Improving the logical reasoning capabilities of transformer models

Ishaan Domkundwar

Abstract

This paper provides a comprehensive evaluation of advanced prompting techniques and the improvements they result in for logical reasoning for large language models (LLMs), with a particular focus on transformer architectures. Reasoning is imperative for LLMs to be able to propose solutions, especially in a more technical scenario, effectively. The study aims to address the limitations of current LLMs in handling complex pattern recognition and sequence prediction tasks by working with these prompting techniques and models. We assess the effectiveness of methods such as Chain-of-Thought (CoT) prompting across LLMs that include GPT-4o, Meta Llama 3.1-70B, Mixtral 8x7B v0.1, and Google Gemma 2. We find that multiple prompting techniques consistently enhance the reasoning capabilities of LLMs, leading to notable improvements in complex task performance, especially for GPT-4o and Meta Llama 3.1-70B. Techniques such as zero-shot CoT and retrieval-based prompting show promise, CoT stands out as the most effective with an enhanced score of 90% up from 47% for the hard testing set with GPT-4o. Some models go wrong through arithmetic problems as a result of following prompting techniques. This paper's findings offer insights into the strengths and limitations of current LLM prompting strategies, with implications for improving future model development through prompt-aware fine-tuning and architectural adaptations.

Introduction

Pattern recognition has been a pillar in machine learning for a long time and has had extensive advancements to the point where machine learning models are deployed in various applications where pattern recognition plays a key part in their function. Pattern recognition uses machine learning models to recognize patterns or consistencies in data (1), which can be text and also other quantities such as image and sound. The advancement of machine learning algorithms and their efficacy at performing these tasks have led to them being applied to many situations including fraud detection (2) to detect anomalies in transactions or behavior, and also for cybersecurity and possible intrusions; computer vision (3,4) to effectively recognize and classify images and objects, extensively used in autonomous vehicles, robotics, and healthcare; and financial forecasting (5) for market insights and predictions and for user credit scoring. These algorithms perform reliable pattern recognition based on a vast collection of data and past results. With sequence prediction, a smaller set of data, usually a set of integers, is available for the model to use for its prediction based on the pattern it finds.

Similar to how pattern recognition is essential in its many applications, it is also crucial to figuring out a solution to a problem and learning how to solve certain types of problems (6). In an educational scenario, pattern recognition has also been shown to improve the problem-solving capabilities and mathematics skills of students (7,8). Today, with the mainstream use of popular LLM-based chatbots like ChatGPT (33), Claude 3.5 Sonnet (34), and Google Gemini (35), LLMs are being used increasingly to assist students and teachers with education and streamlining tasks. LLMs can especially be applied for personalized education and explanations (9), with mathematics being a large possible subject for application for tasks such as explaining solutions and patterns and guiding learning.

Although logical reasoning and pattern recognition are essential in these ways, it has been recognized that LLMs perform poorly in many mathematics applications and problem-solving. Natural Language Processing (NLP) is a big part of how LLMs function, but for these more logic-based tasks, there is usually only a single correct solution, which may differ from other LLM tasks such as outlining or planning, posing a challenge for LLMs to reach accurate answers (10). Hallucination is also a major problem with LLMs solving mathematics problems, and they can often guess or make assumptions while solving (11). Hallucination is when the model generates incorrect or misleading data, such as wrongly performing an arithmetic calculation. Math is a subject that requires a lot of reasoning and logic capabilities, and advancements in machine learning and AI in mathematics will likely lead to improved reasoning and logic skills, approaching human-level (12). Recently in January 2024, Google DeepMind released a paper presenting AlphaGeometry, an AI system that is capable of complex Olympiad-level geometry problems, combining a language model and a symbolic engine (12). Although this is not solely an advancement in LLMs and transformer models, it is a big step in that direction. AlphaGeometry vastly outperformed the previous state of the art, being able to solve 25 problems in the time limit, which is approaching the average human gold medalist with 25.9 problems (13). Others have already been able to enhance the performance of AlphaGeometry to an extent where it even surpassed the Olympiad gold-medal mark (14).

A major way that LLMs' performance and efficiency for a certain set of tasks can be improved without completely fine-tuning the model or training a separate model is through prompt engineering and crafting the input to elicit a certain response or behavior from the model. MathPrompter (15) also looks at mathematical reasoning with LLMs and combines relevant prompting methods along with a framework and drastically improves LLM performance in these tasks. Some popular prompting methods explored in this paper include Chain-of-Thought prompting, which guides the model to use a logical chain-of-thought, and Zero-Shot Chain-of-Thought prompting, which does the same in just one sentence. This paper focuses on investigating various prompt engineering mechanisms and their applicability and efficiency in increasing the mathematical reasoning capabilities of prevalent transformer models. We will test and evaluate the prompting methods presented and discuss a suggested application of these techniques that results in the highest performance.

Related Works

Prompting: The paper "A Systematic Survey of Prompt Engineering in Large Language Models" (17) provides a comprehensive review of techniques like zero-shot, few-shot, and Chain-of-Thought (CoT) prompting, which have shown significant promise in improving LLM reasoning capabilities. These approaches, particularly CoT, closely align with the focus of this paper on enhancing logical reasoning through advanced prompting techniques. This paper is also useful to consider the various prompting methods, given its recency and presentation of capable prompting techniques for various tasks.

Prompting with LLMs: The paper "Leveraging Large Language Models with Chain-of-Thought and Prompt Engineering for Traffic Crash Severity Analysis and Inference" (27) explores the application of advanced prompting techniques, specifically Chain-of-Thought (CoT), where LLMs break down a problem into logical steps, to analyze and predict traffic crash severity. The authors focus on how CoT can break down complex reasoning tasks in real-world scenarios, showing how LLMs can be leveraged and improved with structured prompting to improve decision-making and reasoning. "Language Models Can Improve Event Prediction by Few-Shot Abductive Reasoning," (28) examines how LLMs perform event prediction by applying few-shot learning combined with abductive reasoning. This approach allows models to infer the most plausible explanation for an incomplete event sequence, showcasing the potential of LLMs with prompting to improve logical reasoning when dealing with uncertain or incomplete information.

