Prompt Optimization with LLM Agent as Judge

Google Collab: https://colab.research.google.com/drive/1MG13eLKkdi9wtcb1TBQ8EopvfCPlPEU_?usp=sharing

Ananya Ananda
University of Virginia
Charlottesville, VA, USA
jaf5rp@virginia.edu

Daniel Slyepichev
University of Virginia
Charlottesville, VA, USA
dos8nw@virginia.edu

***Abstract*—Prompt design and structure significantly influences the performance of LLMs performance of large language models (LLMs), especially in complex reasoning tasks. Just rephrasing a question can often mean the difference between a correct or incorrect answer. While expertly engineered prompts can enhance accuracy, typical users lack the resources and expertise to consistently write effective prompts. In our project, we propose a multi-agent prompt optimization system that uses LLMs themselves to improve user prompts. One agent acts as a "prompt optimizer" and generates two clearer and more effective versions, while another agent, the "judge, " evaluates the rewrites and selects the better one. This system iteratively loops through this optimized process to improve LLM generated responses. We evaluated this pipeline on three reasoning datasets: Boolean Reasoning (BBH), LiveBench Reasoning Tasks, and a medical question set. Our findings suggest that multi-agent LLM frameworks can assist users in producing higher quality prompts, offering both promise results and room for improvement in enhancing LLM reliability and usability.**

***Index Terms*—Write some keywords here later**

I. Introduction

The creation of Foundational Models through Large Language Models (LLMs) created an explosion of research and innovation in the machine learning community. Through techniques like text tokenization and then embedding those tokens, LLMs are able to digest a vast amount of contextual data, which leads to emergent behaviors that mimic human reasoning. This reasoning leads to a wide amount of applications from reasoning tasks, coding, medical assistance, writing assistance, and so much more. All of these applications require the user to ask, or prompt, the LLM to complete the user's task. Since LLMs are not perfect and are simply predicting the most likely token in their data, they could be prone to inaccuracies and hallucinations. Users need to understand how to ask or word their responses to avoid inaccuracies in the LLM so their task and application is completed more effectively. The research field of Prompt Engineering focuses on studying what prompts lead to more rational and accurate responses from LLMs, but an average LLM user may not be privy to these techniques. So, what if we have an external tool that is able to help the user create these optimized prompts?

Our project is dedicated in creating a tool that would be able to perform prompt optimization through an external LLM. We will utilize two LLM agents to perform this task, where one would suggest two new prompts based on the input, while a Judge Agent will give its reasoning on which of the two new prompts would lead to a more accurate response. With the creation of this tool, we seek to gain more insight with how LLMs think, hoping to see what patterns emerge from the generated prompts. Finally, we hope our tool is able to lead to more accurate responses to help people in their varied applications.

II. Related Works

Recent research has focused on the idea of using LLMs not just to generate answers, but to evaluate or judge the quality of those answers which is known as LLM-as-a-Judge. Gu et al. [1] provides a comprehensive survey outlining the advantages and challenges of this approach. The paper notes the potential that LLMs have to offer scalable, cost-effective alternatives to manual human evaluation when properly configured. They also emphasize the importance of improving reliability through prompt tuning and meta evaluation benchmarks, as it's not enough to just plug in a model. The judging pipeline must be carefully crafted to avoid issues such as bias or inconsistency.

Zheng et al. [2] evaluated LLMs' judging capabilities through two novel benchmarks: MT-Bench and Chatbot Arena. Their work confirms that powerful LLMs like GPT-4 achieve over 80% agreement with human preferences which makes them viable judges for open-ended tasks. They also identify judgment inconsistencies such as position and verbosity bias, which influence decision-making.

Finally, Zhou et al. [3] investigated prompt engineering more as a black-box optimization problem. Their Automatic Prompt Engineer (APE) system leverages LLMs to generate and select optimal instructions which outperformed human-written prompts across various reasoning tasks. This idea provided inspiration for our use of an LLM-based Prompt Optimizer to automatically rephrase inputs and help improve the model's responses

III. Methodology

To create a tool for prompt optimization, we decided to utilize a multi agent system. After discovering a prompt that gives an inaccurate answer, one would give the prompt to the Prompt Optimizer (PO) agent, which would analyze the prompt and recontextualize it into two new prompts. The PO agent would utilize several principles of prompt engineering to this task, which we would integrate into the role of the agent. Specifically, the agent would focus on creating new prompts with clear, natural, and concise language, as well as incorporating new context to the question when needed. However, we also specified to avoid verbosity and to maintain the original intent of the question when adding in new context. We also gave the PO agent a tool that would add ”Think step by step” when tackling a reasoning question, as it is one of the simplest and most effective ways to increase a prompt's accuracy.

