Analysis of Prompt Rephrasing for LLMs

Brandon Wu, Enze Ma, Quynh Le, Yongce Li
University of California, San Diego
`bmwu,e1ma,q3le,yol013@ucsd.edu`

Abstract

We evaluate the performance of ChatGPT3.5 in different types of tasks. After experimenting with five different datasets including code comprehension, knowledge extraction, contest math questions, and broad knowledge multiple choices questions, we find that ChatGPT3.5 exhibit proficiency in knowledge based questions and code related questions, but struggle with math, complex reasoning, and arithmatic tasks. For each dataset, we propose different approaches to increase and decrease the model performance by slightly changing the prompts without modifying their meaning. Additionally, we analyze the weakness of general Chain-of-Thoughts-based prompting methods, particularly their struggle with long-term memory retention, and lay out potential improvements.

1 Introduction

Due to its adaptability, LLMs become more and more popular in problem solving. The increasing popularity also emphasized the importance of utilizing it effectively. To achieve better results when using LLMs, it is crucial to understand the intricacies of how prompts are formulated and structured. LLMs are known to be particularly sensitive to prompts, and identifying patterns and preferences in the way prompts are posed can lead to improved performance and more accurate responses. By analyzing the patterns and preferences that LLMs exhibit, we can gain a deeper understanding of its underlying mechanisms. Ultimately, this will allow us to leverage the full potential of this powerful tool, and maximize its utility in a range of applications and contexts.

In this report, we evaluate the performance of the ChatGPT 3.5 Turbo model across five distinct datasets including code comprehension, knowledge extraction, contest math problems, and broad knowledge-based multiple-choice questions. Our analysis shows that the GPT model performs well in responding to knowledge-based and code-related questions, with a baseline accuracy that varies between 60% and 70%. However, the model encounters challenges when solving tasks requiring mathematics, complex reasoning, and arithmetic, with a lower baseline accuracy range of 20% to 30%. We have also proposed a range of unique approaches to improve the model's ability by modifying the prompts without changing their original meaning. We discovered that by integrating relevant information into the original questions, we could improve the model's overall performance by 3% to 15%. Conversely, adding irrelevant or confusing terms had a bad impact on accuracy. For mathematical queries, we identified three methods to enhance model performance: refining the problem's description, correcting the reasoning steps, and ensuring the calculation steps are accurate. The implementation of them resulted in a significant increase in mathematical problem-solving accuracy by 20.56%. Moreover, we manually inspected errors in the model's Chain-of-Thoughts process, concluding that long or complex steps often led to a drop in accuracy due to the model losing track of its previous outputs. Future works include adding working memory system to CoT prompting to mitigate such issue. In case you are interested in our dataset and code. Data and Code

2 Related Works

(Liu et al., 2021) investigated the sensitivity of GPT-3 to in-context examples. (Lu et al., 2022) showed that few-shot prompts suffer from order sensitivity and proposed efficient prompt ordering for text classification. (Webson and Pavlick, 2022) found that Prompt-based models often learn equally fast with misleading and irrelevant templates as they do with instructive ones. (Wei et al., 2023) showed that by adding reasoning steps to few-shot prompt, LLM can perform better on reasoning task than task-specific fine-tuned models. (Kojima et al.,

2023) showed that besides exemplars, reasoning instruction also matters in prompt.

3 Experiments

3.1 Model and Datasets

We used ChatGPT 3.5 turbo to experiment different prompting patterns on 5 datasets: BoolQ (Clark et al., 2019), CodeQA (Liu and Wan, 2021), MATH (Hendrycks et al., 2021b), Mathematics dataset (Saxton et al., 2019), Multiple choices questions (Hendrycks et al., 2021a).

3.2 BoolQ

BoolQ is a question answering dataset for yes/no questions containing 15942 examples. Each example is a triplet of (question, passage, answer), with the title of the page as optional additional context.

In this section, we ran these 5 experiments to evaluate the performance of the model using random 300 samples of the BoolQ dataset.

Using original question without additional information In this experiment, we simply use the question from the 'question' column in the dataset to ask the model and add "True or False, give one word answer" to the end of the prompt. See Figure 1 for examples.

Using Chain-of-Thought (Let's think step by step) In this experiment, after asking the question, we add the chain-of-thought prompt "Let's think step by step" after asking the question. See Figure 1 for examples. This does not improve or hurt the model accuracy.

