduce costs and save time while achieving better accuracy. The interesting thing is that the user prompts are going to be rewritten by the model and the user does not interfere in the process. In terms of improving the accuracy of the models after rewriting the input prompts, it can also be said that it is of scientific importance.

The practical importance of this work lies in its potential to improve the user experience with LLM, streamline workflows, and improve the quality of LLM-generated content in various domains. Standardizing terminology, removing ambiguities, respecting input-length constraints, and explicitly guiding the model's reasoning can lead to more predictable and desirable outcomes, making LLMs more accessible, effective for a wider range of users and applications, and increasing their accuracy.

2 What you proposed vs. what you accomplished

- ~~Select and preprocess dataset~~
- ~~Creating a dataset of rewritten prompts~~
- ~~Train (Llama-3.2) on selected dataset and examine its performance~~
- *Make fancy model perform better than baseline model*: We failed to do this because we

only started working on our project on May 11

- ~~Perform in-depth error analysis to figure out what kinds of examples our approach struggles with those.~~

3 Related work

Prompt engineering has emerged as a key technique for enhancing the performance of LLMs. Recent studies have explored various methods for optimizing the prompts to improve the model output without modifying the underlying model parameters, our work draws inspiration from several threads of research in prompt engineering and meta-optimization of prompts using LLMs. That Some of which are mentioned below.

Recently, Srivastava et al. (2024) proposed PRoMPTed, an approach where an LLM is used in the loop to rewrite each test instance's prompt in a zero-shot setting. In this approach, a new method called "PRoMPTed" has been introduced to improve the answers of language models, in which language models are in a loop in such a way that, using an auxiliary model (Meta LLM), it rewrites and modifies the question based on the initial output of the main model, so that the main model gives a better answer. That is, the second model looks at the answer of the first model and tries to change the question in such a way that the first model gives a better answer. The results of this method have shown that the rewritten prompts by the auxiliary model (GPT-3.5), which can be weaker than the main model (GPT-4), provide higher accuracy.

Another relevant study is by Li et al. (2024), who focused on personalized text generation and introduced a system to learn prompt rewriting policies. They trained a smaller rewriter model through a combination of supervised learning and reinforcement learning to automatically revise user prompts, aiming to incorporate personal context into the prompt. Their results on writing tasks showed that the rewritten prompts produced by the learned rewriter led to outputs that were preferred over those from the original prompts.

In the context of information retrieval augmented QA, Ma et al. (2023) explored query rewriting for LLMs. They introduced a "Rewrite-Retrieve-Read" pipeline where an LLM rewriter first reformulates the user's query to be more amenable for document retrieval, then the retrieved information is used to answer the query. Their experiments on open-domain QA tasks showed that a learned rewriter (including one implemented by a smaller LLM) can significantly boost the accuracy of a frozen reader LLM by crafting better search queries.

Our approach is also related to efforts in multi-turn conversational systems. Sarkar et al. (2025) present one of the first studies of using LLMs to mediate human-LLM chats by rewriting user queries in situ; in which, an LLM chatbot is used to get a prompt from the user and an LLM rewriter is used to rewrite the prompt. The general idea is that another language model (the rewriter) reads the user's question and rewrites it if necessary. In this way, the chatbot better understands what the user wants. The beauty of this approach is that this rewriter model looks at the conversation history, in addition to the question itself, to understand exactly what the user is looking for.

In another article (Kong et al., 2024), the PRewrite method was introduced, which has a simple idea and first tells LLM itself to rewrite the user's prompt, then gives the user the answer. the point of this method is that it requires the use of a very very large language model.

In one another article (Cheng et al., 2024), the authors decided to train the model using Element, using a robust model (GPT-4) to rewrite the prompts and produce a good dataset.

Overall, these works suggest a growing interest in LLM-based prompt optimization. Unlike traditional prompt engineering which relies on human intuition or brute-force search over prompt variations, using LLMs themselves to improve prompts offers a dynamic and automated strategy. Our project builds upon this foundation by developing a system that automatically rewrites user input prompts to produce more optimized versions. By standardizing terminology, removing ambiguities, and aligning prompts with model input constraints, our approach aims to improve the responsiveness and reliability of LLMs across various tasks.

