The Impact of Prompt Engineering on GPT-4’s Mathematical Reasoning

Saanvi C. Hublikar

Mission San Jose High School, Fremont, California, United States, 94539

KEYWORDS. Artificial Intelligence, Mathematical Reasoning, Large Language Model.

BRIEF. This study uses different methods of prompting to improve large language models’ mathematical reasoning.

ABSTRACT. Artificial intelligence (AI) has the potential to revolutionize science and technology, but even many of the latest AI systems have difficulty solving general math problems due to limited reasoning skills and training data [1]. This study utilizes prompt engineering to understand which methods of prompting are most effective in assisting generative large language models such as GPT-4 in solving math problems. The methods tested were reflection, planning, multi-agent collaboration, and one- and few-shot prompting. Effectiveness of the first three methods was measured through the qualitative analysis of the LLM’s response to Artificial Intelligence Math Olympiad Problems. Success of one- and few-shot problem solving was measured by the percentage of the Massive Multitask Language Understanding (MMLU) problems that the LLM solved. It was seen that while planning and reflection solved problems with similar accuracy as basic, unenhanced prompting, both methods forced ChatGPT to work through the problem in a more organized and methodical manner, improving the model’s reasoning. However, neither method fixed major reasoning errors, especially in cases where the model approached the problem in a completely incorrect way. Similarly, multi-agent collaboration incorporated these methods and provided more structure and logic to the problem-solving process but led to inconsistent results. This method proved to be finicky and did not improve the model’s accuracy overall. One-shot and five-shot prompting, on the other hand, were seen to improve the accuracy of the model.

INTRODUCTION.

Although ChatGPT and other AI systems are widespread today, most LLMS still struggle with mathematical reasoning. When tested on the Massive Multitask Language Understanding (MMLU) dataset, which is used in this study to measure the success of one-shot and few-shot prompting, GPT-3 performed poorly in solving high school mathematics problems [2]. Calculation-based subjects like math and physics showed low accuracy; the success rate when solving high school mathematics questions averaged below 30% [2]. This is likely because GPT-3 acquires declarative knowledge better than procedural knowledge, which can be seen in its higher performance in information-based subjects like high school psychology (~60%), high school government, politics, and geography (~58%), and international law (~56%) [2].

The model used in this work, GPT-4, was previously tested on SAT math and grade-school mathematics and was tested with an overall math factuality evaluation. It scored a 700 out of 800 on SAT math, an 89th percentile mark [3]. This is a significant improvement when compared to GPT-3’s score of 590 (~70th percentile) [3]. When tested on grade school mathematics with five-shot prompting, a method tested in this study in comparison with zero-shot and one-shot prompting, GPT-4 showed a 92% success rate, a major improvement from GPT-3’s 57% accuracy with the same prompting [3]. However, on the overall math factuality evaluation, GPT-4 had a less than 70% success rate, lower than categories like history and science, which are based more on declarative knowledge than reasoning [3].

Some methods of prompting have been tested recently to improve LLMs’ performance. When GPT-3 was tested on closed book questions answering with zero-, one-, and few-shot prompting, the model showed an overall performance improvement of 3.7% accuracy for one-shot prompting and an additional 3.2% for few-shot prompting [4]. However, it is important to note that these tasks do not require mathematical reasoning and therefore cannot be used as a direct comparison to the results of this study. When tested with common sense reasoning tasks, GPT-3 showed a much lower performance improvement (~2.9% from zero-shot to few-shot prompting) [4].

So far one of the most successful models is the breakthrough model AlphaProof, created by fine-tuning a Gemini model [1]. This model trains itself to prove mathematical statements using a reinforcement learning-based algorithm for formal mathematical reasoning [1]. While proofs involving mathematical reasoning can be formally identified for correctness, they are constrained by a limited amount of written human data [1]. Natural language-based approaches, on the other hand, have much more data but can hallucinate plausible answers; in other words, the LLM will provide incorrect or misleading results and present them as facts [1]. The AlphaProof model combines these approaches by translating natural language problems to formal language, generating solutions, and then proving or disproving them. These proofs are then used to reinforce the model and improve its performance [1].

MATERIALS AND METHODS.