Using original question with additional information In this experiment, we add the information provided in the 'passage' column to the prompt after the question prompt. See Figure 1 for examples. This is proven to boost the performance significantly, achieving 85% accuracy when being tested with 300 samples.

Adding perturbation word 'unknown' to the front of the prompt After running some preliminary exploration, we observed that prompts containing the word 'unknown' confused the model. In this experiment, we add 'unknown' to the beginning of the prompt, before the question to evaluate how the model gets affected. See Figure 1 for examples. This does not negatively impact the model's performance.

Adding perturbation word 'unknown' to the back of the prompt In this experiment, we repeat the previous one but we add the word 'unknown' after the question. See Figure 1 for examples. This hurts the model's performance the greatest.

Swapping "True" "False" order in the prompt In this last experiment, we swap the order of "True" and "False" in the prompt; instead of asking "True or False", we use "False or True". This does not impact the model.

```
original input

is a wolverine the same as a badger. True or False, give one word answer

with 'passage'

Badgers are short-legged omnivores in the family Mustelidae, which also includes
the otters, polecats, weasels, and wolverines. They belong to the caniform
suborder of carnivoran mammals. The 11 species of badgers are grouped in three
subfamilies: Melinae (Eurasian badgers), Mellivorinae (the honey badger or
ratel), and Taxideinae (the American badger). The Asiatic stink badgers of the
genus Mydaus were formerly included within Melinae (and thus Mustelidae), but
recent genetic evidence indicates these are actually members of the skunk
family, placing them in the taxonomic family Mephitidae. is a wolverine the same
as a badger. True or False, give one word answer

add unknow front

unknown is a wolverine the same as a badger. True or False, give one word answer

add unknow back

is a wolverine the same as a badger unknown. True or False, give one word answer

swap true false

is a wolverine the same as a badger. False or True, give one word answer

chain of thoughts

is a wolverine the same as a badger. Let's think step by step. True or False,
give one word answer
```

Figure 1: Examples of experiments using BoolQ dataset

Method	Accuracy
Using 'question' only	71%
CoT	71%
Using 'passage'	**85%**
adding 'unknown' front	72%
adding 'unknown' back	55%
swapping true false	73%

Table 1: Accuracy on BoolQ dataset

3.3 CodeQA

The CodeQA dataset consists of 7k questions, most of which have short answer responses. Each question is based on an accompanied python code snippet. These snippets have been cleaned so there are no new lines and most of the special characters have been removed.

Some questions in the dataset were difficult to interpret either because it was too discipline specific or just cryptic in general. This made it hard to manually evaluate model output, not to mention automatically. In order to combat the discrepancy in understanding as well as have an easier time evaluating model output, we elected to experiment only on the 201 yes/no questions. There were 101 yes questions and 100 no questions. Several prompting methods were testing, each with varying levels of success.

Yes or no First, we simply concatenated the code snippet to the question. Then, in order to get the yes or no in the answer for automatic evaluation, we concatenated "Please answer with a yes or no." to the end of the model input. This format is used in the following methods, with the concatenation of question and code snippet at the beginning, and yes or no prompt at the end.

Zero shot We tested adding "Let's think step by step." to the prompt, directly before the yes or no prompt.

Few shot While analyzing the incorrect responses the model had of the baseline prompting method, we realized that there were many instances in which the model had reasonable responses that were marked as incorrect. Using some of these, we formulated some few shot examples. We also added chain of thought by adding in the phrase "Explain your reasoning." These were added to the model input before the query.

> Does the operating system support symlinks ? def stub islink path return False Explain your reasoning. Please answer with a yes or no: The function returns false so therefore the operating system does not support symlinks. No.

> Does the code return the number of attracting components in g ? @not implemented for 'undirected' def number attracting components G n len list attracting components G return n Explain your reasoning. Please answer with a yes or no: The code gets the attracting components of G and then converts that to a list. It then uses len to find the length and returns it. Therefore it does return the number of attracting components in g. Yes.

> Are orgs considered within a microsite ? def get all orgs return BACKEND get all orgs Explain your reasoning. Please answer with a yes or no: The code calls BACKEND get all orgs which returns all orgs. Therefore orgs are considered within a microsite. Yes.