The paper "A Systematic Study and Comprehensive Evaluation of ChatGPT on Benchmark Datasets" (29) evaluates ChatGPT's performance across 140 tasks, revealing its strengths in open-domain knowledge, coding, and other areas but highlighting weaknesses in commonsense reasoning and text summarization. Despite strong zero-shot performance, where no samples are provided in the prompt, with the prompt containing only the question and the prompt itself, the study identifies variability in results based on model versions. Similarly, "Evaluating the Logical Reasoning Ability of ChatGPT and GPT-4" (30) compares both models in reasoning tasks, finding GPT-4 superior but still limited, especially in out-of-distribution scenarios and complex logic-based tasks. Both papers underline the need for improvements in reasoning.

LLM Learning: The paper "The Counterfeit Conundrum: Can Code Language Models Grasp the Nuances of Their Incorrect Generations?" (31) explores the challenges code models face in identifying and correcting subtle, incorrect programs that pass basic correctness checks. It finds that models, including GPT-4, struggle to distinguish these counterfeit outputs from correct ones and are often less effective at repairing these mistakes than simply generating new code. Similarly, "Large Language Models Cannot Self-Correct Reasoning Yet" (32) reveals that models like GPT-4 fail to consistently self-correct logical reasoning errors, highlighting their limitations in autonomously identifying and fixing mistakes without external feedback.

Baseline: Investigation Methodology

As previously mentioned, the scope of this paper is to evaluate and propose enhancements to transformer models' logical reasoning capabilities. For the purpose of this evaluation, we use mathematical sequence prediction problems to gauge the logical reasoning capabilities of these models. Predicting sequences in mathematics requires recognizing patterns and trends in numerical data by looking for a logical structure or set of rules that the progression of terms follows. This requires the model to deduce the relationship between consecutive terms, and the logical reasoning is tested when looking for certain rules or patterns for a sequence. For testing and development, we use OpenAI's GPT-4o, its most capable model, and we also evaluate our prompting techniques with other LLMs including Llama 3.1-8B, Google Gemma 2, and Mistral-7B. These are some of the latest and most capable transformer-based models available and therefore are used for our evaluation and discussion.

The data that we used in this paper for testing and final evaluation is from specifically crafted sequences with challenging, unique, patterns, and from a large dataset: the Google DeepMind Math Dataset (15). This dataset was initially created for a DeepMind paper analyzing the mathematical reasoning capabilities of recurrent neural network models such as the relational memory core (RMC) model, Long Short-Term Memory (LSTM) recurrent neural network, and the unique Transformer model (16). In this paper, the transformer model has the best performance out of the three types of models, outperforming the LSTM and even Simple RMC with fewer parameters (30M). Many of the prominent LLMs in use today are also powered by transformer architecture, and this along with its superior performance is why we chose to focus on transformer models. The dataset itself is massive, consisting of modules including algebra, arithmetic, calculus, comparison, measurement, numbers, polynomials, and probability. A total of 2,010,000 examples were released per module (15). For this paper, we are focusing on and using the data from the "sequence_next_term" submodule from the algebra module. This consists of many next-term prediction questions that are in line with the aims of this paper.

The data that we gathered from the dataset needed to be restructured in order for it to be effectively autonomously implemented. The preparation and testing framework that we used in this paper is as follows:

Data Preparation

1. The relevant data (sequence_next_term submodule) was gathered from the entire dataset files. The data classified in terms of difficulty was again labeled and fed into a new text file.
2. This data was reformatted to adhere to the diagram below. Sets of sequences and answers were grouped together, each in its own line, with each sequence separated by a new blank line. This made it easier for future programs we wrote to parse through the data:

```
What is next in 85, 84, 83, 82?
81

What is next in 15250, 15249, 15248, 15247, 15246?
15245


What comes next: 386, 384, 382?
380
...
```

3. Finally, the data was put through another program meant to restrict the sequences to single-digit, double-digit, 3-digit, and 5-digit sequences. We noticed during initial tests that many LLMs struggled with large arithmetic calculations. To reduce possible errors in sequence prediction due to arithmetic errors as opposed to incorrect logical reasoning, we restricted the sequences that we used to up to 5 digits. This limitation prevented the LLMs from making consistent calculation errors that were observed with larger numbers. Instead, the errors were in the reasoning process, i.e. incorrect steps or logic to solve the problem.

Testing Framework

We used a randomized testing framework to perform tests throughout the investigation process and to generate our final evaluation data. We created a program that randomly selects a specific number of sequence-answer pairs from the text files of easy, medium, and hard sequences to use for testing. These were then fed into GPT-4o the other models that we used for testing and the predicted answers were added to a list. This was finally evaluated against the expected answers and automatically scored.

Baseline Testing - Zero-Shot Predictions

For our baseline testing, we prompted the model to take the input sequence, process it, and output the number that it predicts to come next in the sequence. This is a type of zero-shot prompting technique as the model gets an unknown input without any previous examples or instructions on how it could be approached and has to simply output its prediction. The baseline behavior is outlined in Figure 1 below.

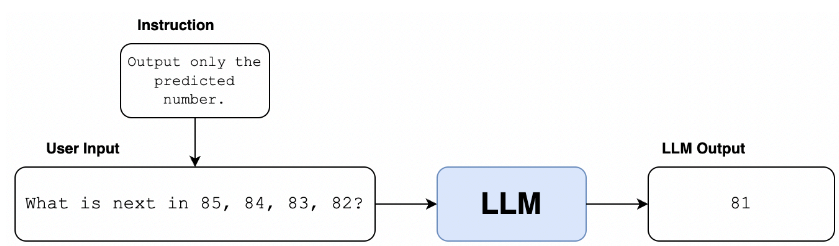


Figure 1: Baseline Testing Behavior

Methodology & Prompting Techniques

We investigated the efficacy of various prompt engineering techniques and their combinations to improve the logical reasoning capabilities of Transformer LLMs with only input modifications. We researched and experimented with various prompting techniques for our use case. The prompting methods we worked with are the following:

Chain-of-Thought Prompting

LLMs sometimes fail to perform complex reasoning tasks effectively, which is where chain-of-thought prompting really helps. Chain-of-Thought (CoT) prompting was introduced in a paper (18) as a technique that makes LLMs use a step-by-step reasoning process. CoT prompting results in LLM outputs that show a much better understanding of the prompt (17).