Dataset. Two datasets of math problems were used to check the AI model’s accuracy. The training problem set from the 2024 Artificial Intelligence Mathematical Olympiad (AI|MO) is a set of 10 training problems publicly available through Kaggle. The dataset contains the ID, problem, and answer for each question [7]. The MMLU high school mathematics test dataset is a set of 270 high school level math problems used as a benchmark to measure a LLM’s problem solving ability with one-shot prompting [5]. This dataset contains math problems, four possible answer choices from A-D, and the solution for each question. All 270 problems from this dataset were used to find GPT-4's success rates for one-shot and five-shot prompting.

All 10 AI|MO problems were tested, but the responses to 6 problems were further studied because ChatGPT was unable to provide any solution, whether correct or incorrect, to the other problems. The following is a list of the 6 important AI|MO problems referenced in this study [7]:

Problem 1: Let k, $l > 0$ be parameters. The parabola

$$y = kx^2 - 2kx + l \tag{1}$$

intersects the line $y = 4$ at two points A and B. These points are distance 6 apart. What is the sum of the squares of the distances from A and B to the origin?

Problem 2: Each of the three-digits numbers 111 to 999 is colored blue or yellow in such a way that the sum of any two (not necessarily different) yellow numbers is equal to a blue number. What is the maximum possible number of yellow numbers there can be?

Problem 5: There exists a unique increasing geometric sequence of five 2-digit positive integers. What is their sum?

Problem 6: For how many positive integers *m* does the equation

$$\left||x-1|-2\right| = \frac{m}{100} \tag{2}$$

have 4 integer solutions?

Problem 7: Suppose that we roll four 6-sided fair dice with faces numbered 1 to 6. Let *a*/*b* be the probability that the highest roll is a 5, where *a* and *b* are relatively prime positive integers. Find *a* + *b*.

Problem 9: Let *ABCD* be a unit square. Let *P* be the point on *AB* such that

$$|AP| = \frac{1}{20} \tag{3}$$

and let *Q* be the point on *AD* such that

$$|AQ| = \frac{1}{24} \tag{4}$$

The lines *DP* and *BQ* divide the square into four regions. Find the ratio between the areas of the largest region and the smallest region.

Basic Prompting. An OpenAI agent was created with the following system prompt: "You are an expert mathematician who is seasoned at solving problems and explaining your solutions." AI|MO problems were given exactly how they appeared in the training problem set, preceded by the prompt "Solve this problem."

Reflection. Reflection prompts the LLM to review its work instead of generating the final output in the first try, which often leads to preventable errors. In the initial stage, this method was implemented by adding the phrase "Solve this problem, then revise your solution," to the agent's prompt. Then, more detailed instructions were included (see *Reflection #2* in Table 1 for details). Reflection was also utilized through multi-agent collaboration (see *Multi-Agent Collaboration*).

Planning. Planning prompts the LLM to create a list of steps before solving a problem, leading to a more efficient agentic workflow. The creative problems provided by the AI|MO problem set require reasoning and multiple mathematical skills and therefore cannot be solved in a single step; however, these steps cannot be specified in the prompt due to how varied the problems and skill sets required to solve them are. Reflection was also utilized through multi-agent collaboration (see *Multi-Agent Collaboration*).

Multi-Agent Collaboration. The first method of multi-agent collaboration was through conversable agents by Autogen, an open-source programming framework for agentic AI by Microsoft [6]. ConversableAgent is a general class for agents capable of exchanging messages to collaborate to perform a task. This was used to create two agents initialized as "expert mathematicians" and prompted to chat with each other to plan out the solution, solve the problem, and correct each other's solutions until they reached the correct answer. Certain trials specifically prompted the agents to use planning or reflection when solving the problem.

The second method of multi-agent collaboration utilized Autogen's group chat tool. Four agents were created: an admin that was tasked with presenting the problem, a planner that listed the steps to the problem, a calculator that solved the problem based on the planner's steps, and a checker that revised the calculator's solution. This method implemented planning and reflection through the planner and checker agents. The group chat method was initially implemented with Autogen's GroupChatManager agent having full control over which agents were called to speak.

The final tested method of multi-agent collaboration was a further restricted version of the group chat method. Instead of using Autogen's tools, agents were created with highly specific system prompts and instructions. Similarly to the group chat method, a planner, mathematician, and checker were created; however, the admin and GroupChatManager were removed and replaced by a hard-coded speaking order. Two additional agents were also created, one with the purpose of revising the plan and the other tasked with summarizing the final solution. This method began with the planner creating a step-by-step plan for solving the given problem. This plan was then passed to another agent which either revised the plan or decided that the original plan was efficient. This new plan was then given to a mathematician agent, which solved only the first step of the problem. The checker then revised that single step before passing it back to the mathematician to solve the next step. This created a back and forth between the two agents until a list of the checker's revised solutions to each step was passed to the final agent to summarize the work and provide the solution.