Code format Python is a language that requires particular spacing, which means the text cleaning of the code had especially dire consequences to the readability and interpretability of the code snippets. As such, we hypothesized that proper formatting could lead to higher accuracy of the model output. We thought that ChatGPT might be able to format said code well enough. We first asked the model "Please properly format the following python code." and concatenated the code snippet to the end. Once getting that response, we feed it back into ChatGPT as chat history and concatenate the question with the same "Please answer with a yes or no."

Code conversion With a similar train of thought, we hypothesized that perhaps converting the python code into a language that did not require particular spacing, such as C or Java, might perform better. However, we realized this approach did not work as the model would say that it was impossible and that it could only deal with python, or it would add random portions into strings, which was clearly incorrect.

Repeating and confirmation Another method we tried was repeating questions as well as asking the model followup "Are you sure?" questions. While it seemed promising at first, as it seemed like the model was able to correct its mistakes, we quickly realized that it actually was not learning at all. Instead, it was just taking the opposite stance and answer as its previous response. Continuing this process had the model flipping stances back and forth until finally saying it did not have enough information to answer the question.

Method	Accuracy	F1
Baseline (Yes/No)	**64.1%**	**0.709**
Zero Shot	56.7%	0.638
Few Shot	62.6%	0.660
Code Format	58.2%	0.552

Table 2: Accuracy on CodeQA dataset

3.4 Multiple Choices

This section focuses on the utilization of a dataset (Hendrycks et al., 2021a) consisting of a wide-ranging collection of questions, spanning thousands in number and classified into 57 distinct domains. Unlike previous sections that focused on specific domains, this particular section adopts a broader perspective regarding the performance of ChatGPT. To assess the influence of different prompts on the accuracy of ChatGPT's responses, we conducted a series of trials. These trials aimed to investigate the varying effects of prompt formulation on the performance of ChatGPT.

In Trial 1, we employed the prompt format of "question + choices + provide only the alphabetic representation without additional explanations." This prompted ChatGPT to deliver direct answers and restrict step-by-step explanations.

In Trial 2, we utilized the prompt "question + choices + explain step by step, then provide the alphabetic representation." Here, we allowed ChatGPT to explain the process of solving the question in a stepwise manner before presenting the answer in alphabetic form.

In Trial 3, the prompt was modified to "The following questions will be about" + area + ". Now, solve" + question + choices. By providing contextual information about the area of the questions, we aimed to assist ChatGPT in generating more accurate responses.

Trial 4 involved the prompt "Check the following question" + example + ". Now, solve" + question + choices. We aimed to employ a one-shot prompt strategy by presenting a random example within the same domain, intending to aid ChatGPT in answering the following questions.

In Trial 5, we endeavored to investigate the impact of providing related examples. We conducted two variations: one correctly solved with accurate calculations and one where the calculations were intentionally incorrect. Due to limitations in available resources, we could only present a few illustrative examples, precluding an accuracy evaluation but yielding interesting insights.

Recognizing the considerable human effort required for Trials 3-5, we devised an alternative approach in Trial 6. Initially, we posed the question to ChatGPT and recorded its response. Subsequently, we inquired if there were any issues with the previous answer before presenting the same question again. The modified prompt followed the format "Solve" + question + choices + previous answer + "Now, review your previous answers and identify any incorrect responses, then solve" + question + choices.

By employing these various prompt formulations, we aimed to evaluate their influence on the accuracy of ChatGPT's responses in addressing the multiple questions.

Method	Accuracy
trial 1	63.4%
trial 2	64.0%
trial 3	**66.0%**
trial 4	63.9%
trial 6	64.7

Table 3: Accuracy on MLP dataset

3.4.1 Trial 1

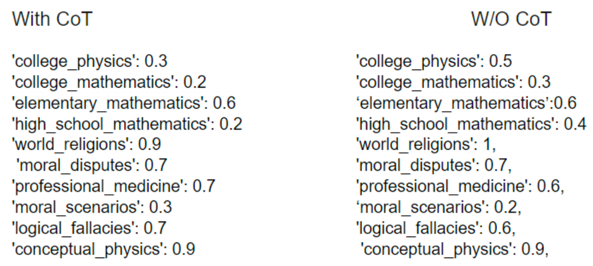
The utilization of this prompt in trial 1 yielded an accuracy rate of approximately 63.4%. While this performance may seem unsatisfactory, we have identified some patterns worthy of discussion.