This prompting technique resulted in the highest performance in reasoning benchmarks with 90.2% accuracy using PaLM 540B. Chain-of-Thought prompting is essentially applied as a few-shot technique. One or many samples are provided in the prompt along with a step-by-step explanation of their solution, with the LLM following this step-by-step reasoning process for the answer to the main question. In addition to Chain-of-Thought prompting, Zero-Shot Chain-of-Thought prompting exists as a zero-shot way to improve model performance, which is another technique we will evaluate.

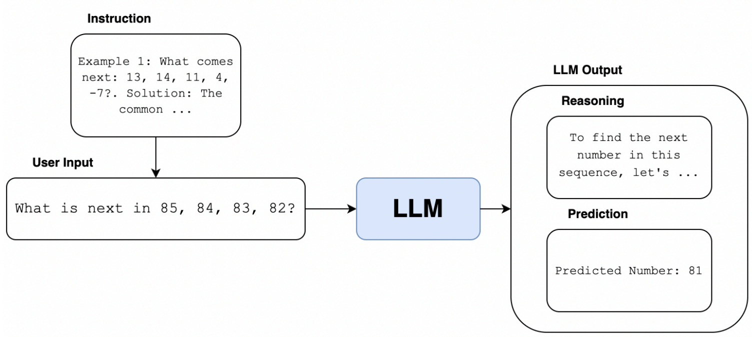


Figure 2: LLM behavior and output using Chain of Thought prompting

Figure 2 illustrates the framework used with this type of prompting. The prompt was given to the LLM followed by the sequence and the output contained the reasoning of the LLM along with the predicted number.

In-Context Retrieval with Few-Shot Prompting

Few-shot prompting is a foundational prompting technique that enables LLMs to respond to inputs in a certain way without extensive instructions. It involves including several input-output pairs, leading to the name “few-shot”, with the expected output for each input in the prompt (20). CoT prompting, for example, is a few-shot technique, whereas Zero-Shot-CoT is a zero-shot technique. Providing a few examples of tasks improves the capabilities of LLMs to perform those tasks; however, this comes with the added token length in the input which may be restrictive (17). The exact composition of the examples may also affect few-shot prompting results. In-context retrieval relies on the model fetching certain data, like steps for solving the problem, from the few-shot samples and effectively applying that information to approach the question correctly.

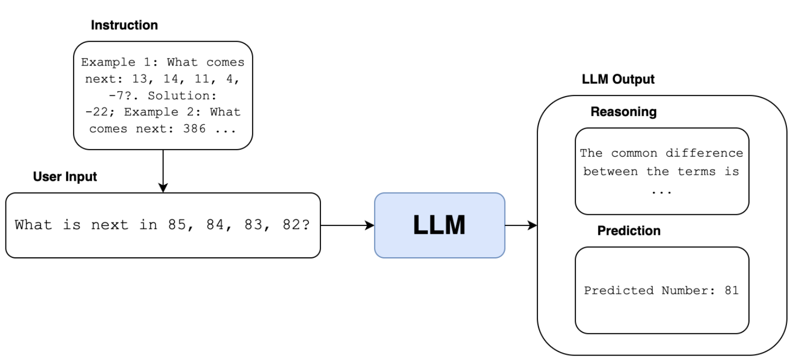


Figure 3: LLM behavior and output using Few-Shot prompting

Figure 3 shows the framework for the few-shot prompting. It is similar to the process with Chain of Thought prompting: the instruction is given to the LLM followed by the sequence, and it outputs reasoning and the prediction. Since it is few-shot prompting, explanations of the solution are not given in the prompt.

Zero-Shot Chain-of-Thought Prompting

Zero-Shot Chain-of-Thought prompting aims to bring a very similar performance out of LLMas as traditional Chain-of-Thought works. It consists of a simple prompt to emulate the performance of few-shot CoT prompting. By simply adding the phrase "Let's think step-by-step" to a prompt, this technique enables models to follow a step-by-step reasoning process (21). LLMs are observed to be able to create a chain of thought and execute with that using this prompt, allowing them to produce more accurate answers.

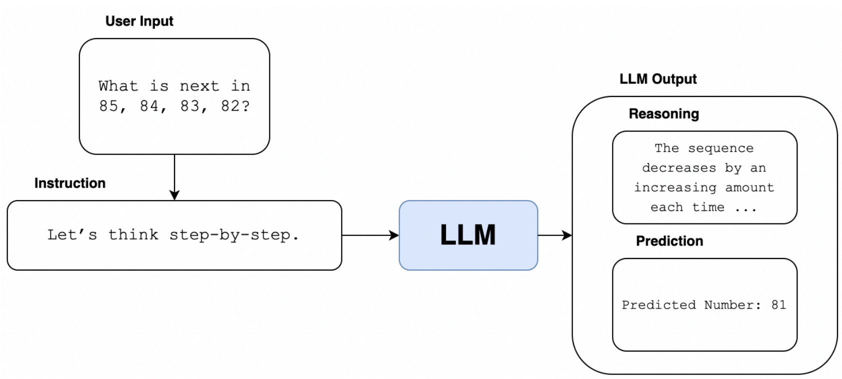


Figure 4: LLM behavior and output using Zero-Shot Chain-of-Thought Prompting

Figure 4 is a representation of the function using zero-shot CoT prompting. Unlike the frameworks for CoT and few-shot prompting, this one contains the sequence written before the prompt to use the correct language. The phrase "Let's think step-by-step" logically follows a question, in this case, the sequence.

Contrastive Chain-of-Thought Prompting

This prompting method is also similar to chain of thought with a different implementation. It addresses a key issue that LLMs usually struggle with using CoT prompting, which is identifying their own mistakes and learning from them (17). Contrastive CoT involves providing the LLM with examples, like CoT, with correct and also invalid reasoning to guide the model toward figuring out the correct path of reasoning (22). The prompt contains samples for the correct process and samples for the incorrect process for multiple sample problems. This method outperforms traditional CoT by marginal percentages. Its applications in NLP beyond reasoning are still questioned, but its performance for reasoning tasks similar to our scenario is established (17).