In other words, our contribution will extend this literature by systematically evaluating prompt rewriting on explicit reasoning benchmarks and by comparing different model pairings for the rewriter-answerer roles. We also intend to contribute insights on the kinds of prompt transformations that are most helpful (e.g., adding step-by-step structure, clarifying ambiguous references, etc.), which could inform future prompt engineering guidelines. In summary, we build on the idea that prompts can be treated as another output to be optimized, with LLMs as the optimizers.

4 Your dataset

In this project, we plan to use the **GSM8K**(Cobbe et al., 2021) dataset, which is an open source dataset, and in the next phases, a dataset will be created from it that has an additional column called the rewritten prompt. The reason for using this dataset was that since the math questions have been answered and each question has a unique answer, it is easier to reward the model and complete its training.

So, we will primarily evaluate our approach on benchmarks that require reasoning, as these are expected to benefit most from better prompts. Our starting point is the GSM8K dataset, a collection of 8.79K grade-school math word problems that demand multi-step arithmetic reasoning. GSM8K is a standard test for LLM reasoning ability, and its problems come with detailed solutions, enabling automatic checking of answer correctness. We anticipate that many GSM8K questions (which are written in plain language) could be rephrased to highlight relevant details or logical structure, helping the main LLM avoid mistakes.

For consistency, prompts in the dataset will be posed to the models in a zero-shot manner (no task-specific examples given in the context), since our focus is on zero-shot prompt effectiveness. In the rewritten prompt condition, the original query will first be transformed by the rewriter LLM (with any task-specific instructions, like “let’s think step by step” if we choose to include them), and the resulting prompt will be what the main LLM sees. The baseline condition will use the original query directly with the main LLM. By comparing accuracy and other metrics between these conditions, we will quantify the improvement attributable to prompt rewriting.

4.1 Data preprocessing

In the preprocessing stage, what we will do is select a part of the dataset where models usually give incorrect answers to questions, because the accuracy of the models is good when answering all questions, and our goal is to help increase the accuracy of the model in answering questions that are usually answered incorrectly by rewriting the questions.

5 Baselines

The baseline considered is the accuracy of the models answering the questions without making any changes to the questions. The goal is to improve this accuracy by rewriting the questions. 70% of the dataset is considered as the train dataset, 10% is validation, and 20% is test dataset for testing the model. It should be noted that the accuracy considered for the baseline is the accuracy of the models answering the questions in this 20% test dataset, and the model we train does not see these 20% questions at all during training.

6 Your approach

In our approach, we want to use an auxiliary model to rewrite user prompts for the question answering task, in such a way that the initial prompt is given to the main model. If the model does not answer the question correctly, the prompt is rewritten by the auxiliary model and given again to the main model, this process continues until the model answers the prompt correctly. At each stage, the original prompt and the rewritten prompt, along with the answer given by the original model, are sent to the reward function, where if the question is answered correctly, 1 point is awarded, if the length of the rewritten prompt is shorter than the original prompt, 0.2 points are awarded, and depending on the semantic similarity of the rewritten prompt and the original prompt, up to 0.3 points are awarded to the auxiliary model during the training process to learn how to rewrite the prompt in the desired way. By repeating the process for all the trained prompts, an auxiliary model is trained using the RLHF method that takes the user prompt as input and produces a rewritten output in a way that the main model can give a better answer to it.

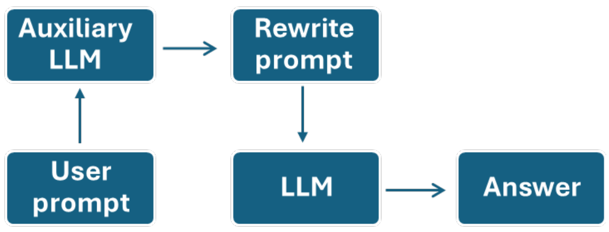

Figure 1: Roadmap

Our approach consists of a pipeline with two LLM modules: (1) a Main Answering LLM that takes the original user prompt and also receives the revised prompt and generates the final answer, and (2) a Prompt Rewriter LLM (auxiliary model) that takes the original user prompt from Main Answering LLM and generates a rewritten prompt. We will design a prompt template or instruction for the rewriter LLM that encourages it to preserve the user's intent while making the query more explicit, structured, or detailed as needed. For example, the rewriter may be prompted with: "*Rewrite the following instruction via rephrasing and adding specific details to make it clearer, without changing its meaning:*" followed by the user's prompt. The rewritten output will then be fed into the main model to generate an answer. Figure 1 illustrates this LLM-in-the-loop prompting pipeline. We will compare this two-step approach with a direct prompting baseline.