Notably, we observed that context-based questions demonstrated significantly higher accuracy rates, exemplified by domains such as 'high school European history' (0.8), 'world religions' (0.9), and 'public relations' (0.8). Conversely, domains involving extensive calculations exhibited lower accuracy rates, as evidenced by 'college mathematics' (0.2) and 'college physics' (0.2). This outcome aligns with our expectations, as ChatGPT is not primarily designed for computational tasks, leading to potential errors in logical reasoning or calculations is acceptable. Furthermore, our findings suggest that ChatGPT struggles to comprehend human moral rules, as it performed poorly in moral scenarios, achieving a mere 20% accuracy. Consequently, we may refrain from employing ChatGPT for answering moral questions or any area that needs moral conditions.

Surprisingly, we encountered an area where ChatGPT displayed poor performance despite it being context-based and not reliant on calculations. This particular domain exhibited a disheartening accuracy rate of 20%. This finding contradicts our previous assumptions, prompting us to propose a simple conjecture. It is conceivable that during OpenAI's training process for ChatGPT, insufficient data was used incorporating knowledge about virology, resulting in an inadequate knowledge base for processing this domain accurately. However, due to time constraints, we were unable to conduct an in-depth investigation into this matter.

3.4.2 Trial 2

Based on Figure 2, it can be concluded that CoT is not a promising approach for enhancing ChatGpt. When the problem involves mathematical



With CoT	W/O CoT
'college_physics': 0.3	'college_physics': 0.5
'college_mathematics': 0.2	'college_mathematics': 0.3
'elementary_mathematics': 0.6	'elementary_mathematics':0.6
'high_school_mathematics': 0.2	'high_school_mathematics': 0.4
'world_religions': 0.9	'world_religions': 1,
'moral_disputes': 0.7	'moral_disputes': 0.7,
'professional_medicine': 0.7	'professional_medicine': 0.6,
'moral_scenarios': 0.3	'moral_scenarios': 0.2,
'logical_fallacies': 0.7	'logical_fallacies': 0.6,
'conceptual_physics': 0.9	'conceptual_physics': 0.9,

Figure 2: Some changes after using CoT

logic or calculations, allowing ChatGpt to explain its thought process can significantly improve the model's accuracy. However, for context-based questions that don't require calculations, using CoT may have little to no positive impact and even potentially harm the model's performance. It is assumed that explaining does not assist ChatGpt in finding relevant contextual information to answer questions but rather aids in performing mathematical calculations. Therefore, it is not advisable to utilize CoT unless your question necessitates both explanation and calculation.

3.4.3 Trial 3

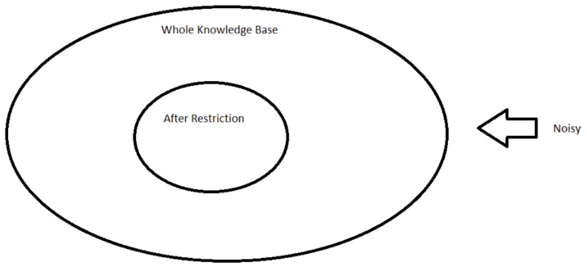


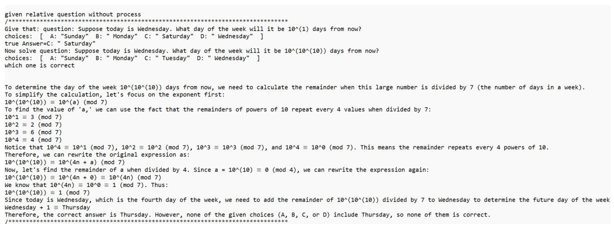
Figure 3: simple assumption

According to Table 2 (see 3), incorporating additional information about the question's area appears to be a promising approach. This inclusion contributes to an improvement of approximately 3% in ChatGPT's performance. Our assumption, as depicted in Figure 3, is relatively straightforward. The knowledge base used by ChatGPT is both vast and noisy. However, by providing supplementary information about the question, we can constrain the knowledge base, making it smaller and less prone to noise. Let us assume that the original probability of ChatGPT providing the correct answer is denoted as $P(\text{correct answer}|\text{question})$. With the introduction of the area information, this probability becomes $P(\text{correct answer}|\text{question, area})$. The inclusion of the area information should enhance the probability since it is directly related to the correct answer, thereby facilitating the emergence of the correct answer while reducing noise. Thus, it appears promising to incorporate this type of information when employing ChatGPT for downstream tasks.