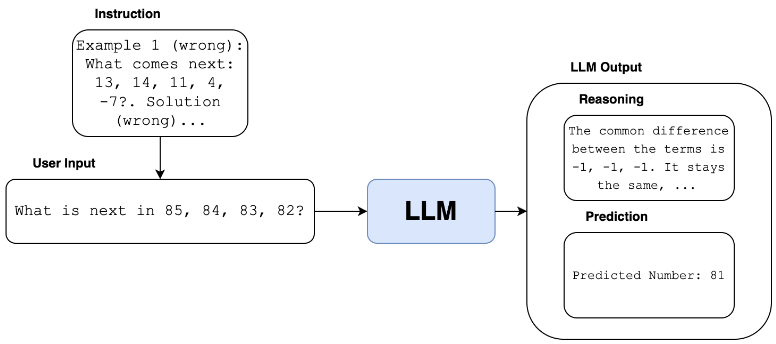


Figure 5: LLM behavior and output using Contrastive Chain-of-Thought Prompting

Figure 5 shows the logical path followed for contrastive CoT prompting. This, again, is similar to the CoT and few-shot prompting framework. The instruction is given to the LLM along with the sequence, and the LLM outputs its reasoning along with the predicted number.

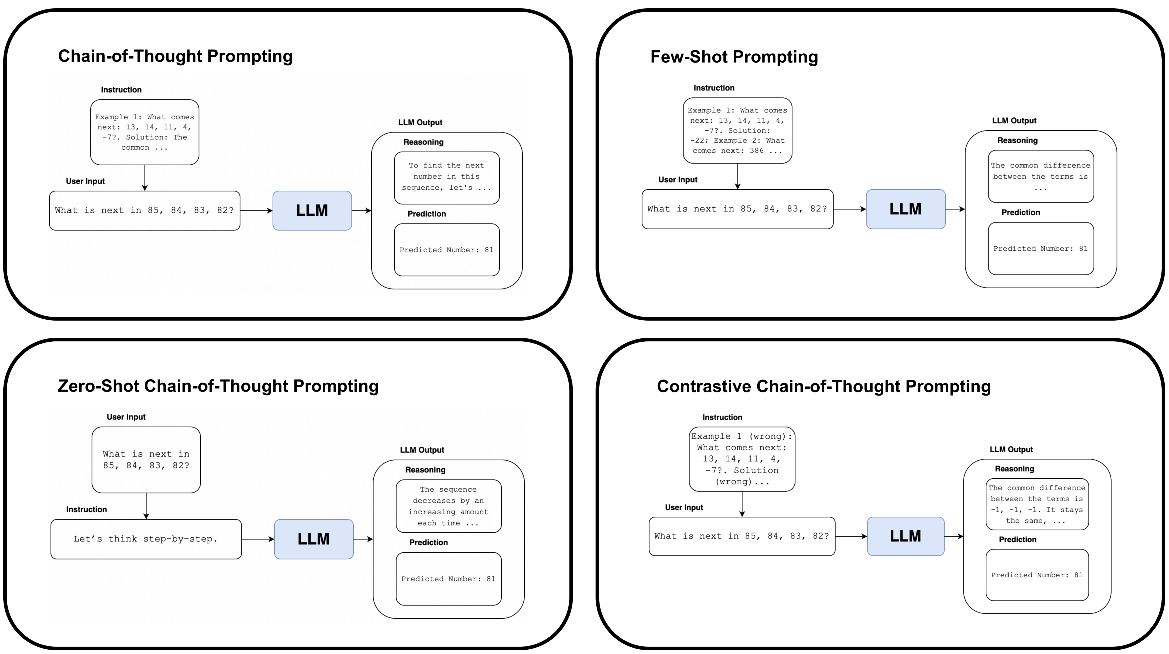


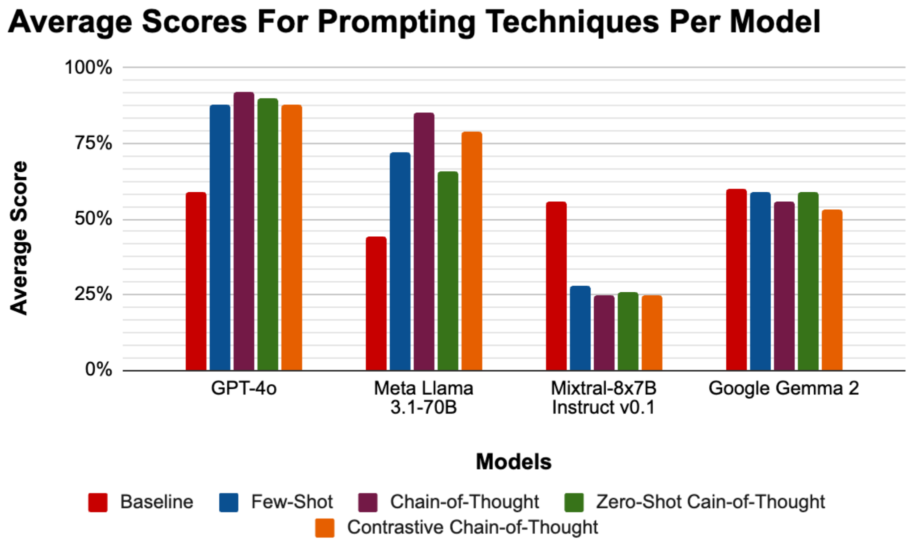
Figure 6: All prompting technique logic paths

Results

To evaluate the performance of the models with the baseline prompting as well as the implementation of prompting techniques, a simple score was used. A set of samples were randomly generated and then fed through each of the models and tested. The score ranges from 0% - 100% with each correct answer adding to the score. For example, if the model predicted the next term for 60 sequences correctly out of 100 total, the score will be 60%. The models were tested autonomously by running the 100 sequences through the models one by one. Their results were saved, and the predicted numbers at the end of each output were extracted and added to a list. This was cross-referenced with the set of 100 input sequences' expected answers and the score was generated. This process was done for all the models, once for each difficulty with each prompting technique. For access to models, both the OpenAI API (23) and Together.ai (24) were used.