6.1 LLMs for Prompt Rewriting and Answering

We identify specific LLMs to serve as the main answer-generating model and as the auxiliary prompt-rewriting model. Our choices prioritize models that are accessible via free or research-friendly APIs and can be run on modest hardware (such as a single NVIDIA T4 or L4 GPU in a Google Colab[1] environment). To enable comparative evaluation, we will experiment with at least three different LLMs:

- **meta-llama/Llama-3.2-1B[2]:** A 1.23-billion-parameter text-only member of Meta's Llama-3.2 release (Sept 2024) that targets mobile and edge deployment. It supports a 128 K-token context window, offers multilingual coverage (8 officially supported languages, expandable via fine-tuning), and inherits safety/RLHF alignment from its larger multimodal siblings. Its permissive community license and tiny footprint make it an ideal on-device prompt-rewriter or lightweight answer model.

- **microsoft/Phi-4-mini-instruct[3]:** Microsoft's newest 3.8 B-parameter decoder-only SLM with grouped-query attention, a 200 K-token vocabulary, and a 128 K-token context window. Trained with supervised fine-tuning and direct-preference optimization, it excels at math, logic, and function-calling while remaining fast enough for single-GPU or CPU inference—making it a strong candidate for rapid local prompt rewriting as well as answer generation.

- **google/gemma-3-1b-it[4]:** A 1 B-parameter instruction-tuned variant in Google's Gemma-3 family (Mar 2025) that brings Gemini-grade research to an open-weights model. Although the 1 B version is text-only (32 K-token context window), it shares the family's multimodal design philosophy, supports 140+ languages, and is optimized for laptops, desktops, and other resource-constrained devices—making it well-suited for on-device prompt-rewriting experiments or as a compact answer model. end

Using this selection, we will compare multiple configurations. For example, Llama-3.2-1B serves first as the prompt rewriter, followed by the same model acting as the answerer. The 1.23B-parameter model offers a huge 128K-token context window and is optimized for mobile or edge deployment, making it ideal for on-device rewriting and answering. By using the model in both roles, we test whether self-rewriting can enhance clarity and coherence when prompts and answers come from the same lightweight, efficient LLM.

Here, Phi-4 (3.8B parameters) rewrites prompts that are then passed to Llama-3.2-1B as the answer model. Phi-4 offers powerful logic and multilingual capabilities, a 128K-token context, and grouped-query attention, making it well-suited for generating clear, structured prompts. This configuration explores how a more capable rewriter can help a smaller model interpret and respond more effectively.

In this pairing, gemma-3-1b rewrites prompts for Phi-4, which then produces the answers. Gemma-3-1b is an instruction-tuned, 1B-parameter model with a 128K-token window, multilingual support (140+ languages), and efficiency tailored to laptops and resource-constrained environments. We aim to observe how a lightweight, multilingual

[1]https://colab.research.google.com
[2](Touvron et al., 2023)
[3](Abdin et al., 2024)
[4](Team et al., 2025)

prompt-rewriter influences the output quality of a mid-sized model.

Each configuration will be compared to a baseline where prompts are fed directly (no rewriting). We'll evaluate performance using consistent task sets and a mix of automatic metrics (e.g., math accuracy) and human assessments (clarity, correctness, helpfulness). This design will help us discern which models excel as prompt rewriters and how different pairings impact response quality.

If possible, we plan to use the trained auxiliary model to generate a dataset including the prompt and the rewritten prompt, and then train the main model on it using the DPO method to improve its accuracy compared to the previous state.