3.4.4 Trial 4

Based on the findings presented in Table 2 (refer to 3), this particular approach does not yield favorable results. Essentially, it does not offer any noticeable positive effect on the model's performance. This outcome can be reasonable because the fact that questions within the same area do not necessarily imply the need for a shared approach or knowledge to solve them. Therefore, based on this observation, we proceed with the next trial.

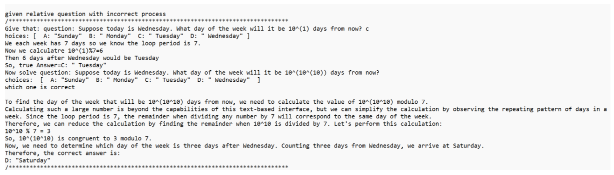
3.4.5 Trial 5



```
given relative question without process
/********************************************************************************
Give that: question: Suppose today is Wednesday. What day of the week will it be 10^(1) days from now?
choices:  [  A: "Sunday"  B: " Monday"  C: " Saturday"  D: " Wednesday"  ]
true Answer=C: " Saturday"
Now solve question: Suppose today is Wednesday. What day of the week will it be 10^(10^(10)) days from now?
choices:  [  A: "Sunday"  B: " Monday"  C: " Tuesday"  D: " Wednesday"  ]
which one is correct


To determine the day of the week 10^(10^(10)) days from now, we need to calculate the remainder when this large number is divided by 7 (the number of days in a week).
To simplify the calculation, let's focus on the exponent first:
10^(10^(10)) ≡ 10^(a) (mod 7)
To find the value of 'a,' we can use the fact that the remainders of powers of 10 repeat every 4 values when divided by 7:
10^1 ≡ 3 (mod 7)
10^2 ≡ 2 (mod 7)
10^3 ≡ 6 (mod 7)
10^4 ≡ 4 (mod 7)
Notice that 10^4 ≡ 10^1 (mod 7), 10^2 ≡ 10^2 (mod 7), 10^3 ≡ 10^3 (mod 7), and 10^4 ≡ 10^0 (mod 7). This means the remainder repeats every 4 powers of 10.
Therefore, we can rewrite the original expression as:
10^(10^(10)) ≡ 10^(4n + a) (mod 7)
Now, let's find the remainder of a when divided by 4. Since a = 10^(10) ≡ 0 (mod 4), we can rewrite the expression again:
10^(10^(10)) ≡ 10^(4n + 0) ≡ 10^(4n) (mod 7)
We know that 10^(4n) ≡ 10^0 ≡ 1 (mod 7). Thus:
10^(10^(10)) ≡ 1 (mod 7)
Since today is Wednesday, which is the fourth day of the week, we need to add the remainder of 10^(10^(10)) divided by 7 to Wednesday to determine the future day of the week
Wednesday + 1 ≡ Thursday
Therefore, the correct answer is Thursday. However, none of the given choices (A, B, C, or D) include Thursday, so none of them is correct.
/********************************************************************************
```

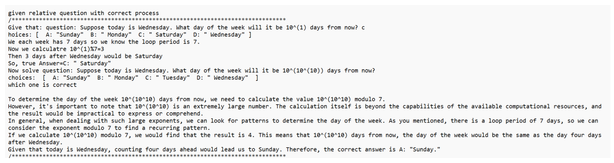
Figure 4: example without procedure



```
given relative question with incorrect process
/********************************************************************************
Give that: question: Suppose today is Wednesday. What day of the week will it be 10^(1) days from now? c
hoices: [  A: "Sunday"  B: " Monday"  C: " Tuesday"  D: " Wednesday" ]
We each week has 7 days so we know the loop period is 7.
Now we calculatre 10^(1)%7=6
Then 6 days after Wednesday would be Tuesday
So, true Answer=C: " Tuesday"
Now solve question: Suppose today is Wednesday. What day of the week will it be 10^(10^(10)) days from now?
choices:  [  A: "Sunday"  B: " Monday"  C: " Tuesday"  D: " Wednesday"  ]
which one is correct

To find the day of the week that will be 10^(10^10) days from now, we need to calculate the value of 10^(10^10) modulo 7.
Calculating such a large number is beyond the capabilities of this text-based interface, but we can simplify the calculation by observing the repeating pattern of days in a
week. Since the loop period is 7, the remainder when dividing any number by 7 will correspond to the same day of the week.
Therefore, we can reduce the calculation by finding the remainder when 10^10 is divided by 7. Let's perform this calculation:
10^10 % 7 = 3
So, 10^(10^10) is congruent to 3 modulo 7.
Now, we need to determine which day of the week is three days after Wednesday. Counting three days from Wednesday, we arrive at Saturday.
Therefore, the correct answer is:
D: "Saturday"
/********************************************************************************
```