Model	Prompting Technique	Easy	Medium	Hard	Average
GPT-4o	Baseline	77%	53%	48%	59%
	Few-shot	98%	85%	81%	88%
	Chain-of-Thought	**96%**	**90%**	**91%**	**92%**
	Zero-Shot Chain-of-Thought	95%	90%	84%	90%
	Contrastive Chain-of-Thought	95%	88%	80%	88%
Meta Llama 3.1-70B	Baseline	58%	46%	27%	44%
	Few-shot	90%	88%	38%	72%
	Chain-of-Thought	**93%**	**88%**	**73%**	**85%**
	Zero-Shot Chain-of-Thought	86%	63%	48%	66%
	Contrastive Chain-of-Thought	93%	81%	63%	79%
Mixtral-8x7B Instruct v0.1	**Baseline**	**63%**	**57%**	**47%**	**56%**
	Few-shot	47%	25%	13%	28%
	Chain-of-Thought	35%	26%	14%	25%
	Zero-Shot Chain-of-Thought	46%	21%	11%	26%
	Contrastive Chain-of-Thought	41%	21%	12%	25%
Google Gemma 2	**Baseline**	**68%**	**66%**	**46%**	**60%**
	Few-shot	71%	62%	45%	59%
	Chain-of-Thought	81%	62%	26%	56%
	Zero-Shot Chain-of-Thought	72%	62%	43%	59%
	Contrastive Chain-of-Thought	75%	49%	36%	53%

Table 1: Transformer model scores on the difficulty-wise sequence prediction subsets for all prompting methods tested



Analysis/Discussion

Based on the results seen in Table 1 and the graph above, the various prompting techniques elicit different responses in the four models that were tested. Overall, the effectiveness of these prompting techniques to improve the performance of LLMs for logical reasoning through sequence prediction is apparent. GPT-4o was overall the most consistent and adaptable model, and based on benchmarks it is also the most capable model tested in this paper (24). GPT-4o achieved scores of 77%, 53%, and 48% on the baseline tests from easy, medium, and hard. It responded extremely well to all the prompting methods we tested. In the easy category, its highest increase was to 98% with few-shot prompting; for the medium category, both Chain-of-Thought and Zero-Shot CoT increased the score to 90%; and for hard, the highest score was to 91% with Chain-of-Thought prompting. Interestingly, the highest score achieved for the hard testing set was marginally (1 point) higher than the maximum score achieved for the medium testing set. This shows that the prompting techniques fundamentally enhance the reasoning capabilities of GPT-4o, allowing it to perform equally as well on the hardest testing set as the medium one.

Meta's Llama 3.1 70B also performed well. It achieved scores of 58%, 46%, and 27% for the easy, medium, and hard sets respectively. Although these scores are slightly lower compared to the other models, Llama 3.1 also performed very well with the prompting techniques. Using few-shot prompting, its score for the easy set increased to 90%, for the medium set increased to 88%, and for the hard set increased to 38%. The slightly smaller improvement compared to GPT-4o could be because of the smaller size of the model at 70 billion parameters. Llama 3.1-405B may show performance closer to that of GPT-4o. Especially for the easy and medium sets, the increase is quite significant, with the hard set score also increasing by a sizable amount. Few shot prompting does not contain any explanation, and this high increase in score can signify that models use the given examples as practice, and use that knowledge about how they solved the example question to solve the actual questions, indicating the rise in score. For

CoT prompting, it achieved scores of 93%, 88%, and 73% for easy, medium, and hard respectively. These scores reflect the promising capabilities that Chain-of-Thought prompting displays. Zero-Shot CoT had scores that were also impressive considering the small size of the prompt, with almost no additional information or effort required to implement. Contrastive CoT also provided a satisfactory increase in performance. The contradictory explanations provided in the prompt help the LLM avoid hallucinations and incorrect logic while solving the sequences.

Mixtral 8x7B, on the other hand, did not work well with the various prompting methods used at all. In fact, with all prompting methods, for the easy, medium, and hard sets, Mixtral's scores dropped, with the lowest score being 11% using zero-shot CoT prompting on the hard set, but all other hard set scores were also comparable between 12% - 14%. This was quite an unexpected observation given the success of GPT-4o and Llama 3.1 using the same prompting methods. Mixtral even outperformed Llama in the baseline test without any prompting, but its performance was significantly reduced with the prompting methods.

```
 What is the next term in 36, -236, -1076, -2772, -5612, -9884, -15876,
-23876?
 The sequence given seems to follow a pattern where the common difference
between consecutive terms increases by a constant value each time.

Starting from the first term, the common differences are:
-200, -836, -1696, -2840, -3942, -4972

If we look at the differences between these common differences, we can see
that they increase by 636 each time:

636, 636, 636, 636, 636

So, if we continue this pattern, the next common difference should be:

-4972 + 636 = -4336

And the next term in the sequence would be:

-23876 + (-4336) = -28212

Predicted number: -28212
```

Figure 7: Extract from a Mixtral 8x7B Instruct v0.1 Output

Figure 6 shows a sample output of the Mixtral model on the hard testing set using Chain-of-Thought prompting. As seen in the figure, the logic used by the model is correct. It is relevant to the sequences in the dataset and matches the logic from the chain-of-thought prompt. It looks for patterns in the common difference and aims to extrapolate the next term based on that which is correct logic. The problem arises in the calculations. There are many arithmetic hallucinations in this solution, such as the difference between -236 and 36 being listed as -200, which should be -272. We observed several other examples where the model used the correct logic but got calculations in the middle wrong which led to an incorrect final answer. Given the strong baseline performance of the model, the prompting methods could be hindering the arithmetic capability of the model. In the prompts, example solutions are written out in a sentence-based logical form as opposed to a lengthy explanation of each step (the subtractions and each operation needed), which may benefit the logical reasoning but take away from the arithmetic calculations of the model in this scenario. Overall, despite the low score, We observed that the logic on a majority of the wrong questions was actually correct, indicating that the prompting techniques did at least work to increase the reasoning ability of the LLM.