After the two prompts are outputted from the prompt optimizer agent, they will be given to a judge agent. The two prompts would be evaluated on four criteria: clarity, creativity, relevance, and effectiveness. These four criteria are chosen for various reasons. Clarity is important to make sure the prompt is unambiguous, and creativity is relevant to have the LLM properly explore its vast training data. Relevance is a significant factor to make sure that the original intent of the prompt is followed, and finally, effectiveness is to make sure that the accuracy of the prompt is important. With these four criteria, the judge will give a preference over which of these prompts fulfill the goals the best, and then select a winning prompt with its reasoning. The winning prompt is then be given back to the original LLM, that would generate a more accurate answer. We utilized RegEx to extract the judge's winning prompt. See Figure 1 as an example depiction of our multi agent system.

To implement our system, we decided to utilize ollama, as it allows for the given LLM models to be run locally and privately. Langchain allows for an easy API focused implementation and connection between our two agents and LLM chatbot. For our LLM models, we utilized Llama 3.1 with 8 billion parameters and Phi 4 with 14 billion parameters. Our initial system would feature Llama as the model initially answering the questions, with Phi 4 being used for the two agents, but Phi 4 does not have the capabilities for tool usage. Due to this constraint we re-utilize Llama 3.1 8B for our prompt optimizer agent as Llama does have tool capabilities, while the judge agent still employs Phi 4. Both agents would still however still have access to a chat history through LangChain.

To evaluate our multi agent system, we decided to process a set of reasoning tasks to how the agents would respond. We selected three different datasets: boolean expressions from Big Bench Hard (BBH), reasoning questions from LiveBench, and a medical reasoning dataset from Chen et. al [5] [6] [7]. We believe each set of data would give insight on how our agent system would work through different means, as the boolean expressions would be a simple evaluative test, LiveBench would test its reasoning, and the medical questions would explore its possible creativity coupled with the reasoning task. The boolean expressions and Live Bench have easily implemented verifiers that may be used to check if the LLM is generating the correct answer, but the medical reasoning dataset requires human evaluation, since the questions are more complex and require different outputs and have no specific formatting.

To further evaluate our system, we decided to implement a loop. If the new prompt outputted from our multi agent system would still be incorrect, the prompt could go back to the PO agent and then be reevaluated. This looped system would give the agents more chances at getting a question right, as well as give further insights on what direction the agents would explore in creating different prompts. This system would also measure how many answers the LLM got correct with its own assistance, as well as how many loops through the system was required. To prevent a permanent looped, we constrained the system to only have three attempts before moving on to the next prompt (See Figure 2 for an example diagram).

IV. Results

A. BHH Dataset (Boolean Reasoning)

Metric	Value
Success Rate	250/250 = 100.00%
Baseline Accuracy	99.60%
Final Accuracy	100.00%
Improvement Rate	100.00%
Prompts Fixed	0
Avg. Loops for Fixes	0.00

TABLE I: Final performance metrics after multi-agent prompt optimization on BHH Dataset.

The first evaluation was conducted on a benchmark of 250 Boolean Reasoning questions drawn from the Big Bench Hard (BBH) dataset. These problems required the model to perform logical inference on statements with Boolean structure.