The accuracy of the other three language models is also measured using the original prompts and the rewritten prompts, and then the amount of improvement or lack of improvement will be noted in the final report.

The project will be implemented in colab and models will be used whose free API is available using Hugging Face[5].

7 Error analysis

editing this section ?????????????????????????

What kinds of inputs do your baselines fail at?

What about your approach?

Are there any semantic or syntactic commonalities between these difficult examples?

We would like to see a manual error analysis (e.g., annotate 100-200 failed examples for various properties, and then discuss the results and hypothesize about why the)

8 Contributions of group members

In this project, an attempt was made to carry out all the segments simultaneously in the laboratory and to the extent possible, not to work separately. However, considering the circumstances in the country that had led to changes in the project process, the following division can be made based on the percentage of progress of the different segments, in which the major contribution of each group member has been identified:

- segment 1: did data selection, data preprocessing and reports writing — done by **Vahid Moradi**
- segment 2: built and trained models and conclusion — done **collaboratively**
- segment 3: analyze the output of the model, error analysis and annotations — done by **Sina Hashemi**

[5]https://huggingface.co/models

9 Conclusion and Future Work

editing this section ?????????????????????????

You've now gotten your hands dirty with NLP tools and techniques!

What takeaways do you have about your project?

What proved surprisingly difficult to accomplish?

Were your surprised by your results?

If you could continue working on your project in the future, what directions would you pursue?

Table 1

9.1 Conclusion

In this project, we have outlined a method to improve LLM output quality by interposing an LLM-based prompt rewriter between the user and the answering model. By having one model rewrite the input prompt, we leverage the language understanding capabilities of LLMs to craft better queries for themselves or other models. This approach addresses a practical pain point: users often do not know the optimal way to ask a question to get a good answer. Our research investigate how much an automated rewriting step can help, under what conditions it works best, and where it might fall short.

We conduct experiments with several state-of-the-art and open LLMs on tasks that require reasoning, and use rigorous evaluation to measure

Table 1: Accuracy of models

Model	Initial accuracy	Accuracy after prompt rewriting	Accuracy after model training
Llama-3.2	40	38	42
Phi-4	**55**	50	52
Gemma-3	50	**60**	**54**

improvements. We contribute empirical evidence on the effectiveness of prompt rewriting. For instance, a positive result would be demonstrating a significant accuracy increase on GSM8K math problems when using an LLM rewriter, without fine-tuning the answering model at all. Such a finding would suggest a new, lightweight technique to boost performance: simply add an LLM prompt rewriter in front of any black-box model.

9.2 Future Work

There are multiple avenues for future work. Because our results indicate that prompt rewriting is beneficial, one could explore cascades of rewriters (where prompts are refined in stages) or specialized rewriters (where the rewriting LLM is fine-tuned or otherwise optimized for particular domains or user styles). It would also be interesting to apply prompt rewriting in another datasets and contexts like complex or descriptive question answering, code generation or complex interactive tasks, to see if a similar performance gain is observed. Additionally, user studies could be conducted to see if users prefer interacting with a system that automatically refines their questions — this could improve user satisfaction by reducing the need for them to iteratively figure out a good prompt. Also, the work done can be further expanded using various other datasets and the conclusions can be further generalized and expanded.

Another future direction is making the rewriting step more transparent and controllable. Users might want to see how their query was reinterpreted by the system. Providing the rewritten prompt back to the user for verification (especially in high-stakes scenarios) could be a way to maintain trust and allow corrections if the rewriter misunderstood the intent. Incorporating user feedback to iteratively improve the rewritten prompt could combine human insight with LLM power for even better outcomes.

In summary, this project explore a novel facet of human-LLM interaction: using one AI to help communicate with another. Prompt rewriting by LLMs could become a broadly applicable technique to enhance the performance of large language models on a wide array of tasks, effectively making them not just generators of answers, but also editors of questions. The insights gained will not only improve our understanding of prompt dynamics in AI systems but also yield immediately practical tools for deploying LLMs in real-world applications where end-users can benefit from higher quality AI assistance.