One-Shot and Few-Shot Prompting. One-shot and few-shot prompting were tested on the MMLU high school mathematics dataset, and data was collected to find the success rate of each method. Accuracy was recorded for the following conditions: basic prompting, asking to insert scratchwork, one-shot prompting, and five-shot prompting.

Table 1. Overview: Methods of Prompting.

Method	Example/Explanation
Basic Prompting	"Solve this problem: [AI|MO problem]"
Reflection #1	"Solve this problem, then revise your solution: [AI|MO problem]"
Reflection #2	"1) Solve this problem: [AI|MO problem] 2) Now assume that this solution is incorrect. List out all the mistakes you made when solving this problem. 3) Solve the problem again, considering the mistakes you found in your original solution."
Planning	"1) Consider this problem: [AI|MO problem] 2) First list out the steps you will take to solve the problem. 3) Solve the problem."
Multi-Agent Collaboration #1	Two Autogen conversable agents both initialized as "expert mathematicians"
Multi-Agent Collaboration #2	Four Autogen conversable agents initialized as an admin, planner, calculator, and checker; regulated by Autogen's GroupChatManager
Multi-Agent Collaboration #3	Five Autogen conversable agents initialized as a planner, calculator, plan checker, calculation checker, and summarizer; regulated by a hard-coded speaking order
One-shot Prompting	"Solve this problem: [MMLU problem] Here is an example of how to solve a problem: Example problem: [different MMLU problem] Example solution: [human solution to the example problem]"
Five-shot Prompting	Similar to one-shot prompting but with five examples rather than one

RESULTS.

Basic Prompting. When provided with only a question from the AI|MO dataset and no further instructions, GPT-4 was able to solve only Problem 6, albeit with inconsistent results.

Common mistakes included a tendency to misunderstand or ignore specific requirements of the problem, which can be seen in problems 1 and 5. Though problem 1 required an integer solution, GPT-4 provided an answer in terms of the variable *k* used in the question. In problem 5, which specifies a "unique increasing geometric sequence of five 2-digit positive integers," GPT-4 responses in many trials included non-integer terms. The model also arbitrarily assigned incorrect values without explanation; for example, in problem 5, it suggested that the "geometric sequence could be identified starting with 'a' as 12 and 'r' as ⅔," then concluded that the problem could not be solved because this would result in a decreasing geometric sequence. Other mistakes included executing the wrong steps to solve the problem or misunderstanding a graphical representation of a question.

Reflection. The reflection method yielded similar results; however, it was commonly able to identify its mistakes even if it was not able to provide a final correct solution. We can look at problem 5, where GPT-4 originally generated a response that set the value of the ratio to be 1, which does not fulfill the definition of an increasing geometric sequence. When revising its answer, it pointed out this mistake and correctly changed the ratio to 1.5. However, it erred when finding the sequence's first term and therefore could not provide the correct solution.

Planning. The planning method also generated similar results to basic prompting and reflection. However, it was seen that prompting the LLM to create a list of steps beforehand almost always resulted in an efficient and logical plan, which was lacking from GPT-4's responses to basic prompting. With planning, the issue lay instead in the execution of the plan—GPT-4 commonly ignored requirements of the problem despite them being specified in the plan or made a calculation or reasoning error while solving the problem.

Trials from problem 5 can be examined to determine planning's effect on LLM responses. In trial 1 of the planning method, ChatGPT detailed a plan based on guessing and checking values for the first term and ratio of the sequence which should have yielded an accurate solution. However, after multiple rounds of guessing and checking, the model ignored one of the major requirements of the problem—that all terms of the sequence must be 2-digit numbers—and provided a sequence involving 1-digit terms. In trial 2, ChatGPT generated the correct ratio but automatically assigned the first term of the sequence to be 10 because it is the smallest 2-digit number, resulting in an incorrect answer.

Multi-Agent Collaboration. Multi-agent collaboration provided better results than the previous two due to its more efficient incorporation of planning and reflection, though it was the most finicky method. Compiled results from conversable agents, task decomposition, and restricted agents show that when multiple agents were asked to work together to solve the problem, the model was able to solve problems 1, 4, 5, 6, and 7 from the AI|MO training dataset at least once. Conversable agents by Autogen solved problem 1 through planning and reflection: agents provided a list of steps, executed them to reach an incorrect solution, and revised the solution to generate the right answer. Similar processes occurred when the task decomposition method was utilized with problems 1, 5, and 6, and when the restricted agent's method was used to solve problems 1 and 7.