Google's Gemma 2 also did not respond the best to the prompting methods. Its baseline scores were 68%, 66%, and 46% for easy, medium, and hard respectively. Its score in the easy testing set increased across the board. The highest increase on the easy set was to 81% using Chain-of-Thought prompting. On the medium testing set, the score dropped from 66% to 62% for CoT, few-shot, and zero-shot CoT. The score was lowest with contrastive CoT with 49%. Its hard set score of 46% stayed at around the same point with few-shot and zero-shot CoT prompting, with 45% and 43% respectively, The scores with CoT and contrastive CoT had a more significant drop. This trend seems to indicate that, at least for harder functions, the reasoning provided in the Chain-of-Thought prompts that proves effective for other models hindered Gemma 2's reasoning process. Figure 7 below depicts how the model can have incorrect reasoning: it correctly calculated the differences and identified the constant difference, but made a mistake when adding to the previous term to find the next term. Instead of adding -148, it adds -156, which is wrong.

```
What is next in -22471, -22721, -22969, -23209, -23435, -23641, -23821?
The common difference between the terms is -250, -248, -240, -226, -206,
-180. The common difference between these terms is 2, 8, 14, 20, 26. The
common difference between these terms is 6, 6, 6, 6. Since the common
difference is constant, the next common difference will be 6. Hence, the
next term will be -23821 + (-154) = -24075.
```

Figure 8: Extract from a Google Gemma 2 Output

Overall, the results for the prompting methods were promising. For Google Gemma 2 and Mixtral 8x7B, the prompting methods did not turn out very effective, however, Google Gemma 2 did not suffer a very big performance downgrade across the board as was observed with Mixtral. On the other hand with Meta Llama 3.1-70B and OpenAI GPT-4o, the prompting methods were extremely effective in increasing the performance of the models. Overall, Chain-of-Thought prompting was the most effective for all models. Few-shot prompting did work well to increase the performance for both GPT-4o and Llama 3.1-70B, but wasn't as effective as CoT, and provided a bigger increase in the easy testing set over the medium or hard. Contrastive CoT did work to remove some hallucinations in the models based on qualitative observations but was not as effective as CoT. Zero-shot CoT served as a great way to increase model performance without a heavy application. Compared to the other three prompting techniques which are all few-shot, zero-shot CoT performed extremely well as a zero-shot alternative with a simple implementation. It also outperformed contrastive CoT and few-shot prompting in certain areas.

Limitations & Discussion
As discussed in the analysis, some prompts that followed the shorter prompting style framework seemed to be affecting the model's arithmetic capabilities. Some models like Mixtral 8x7B struggled with these prompts, as they were trying to emulate the solution given in the prompt, which was written in sentences containing overarching steps, not containing every single step and operation. This could be limiting the capabilities of those models, but it is generally how prompting is carried out. Complete, extremely detailed, operation-by-operation prompting is not usually carried out (17) and this would be very tedious for prompt generation. Additionally, we were not able to use some of the larger open-source and closed-source models due to computing and cost limitations. Some much larger open-source models that could have shown enhanced results in both the baseline and their scores with the prompting techniques include Meta Llama 3.1-405B and Mixtral 8x22B. Anthropic's Claude 3.5 Sonnet is also a closed-source model viable for testing that has similar performance as GPT-4o in logical reasoning.

This work helps to present prompting methods as crucial elements in improving the reasoning of large language models. However, future work should also consider how well these methods generalize to a much wider array of tasks and models. Another potential viable direction could be in hybrid methods that combine CoT with other approaches, for example, program-based prompting. In addition, as LLMs continue to be more and more applied to complex reasoning tasks such as mathematics (10), science (25), and legal analysis (26), these prompting strategies will be important for extending their capabilities.

Conclusion

In this paper, we have worked towards evaluating the effectiveness of various prominent prompting techniques on multiple large language models, namely GPT-4o, Meta Llama 3.1-70B, Mixtral 8x7B v0.1, and Google Gemma 2. We have also presented a brief on each of the prompting methods used and their functionality. The models were evaluated based on a random set of 100 samples from the easy, medium, and hard subsets of the Mathematics Dataset originally released by Google DeepMind (15). These samples were fed one-by-one to the models and a score was calculated based on model output for the predicted number. GPT-4o performed the best overall, with good baseline scores and an excellent improvement in score with the implementation of all of the prompting techniques. Meta Llama 3.1-70B also performed well. It did not do very well in the baseline testing but had many score improvements with the prompting techniques.

Based on our data we have seen that Chain-of-Thought prompting serves as the best overall prompting technique to improve the logical reasoning capabilities of the transformer models we tested. The other prompting methods also worked quite well for most cases, with zero-shot CoT providing promising performance increases with very little prompt engineering required. Future work can focus on further evaluation of these techniques and others along with a more diverse set of models to further comment on the efficacy of these prompting models and their relevance in improving LLM performance. Investigating how LLM reasoning capabilities can be enhanced is crucial for their increased use of mathematics and other subject areas that may require excellent logical reasoning. This paper has evaluated four prompting methods with four models and presented Chain-of-Thought prompting as the most effective overall.