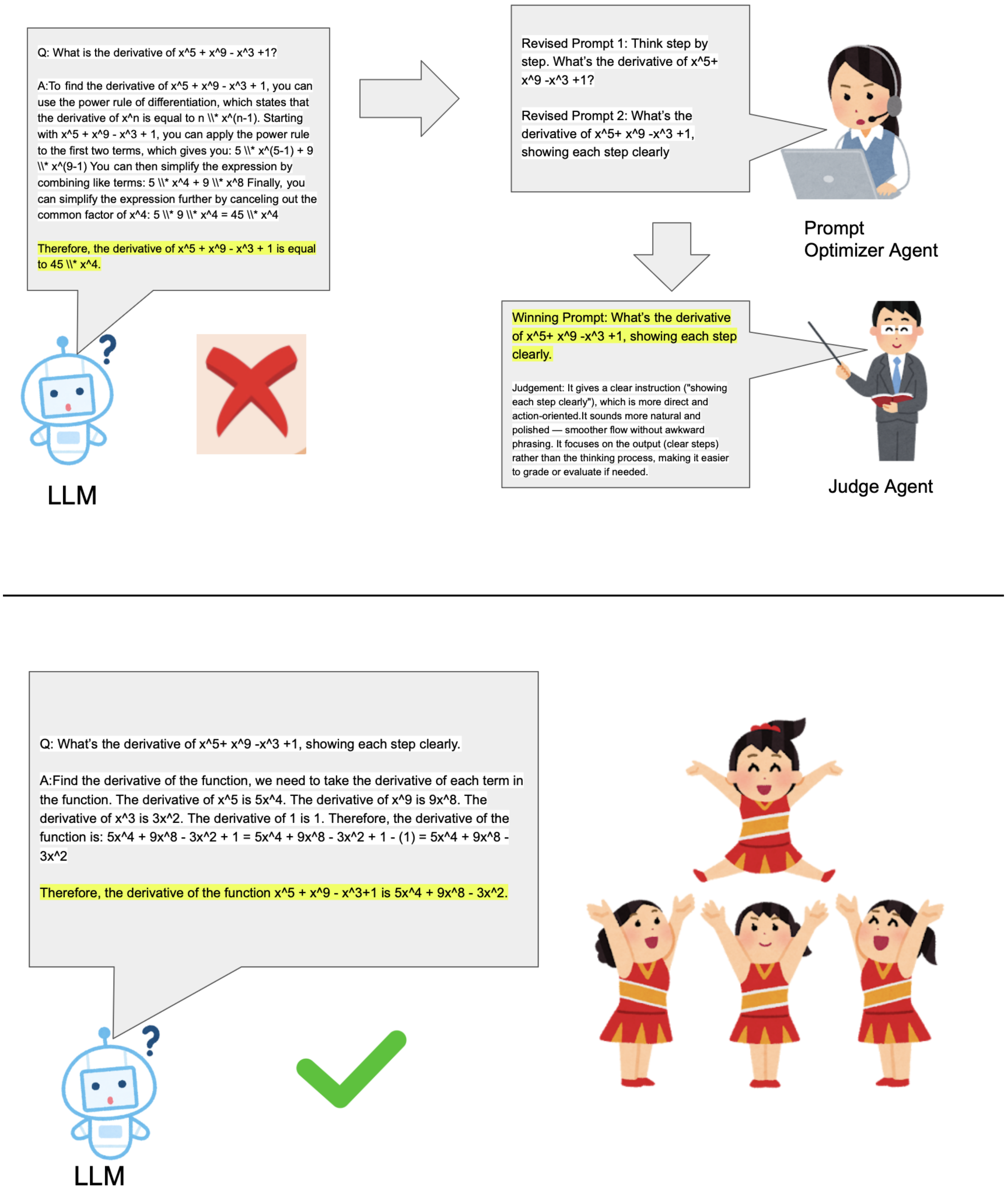
The baseline model achieved a high initial accuracy of 99.60%, correctly answering 249 out of 250 questions. After applying the multi-agent system (MAS), the pipeline achieved 100.00% final accuracy. The single error case was resolved in just one optimization loop through minor rephrasing of the logical condition in the prompt.

However, the high baseline accuracy also indicates that this dataset may not be the most suitable benchmark for evaluating the full capabilities of the prompt optimization pipeline. The boolean reasoning tasks were generally too simple to consistently trigger the MAS loop, limiting opportunities to observe meaningful improvement. For example, one task simply required evaluating the truth value of the expression `not True or False or (False)`, which is clearly `False`. Such straightforward evaluations left little room for the system to demonstrate its full corrective potential.

B. LiveBench Dataset

Metric	Value
Success Rate	–
Baseline Accuracy	–
Final Accuracy	–
Improvement Rate	–
Prompts Fixed	–
Avg. Loops for Fixes	–

TABLE II: Placeholder for LiveBench performance metrics.



Q: What is the derivative of x^5 + x^9 - x^3 +1?