However, when other correct responses were further analyzed, the results showed that it was not always the method itself that generated improved responses. The Autogen conversable agents solved problems 5 and 6 in one attempt using the planner agent, similarly to problems 4 and 7 in task decomposition. Additionally, agents were not performing the tasks they had been assigned through their system prompt; both correct and incorrect responses showed the planner and checker agents solving the problems without delegating any tasks to the calculator method. To limit this, agents were restricted with hard-coded speaking orders and response permissions.

Despite this, restricted agents yielded similar results to conversable agents and task decomposition. The scratchwork was improved, as agents were performing the correct roles and switching speakers between each step of the problem; however, this did not improve accuracy in the final answers.

One-Shot and Few-Shot Prompting. When tested on a randomly selected 100 problems from the MMLU high school mathematics dataset, GPT-4 with basic, unenhanced prompting had a 69% success rate. When prompted to insert scratchwork, GPT-4 yielded similar results. With one-shot prompting, where one example question from the dataset was solved and provided in the prompt, the LLM solved them with a 74% success rate, and five-shot prompting produced 75% accuracy.

Table 2. Overview: Results of Prompting.

Method	Solved Problems/Success Rate
Basic Prompting	AI|MO Problem 6 (inconsistent results)
Reflection (#1 & #2)	No solved problems, but it generally produced closer results than basic prompting
Planning	AI|MO Problem 6
Multi-Agent Collaboration (#1, #2, & #3)	AI|MO Problems 1, 4, 5, 6, & 7 (inconsistent results)
One-shot Prompting	74% success rate for MMLU problems (compared to a 69% success rate for prompting with no examples)
Five-shot Prompting	75% success rate for MMLU problems

DISCUSSION.

Overall, it can be determined that of all the methods tested, one-shot and few-shot prompting most consistently improved LLM performance when solving math problems. These methods showed steady, improved results throughout multiple trials. One-shot prompting showed an average improvement in accuracy of around 5%, while few-shot prompting increased it by another 1%.

While GPT-4 was unable to consistently solve any of the AI|MO testing problems on its own, planning and reflection did not significantly improve its accuracy when these methods were applied individually. However, performance improved when these methods were incorporated into multi-agent collaboration, though it is important to note that these results were inconsistent. Agents were not always performing tasks according to their assigned roles, and running the same problem with the same prompting at different times led to different results—some attempts provided the right answer while other attempts did not even use the correct method.

The results show that while multi-agent collaboration was able to solve more difficult problems than one-shot and few-shot prompting, the latter methods provided consistency that was lacking in multi-agent collaboration. It can be concluded that multi-agent collaboration can assist LLMs in solving difficult math problems over multiple trials, while one-shot and few-shot prompting can consistently improve an LLM's mathematical reasoning.

ACKNOWLEDGMENTS.

Thank you to Dr. George Shakan for invaluable guidance on this paper.

REFERENCES

1. *Google DeepMind*. AI achieves silver-medal standard solving International Mathematical Olympiad problems. https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/. 25 July 2024.
2. D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, J. Steinhardt, Measuring Massive Multitask Language Understanding. arXiv:2009.03300v3 [cs.CY] (2021). [arXiv]
3. J. Achiam, S. Adler, et al., GPT-4 Technical Report. arXiv:2303.08774v6 [cs.CL] (2024). [arXiv]
4. T. B. Brown, B. Mann, et al., arXiv:2005.14165v4 [cs.CL] (2020). [arXiv]
5. D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, J. Steinhardt (2021). *Measuring Massive Multitask Language Understanding* [Software]. GitHub. https://github.com/hendrycks/test.
6. Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, S. Zhang, X. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, C. Wang (2024). *Microsoft AutoGen* [Software]. GitHub. https://github.com/microsoft/autogen.
7. XTX Investments. AI Mathematical Olympiad - Progress Prize 1. https://kaggle.com/competitions/ai-mathematical-olympiad-prize, 2024. Kaggle.
8. P. Liu, MMLU Dataset, https://www.kaggle.com/datasets/peiyuanliu2001/mmlu-dataset.

Saanvi C. Hublikar is a student at Mission San Jose High School in Fremont, California. She participated in a research internship through the College Impact.