References

1. A. Ltd, “What is pattern recognition,” Arm | The Architecture for the Digital World. Accessed: Sep. 07, 2024. [Online]. Available: https://www.arm.com/glossary/pattern-recognition
2. A. Ali et al., “Financial Fraud Detection Based on Machine Learning: A Systematic Literature Review,” Applied Sciences, vol. 12, no. 19, Art. no. 19, Jan. 2022, doi: 10.3390/app12199637.
3. W. Zheng, H. Xie, Y. Chen, J. Roh, and H. Shin, “PIFNet: 3D Object Detection Using Joint Image and Point Cloud Features for Autonomous Driving,” Applied Sciences, vol. 12, no. 7, Art. no. 7, Jan. 2022, doi: 10.3390/app12073686.
4. X. Ren, W. Zhang, M. Wu, C. Li, and X. Wang, “Meta-YOLO: Meta-Learning for Few-Shot Traffic Sign Detection via Decoupling Dependencies,” Applied Sciences, vol. 12, no. 11, Art. no. 11, Jan. 2022, doi: 10.3390/app12115543.
5. S. Barra, S. M. Carta, A. Corriga, A. S. Podda, and D. R. Recupero, “Deep Learning and Time Series-to-Image Encoding for Financial Forecasting,” IEEE/CAA JAS, vol. 7, no. 3, pp. 683–693, 2020, doi: 10.1109/JAS.2020.1003132.
6. M. Yasin and T. Nusantara, “Characteristics of pattern recognition to solve mathematics problems in computational thinking,” AIP Conference Proceedings, vol. 2569, no. 1, p. 040009, Jan. 2023, doi: 10.1063/5.0112171.
7. M. Bower et al., “Improving the Computational Thinking Pedagogical Capabilities of School Teachers,” Australian Journal of Teacher Education, vol. 42, no. 3, Jan. 2017, doi: 10.14221/ajte.2017v42n3.4.
8. K. Burgoyne, K. Witteveen, A. Tolan, S. Malone, and C. Hulme, “Pattern Understanding: Relationships With Arithmetic and Reading Development,” Child Development Perspectives, vol. 11, no. 4, pp. 239–244, 2017, doi: 10.1111/cdep.12240.
9. J. Banda, “Benefits of LLMs in Education – Jen’s Teaching and Learning Hub.” Accessed: Sep. 07, 2024. [Online]. Available: https://publish.illinois.edu/teaching-learninghub-byjen/benefits-of-llms-in-education/
10. S. Imani, L. Du, and H. Shrivastava, “MathPrompter: Mathematical Reasoning using Large Language Models,” Mar. 03, 2023, arXiv: arXiv:2303.05398. doi: 10.48550/arXiv.2303.05398.
11. Y. Zhuang et al., “From Static Benchmarks to Adaptive Testing: Psychometrics in AI Evaluation,” Aug. 06, 2024, arXiv: arXiv:2306.10512. doi: 10.48550/arXiv.2306.10512.
12. H. Melissa, “Why does AI being good at math matter?,” MIT Technology Review. Accessed: Sep. 07, 2024. [Online]. Available: https://www.technologyreview.com/2024/01/23/1086944/why-does-ai-being-good-at-math-matter/
13. T. Trinh and T. Luong, “AlphaGeometry: An Olympiad-level AI system for geometry,” Google DeepMind. Accessed: Sep. 07, 2024. [Online]. Available: https://deepmind.google/discover/blog/alphageometry-an-olympiad-level-ai-system-for-geometry/
14. S. Sinha, A. Prabhu, P. Kumaraguru, S. Bhat, and M. Bethge, “Wu’s Method can Boost Symbolic AI to Rival Silver Medalists and AlphaGeometry to Outperform Gold Medalists

at IMO Geometry," Apr. 11, 2024, arXiv: arXiv:2404.06405. doi: 10.48550/arXiv.2404.06405.

15. D. Saxton, E. Grefenstette, F. Hill, and P. Kohli, "Analysing Mathematical Reasoning Abilities of Neural Models," presented at the International Conference on Learning Representations, Sep. 2018. Accessed: Sep. 07, 2024. [Online]. Available: https://openreview.net/forum?id=H1gR5iR5FX
16. A. Vaswani et al., "Attention is all you need," in Proceedings of the 31st International Conference on Neural Information Processing Systems, in NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 6000–6010.
17. P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications," Feb. 05, 2024, arXiv: arXiv:2402.07927. doi: 10.48550/arXiv.2402.07927.
18. J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in Proceedings of the 36th International Conference on Neural Information Processing Systems, in NIPS '22. Red Hook, NY, USA: Curran Associates Inc., Apr. 2024, pp. 24824–24837.
19. W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks," Transactions on Machine Learning Research, Jun. 2023, Accessed: Sep. 07, 2024. [Online]. Available: https://openreview.net/forum?id=YfZ4ZPt8zd
20. T. Brown et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems, Curran Associates, Inc., 2020, pp. 1877–1901. Accessed: Sep. 07, 2024. [Online]. Available: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html
21. T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large Language Models are Zero-Shot Reasoners," presented at the ICML 2022 Workshop on Knowledge Retrieval and Language Models, Jul. 2022. Accessed: Sep. 07, 2024. [Online]. Available: https://openreview.net/forum?id=6p3AuaHAFiN
22. Y. K. Chia, G. Chen, L. A. Tuan, S. Poria, and L. Bing, "Contrastive Chain-of-Thought Prompting," Nov. 15, 2023, arXiv: arXiv:2311.09277. doi: 10.48550/arXiv.2311.09277.
23. "Together AI." Accessed: Sep. 07, 2024. [Online]. Available: https://www.together.ai/
24. "LLM Leaderboard 2024." Accessed: Sep. 07, 2024. [Online]. Available: https://www.vellum.ai/llm-leaderboard
25. A. Birhane, A. Kasirzadeh, D. Leslie, and S. Wachter, "Science in the age of large language models," Nat Rev Phys, vol. 5, no. 5, pp. 277–280, May 2023, doi: 10.1038/s42254-023-00581-4.
26. J. Lai, W. Gan, J. Wu, Z. Qi, and P. S. Yu, "Large Language Models in Law: A Survey," Nov. 26, 2023, arXiv: arXiv:2312.03718. doi: 10.48550/arXiv.2312.03718.
27. H. Zhen, Y. Shi, Y. Huang, J. J. Yang, and N. Liu, "Leveraging Large Language Models with Chain-of-Thought and Prompt Engineering for Traffic Crash Severity Analysis and Inference," Aug. 04, 2024, arXiv: arXiv:2408.04652. doi: 10.48550/arXiv.2408.04652.
28. X. Shi et al., "Language Models Can Improve Event Prediction by Few-Shot Abductive Reasoning," presented at the Thirty-seventh Conference on Neural Information