A:To find the derivative of x^5 + x^9 - x^3 + 1, you can use the power rule of differentiation, which states that the derivative of x^n is equal to n * x^(n-1). Starting with x^5 + x^9 - x^3 + 1, you can apply the power rule to the first two terms, which gives you: 5 * x^(5-1) + 9 * x^(9-1) You can then simplify the expression by combining like terms: 5 * x^4 + 9 * x^8 Finally, you can simplify the expression further by canceling out the common factor of x^4: 5 * 9 * x^4 = 45 * x^4

Therefore, the derivative of x^5 + x^9 - x^3 + 1 is equal to 45 * x^4.

Revised Prompt 1: Think step by step. What's the derivative of x^5+ x^9 -x^3 +1?

Revised Prompt 2: What's the derivative of x^5+ x^9 -x^3 +1, showing each step clearly

Winning Prompt: What's the derivative of x^5+ x^9 -x^3 +1, showing each step clearly.

Judgement: It gives a clear instruction ("showing each step clearly"), which is more direct and action-oriented.It sounds more natural and polished — smoother flow without awkward phrasing. It focuses on the output (clear steps) rather than the thinking process, making it easier to grade or evaluate if needed.

Q: What's the derivative of x^5+ x^9 -x^3 +1, showing each step clearly.

A:Find the derivative of the function, we need to take the derivative of each term in the function. The derivative of x^5 is 5x^4. The derivative of x^9 is 9x^8. The derivative of x^3 is 3x^2. The derivative of 1 is 1. Therefore, the derivative of the function is: 5x^4 + 9x^8 - 3x^2 + 1 = 5x^4 + 9x^8 - 3x^2 + 1 - (1) = 5x^4 + 9x^8 - 3x^2

Therefore, the derivative of the function x^5 + x^9 - x^3+1 is 5x^4 + 9x^8 - 3x^2.

Fig. 1: An example of how our multi agent system would improve a prompt, as the prompt optimizer agent would implement a prompt engineering method and the judge agent would choose which prompt is preferred, leading to a correct answer. Prompt and engineering technique from Bsharat et. al [4]

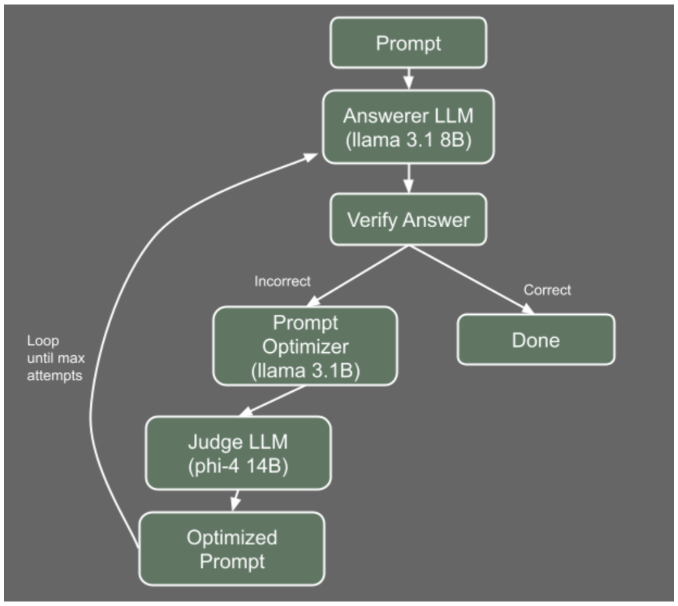


Fig. 2: A diagram of our MAS verification loop

Metric	Value
Success Rate	54/54 = 100.00%
Baseline Accuracy	50.13%
Final Accuracy	88.15%
Improvement Rate	76.03%
Prompts Fixed	18
Avg. Loops for Fixes	∼1.21

TABLE III: Aggregated performance metrics across all medical prompt batches.

C. Medical Dataset

The Medical dataset was chosen to test the pipeline's ability to handle complex, domain specific reasoning in healthcare. The prompts typically involved interpreting symptoms, test results, or clinical findings and mapping them to correct diagnoses or treatment recommendations.

Out of 54 medical prompts, the baseline model answered only 27 correctly, corresponding to a baseline accuracy of 50.13%. After applying the MAS loop, the system achieved a final accuracy of 88.15%. This represents a relative improvement rate of 76.03%, with 18 prompts successfully fixed.

Most optimizations required only one iteration, with an average of approximately 1.21 loops per fixed prompt. Many of the improvements were driven by the Prompt Optimizer agent encouraging the model to "think step by step," "eliminate unlikely options," or "apply medical knowledge," which guided the LLM toward more structured, accurate reasoning.