References

Abdin, M., Aneja, J., Behl, H., Bubeck, S., Eldan, R., Gunasekar, S., Harrison, M., Hewett, R. J., Javaheripi, M., Kauffmann, P., Lee, J. R., Lee, Y. T., Li, Y., Liu, W., Mendes, C. C. T., Nguyen, A., Price, E., de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Wang, X., Ward, R., Wu, Y., Yu, D., Zhang, C., and Zhang, Y. (2024). Phi-4 technical report.

Cheng, J., Liu, X., Zheng, K., Ke, P., Wang, H., Dong, Y., Tang, J., and Huang, M. (2024). Black-box prompt optimization: Aligning large language models without model training. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3201–3219, Bangkok, Thailand. Association for Computational Linguistics.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems.

Kong, W., Hombaiah, S., Zhang, M., Mei, Q., and Bendersky, M. (2024). PRewrite: Prompt rewriting with reinforcement learning. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 594–601, Bangkok, Thailand. Association for Computational Linguistics.

Li, C., Zhang, M., Mei, Q., Kong, W., and Bendersky, M. (2024). Learning to rewrite prompts for personalized text generation. In *Proceedings of the ACM Web Conference 2024*, WWW '24, page 3367–3378. ACM.

Ma, X., Gong, Y., He, P., Zhao, H., and Duan, N. (2023). Query rewriting for retrieval-augmented large language models.

Sarkar, R., Sarrafzadeh, B., Chandrasekaran, N., Rangan, N., Resnik, P., Yang, L., and Jauhar, S. K. (2025). Conversational user-ai intervention: A study on prompt rewriting for improved llm response generation.

Srivastava, S., Huang, C., Fan, W., and Yao, Z. (2024). Instances need more care: Rewriting prompts for instances with LLMs in the loop yields better zero-shot performance. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6211–6232, Bangkok, Thailand. Association for Computational Linguistics.

Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., bastien Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.-T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A. M., Goedeckemeyer, A., Saade, A., Feng, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A. S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C. L., Choquette-Choo, C. A., Carey, C., Brick, C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D. S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F., Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Plucińska, H., Singh, H., Mehta, H., Lehri, H. T., Hazimeh, H., Ballantyne, I., Szpektor, I., Nardini, I., Pouget-Abadie, J., Chan, J., Stanton, J., Wieting, J., Lai, J., Orbay, J., Fernandez, J., Newlan, J., yeong Ji, J., Singh, J., Black, K., Yu, K., Hui, K., Vodrahalli, K., Greff, K., Qiu, L., Valentine, M., Coelho, M., Ritter, M., Hoffman, M., Watson, M., Chaturvedi, M., Moynihan, M., Ma, M., Babar, N., Noy, N., Byrd, N., Roy, N., Momchev, N., Chauhan, N., Sachdeva, N., Bunyan, O., Botarda, P., Caron, P., Rubenstein, P. K., Culliton, P., Schmid, P., Sessa, P. G., Xu, P., Stanczyk, P., Tafti, P., Shivanna, R., Wu, R., Pan, R., Rokni, R., Willoughby, R., Vallu, R., Mullins, R., Jerome, S., Smoot, S., Girgin, S., Iqbal, S., Reddy, S., Sheth, S., Põder, S., Bhatnagar, S., Panyam, S. R., Eiger, S., Zhang, S., Liu, T., Yacovone, T., Liechty, T., Kalra, U., Evci, U., Misra, V., Roseberry, V., Feinberg, V., Kolesnikov, V., Han, W., Kwon, W., Chen, X., Chow, Y., Zhu, Y., Wei, Z., Egyed, Z., Cotruta, V., Giang, M., Kirk, P., Rao, A., Black, K., Babar, N., Lo, J., Moreira, E., Martins, L. G., Sanseviero, O., Gonzalez, L., Gleicher, Z., Warkentin, T., Mirrokni, V., Senter, E., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Matias, Y., Sculley, D., Petrov, S., Fiedel, N., Shazeer, N., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Alayrac, J.-B., Anil, R., Dmitry, Lepikhin, Borgeaud, S., Bachem, O., Joulin, A., Andreev, A., Hardin, C., Dadashi, R., and Hussenot, L. (2025). Gemma 3 technical report.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models.