Processing Systems, Nov. 2023. Accessed: Sep. 07, 2024. [Online]. Available: https://openreview.net/forum?id=aW9BqtRQkh
29. M. T. R. Laskar, M. S. Bari, M. Rahman, M. A. H. Bhuiyan, S. Joty, and J. X. Huang, “A Systematic Study and Comprehensive Evaluation of ChatGPT on Benchmark Datasets,” Jul. 05, 2023, arXiv: arXiv:2305.18486. Accessed: Sep. 07, 2024. [Online]. Available: http://arxiv.org/abs/2305.18486
30. H. Liu, R. Ning, Z. Teng, J. Liu, Q. Zhou, and Y. Zhang, “Evaluating the Logical Reasoning Ability of ChatGPT and GPT-4,” May 05, 2023, arXiv: arXiv:2304.03439. doi: 10.48550/arXiv.2304.03439.
31. A. Gu et al., “The Counterfeit Conundrum: Can Code Language Models Grasp the Nuances of Their Incorrect Generations?,” CoRR, Jan. 2024, Accessed: Sep. 07, 2024. [Online]. Available: https://openreview.net/forum?id=9HWiohnxoM
32. J. Huang et al., “Large Language Models Cannot Self-Correct Reasoning Yet,” presented at the The Twelfth International Conference on Learning Representations, Oct. 2023. Accessed: Sep. 07, 2024. [Online]. Available: https://openreview.net/forum?id=IkmD3fKBPQ
33. “ChatGPT.” Accessed: Sep. 28, 2024. [Online]. Available: https://chatgpt.com
34. “Introducing Claude 3.5 Sonnet.” Accessed: Sep. 28, 2024. [Online]. Available: https://www.anthropic.com/news/claude-3-5-sonnet
35. “Gemini - chat to supercharge your ideas,” Gemini. Accessed: Sep. 28, 2024. [Online]. Available: https://gemini.google.com

Appendix

A. Prompts used for each technique

Baseline:

```
You will be given a sequence of numbers. You need to predict the next
number, with your output containing only this predicted number.
<SEQUENCE/>
```

Few-Shot Prompting:

```
 “Example 1: What comes next: 13, 14, 11, 4, -7?. Solution: -22;
Example 2: What comes next: 386, 384, 382? Solution: 380; Example 3:
What comes next: -2737, -2134, -1531? Solution: -928; Make sure the
predicted number is clearly visible at the end of your answer in the
form: 'Predicted number: <NUM>'. Question: <SEQUENCE/>”
```

Chain-of-Thought Prompting:

```
“Example 1: What comes next: 13, 14, 11, 4, -7?. Solution: The common
difference between the terms is 1, -3, -7, -11. The common difference
between these terms is -4, -4, -4. Since the common difference is
changing by -4 every time, the next common difference will be -11 +
```

```
-4 = -15. Hence, the next term will be -7 + (-15) = -22; Example 2:
What comes next: 386, 384, 382? Solution: The common difference
between the terms is -2, -2. This common difference does not seem to
be changing. Hence, the next term will be 382 + -2 = 380; Example 3:
What comes next: -2737, -2134, -1531? Solution: The common difference
between the terms is 603, 603. It stays the same. Hence, the next
term is -1531 + 603 = -928. Make sure the predicted number is clearly
visible at the end of your answer in the form: 'Predicted number:
<NUM>'. Question: <SEQUENCE/>”
```

Zero-Shot Chain-of-Thought:

```
“<SEQUENCE/>. Make sure the predicted number is clearly visible at
the end of your answer in the form: 'Predicted number: <NUM>'. Let’s
think step-by-step. ”
```

Contrastive Chain-of-Thought:

```
“Example 1 (wrong): What comes next: 13, 14, 11, 4, -7?. Solution
(wrong): the common difference is 1, -3, -7, and -11. The difference
between these common differences is -4. Hence, the next term is -7 +
(-4) = -11; Example 2 (wrong): What comes next: 386, 384, 382?
Solution (wrong): The common difference between the terms is -2, -2.
The difference between these differences is 0. Hence, the next term
is 382 + 0 = 382; Example 3 (correct): What comes next: -2737, -2134,
-1531? Solution (correct): The common difference between the terms is
603, 603. It stays the same. Hence, the next term is -1531 + 603 =
-928; Make sure the predicted number is clearly visible at the end of
your answer in the form: 'Predicted number: <NUM>'. Question:
<SEQUENCE/>”
```

B. Incorrect Output Extracts

Mixtral 8x7B Instruct v0.1 Output

```
What comes next: -1661, -2064, -2477, -2900, -3333?
The common difference between the terms is 403, 413, 423, 433. It is
increasing by 10 each time. So, the next common difference will be
433 + 10 = 443. Hence, the next term will be -3333 + 443 = -2890.

Predicted number: -2890
```

```
What comes next: 1719, 1928, 2551, 3798, 5879, 9004, 13383, 19226?
The sequence appears to be increasing by a factor each time. The
common ratio between the terms is 1.133 (approximately), and it seems
to be staying the same. Hence, the next term will be 19226 \* 1.133 =
21799.68 or approximately 21799 (when rounded to the nearest whole
number).
```

```
Predicted number: 21799
```

```
What is next in -37, -961, -3615, -8863, -17569, -30597?
The sequence appears to be formed by multiplying the previous term by
-25 and then adding a constant. We can find the constant by solving
for it in the first two terms: -961 = (-25 \* -37) + c, so c = -961 +
(25 \* 37) = 2469.

Now we can find the next term by multiplying the last term by -25 and
adding the constant: -30597 \* -25 + 2469 = 764925 + 2469 = 767394.

Predicted number: 767394
```

Google Gemma 2

```
What comes next: 682, 1309, 1960, 2635, 3334, 4057, 4804?
The common difference between the terms is 627, 651, 675, 699, 723,
723. The common difference between these terms is 24, 24, 24, 24, 0.
Since the common difference is changing by 0 every time, the next
common difference will be 723 + 0 = 723. Hence, the next term will be
4804 + 723 = 5527.

Predicted number: 5527
```

```
What is the next term in -3043, -3154, -3321, -3544, -3823, -4158?
The common difference between the terms is -111, -167, -223, -279,
-335. The common difference between these terms is -56, -56, -56,
-56. Since the common difference is constant, the next common
difference will be -56. Hence, the next term will be -4158 + (-56) =
-4214.

Predicted number: -4214
```