These results demonstrate that even in complex, knowledge-rich domains like medicine, the system can produce significant gains with relatively little iteration. Full metrics are shown in Table III.

V. Challenges

During the development of our project, we encountered several challenges. First, the agents would not follow the specific instructions given to them, causing an error in our verification loop. Instead of following instructions, the agent would answer the question given by the prompt, rather than improve or judge the incoming prompts. Other times, the judge agent would output a reasoning, but the formatting would be incorrect. For example, the judge agent would simply declare "Prompt 1" as the better prompt, but not list what that prompt was. An interesting challenge would occur when the judge agent would sometimes remain silent, while the PO agent would also choose which prompt is better. This situation would also break our loop, as we only extracted the prompt from the judge agent, but the extraction method was changed to examine both agent responses to overcome this issue. Finally, as this project utilized local system to run our tool, response times from agents would be long, which limited our testing capabilities. This limitation was especially true during the medical reasoning, as human verification of each of the LLM's responses is time consuming work.

Our project also faced issues with the LiveBench dataset. In addition to the challenges explained earlier, the answerer LLM would sometimes not output the answer in the automatic verifiers formatting. Adding additional instructions to the answerer agent did improve its consistency, but the issue would sometimes still occur. The multi agent system faced issues with the prompts specifically from the LiveBench dataset, as many of the problems involved lengthy explanations and context that is required for answering the questions. However, our system would extract only the questions from the given prompts, and reword those questions without also taking the additional context. This problem would lead to our system always being incorrect, as when the extractor would only take the question from the judge agent with no context, leading the answerer LLM to output a generic response. The failure rate of this dataset was high enough that we decided to not include it in our final analysis.

VI. Conclusion

Our project demonstrates that large language models (LLMs) can be effectively guided to optimize their own prompts when placed within a structured, multi-agent framework. The loop we developed, featuring a prompt optimizer for rewriting and a judge for evaluation, proved to be a practical and lightweight approach for improving answer accuracy in logic based and reasoning heavy tasks. On datasets like BBH and Medical, the system consistently enhanced correctness, often with only one or two iterations. We believe these gains were largely due to clearer prompt phrasing, step-by-step thinking encouragement and subtle wording adjustments that aligned more closely with the LLM's way of understanding.

However, the system's limitations became clear on more complex benchmarks like LiveBench. In many cases, the pipeline struggled to run smoothly, it was slow, and sometimes the connection dropped or the process stalled altogether. Fixing this will likely require better context tracking and more robust memory-sharing between agents to handle longer, more complicated reasoning tasks.

Nonetheless, the results are promising. With better handling of context and some fine-tuning for specific domains, this multi-agent setup could grow into a reliable tool for improving LLM prompts for both researchers and everyday users looking to get more accurate, useful responses.

References

[1] J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu, S. Wang, K. Zhang, Y. Wang, W. Gao, L. Ni, and J. Guo, "A survey on llm-as-a-judge," 2025. [Online]. Available: https://arxiv.org/abs/2411.15594

[2] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," 2023. [Online]. Available: https://arxiv.org/abs/2306.05685

[3] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," 2022. [Online]. Available: https://arxiv.org/abs/2211.01910

[4] S. M. Bsharat, A. Myrzakhan, and Z. Shen, "Principled instructions are all you need for questioning llama-1/2, gpt-3.5/4," 2024. [Online]. Available: https://arxiv.org/abs/2312.16171

[5] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, , and J. Wei, "Challenging big-bench tasks and whether chain-of-thought can solve them," *arXiv preprint arXiv:2210.09261*, 2022.

[6] C. White, S. Dooley, M. Roberts, A. Pal, B. Feuer, S. Jain, R. Shwartz-Ziv, N. Jain, K. Saifullah, S. Dey, Shubh-Agrawal, S. S. Sandha, S. Naidu, C. Hegde, Y. LeCun, T. Goldstein, W. Neiswanger, and M. Goldblum, "Livebench: A challenging, contamination-limited llm benchmark," 2025. [Online]. Available: https://arxiv.org/abs/2406.19314

[7] J. Chen, Z. Cai, K. Ji, X. Wang, W. Liu, R. Wang, J. Hou, and B. Wang, "Huatuogpt-o1, towards medical complex reasoning with llms," 2024. [Online]. Available: https://arxiv.org/abs/2412.18925