Figure 5: example with incorrect procedure



```
given relative question with correct process
/********************************************************************************
Give that: question: Suppose today is Wednesday. What day of the week will it be 10^(1) days from now? c
hoices: [  A: "Sunday"  B: " Monday"  C: " Saturday"  D: " Wednesday" ]
We each week has 7 days so we know the loop period is 7.
Now we calculatre 10^(1)%7=3
Then 3 days after Wednesday would be Saturday
So, true Answer=C: " Saturday"
Now solve question: Suppose today is Wednesday. What day of the week will it be 10^(10^(10)) days from now?
choices:  [  A: "Sunday"  B: " Monday"  C: " Tuesday"  D: " Wednesday"  ]
which one is correct

To determine the day of the week 10^(10^10) days from now, we need to calculate the value 10^(10^10) modulo 7.
However, it's important to note that 10^(10^10) is an extremely large number. The calculation itself is beyond the capabilities of the available computational resources, and
the result would be impractical to express or comprehend.
In general, when dealing with such large exponents, we can look for patterns to determine the day of the week. As you mentioned, there is a loop period of 7 days, so we can
consider the exponent modulo 7 to find a recurring pattern.
If we calculate 10^(10^10) modulo 7, we would find that the result is 4. This means that 10^(10^10) days from now, the day of the week would be the same as the day four days
after Wednesday.
Given that today is Wednesday, counting four days ahead would lead us to Sunday. Therefore, the correct answer is A: "Sunday."
/********************************************************************************
```

Figure 6: example with correct procedure

Initially, it is obvious that providing a sample without a procedure proves to be unhelpful, which aligns with expectations as ChatGPT cannot extract any useful information from such an example. Therefore, we attempt to instruct ChatGPT on how to solve this type of question. However, even when we provide the correct procedure but intentionally introduce incorrect calculations, we observe that

ChatGPT becomes confused and provides an incorrect answer. As a result, we proceed by offering ChatGPT a completely accurate example, allowing the model to follow the pattern and successfully solve the question, ultimately arriving at the correct answer. This observation indicates that both logic and calculation are essential for ChatGPT when presented with an example. However, due to limitations in human effort, we can only provide a brief intuition rather than a detailed analysis.It would be a good idea to provide accurate and detailed examples when letting ChatGPT answer questions.

3.4.6 Trial 6

In the preceding series of experiments, we demonstrated the effectiveness of providing additional information and examples to aid ChatGPT in answering questions. However, an important consideration arises when limited human resources are available for generating such supplementary materials. Hence, we propose a novel approach: let ChatGPT find the issue. As indicated by the results in Table 2 (refer to 3), this approach achieves the second-highest overall accuracy. We assume the reason why it improved is similar to trial 3 3, but with the distinction that ChatGPT now provides additional information itself. Consequently, the previous answer serves as a guiding factor for ChatGPT, enabling it to identify the crucial knowledge required to solve the question more effectively during subsequent attempts. This discovery is remarkable to use as we can combine this approach with trial 3, enabling ChatGPT to provide the area of the question or related information that may aid in achieving improved results. Importantly, this technique does not require any human resources, making it particularly advantageous when addressing multiple-choice questions. We believe that this idea provides valuable insights into the utilization of ChatGPT and sheds light on its inner workings. We hope that this approach will prove beneficial for future work in this field.

3.5 MATH

In this section, we evaluated the math ability of ChatGPT 3.5 turbo (tempurature = 0, top-p = 0.1, Maximum length = 1024) on 600 problems from two math datasets: MATH (Hendrycks et al., 2021b) and Mathematics dataset (Hendrycks et al., 2021a) in different prompt settings. MATH dataset consists of challenging contest mathematics questions which require complex reasoning and calculation. The questions were divided into 5 different levels of difficulties and each question is categorized as one of five subcategories of math: Algebra, Number Theory, Probability, Precalculus, and Geometry. Mathematics dataset contains a comprehensive list of math questions including baisc arithmatic, equation solving, float rounding, calculus differentiating, sorting, GCD calculation, etc.

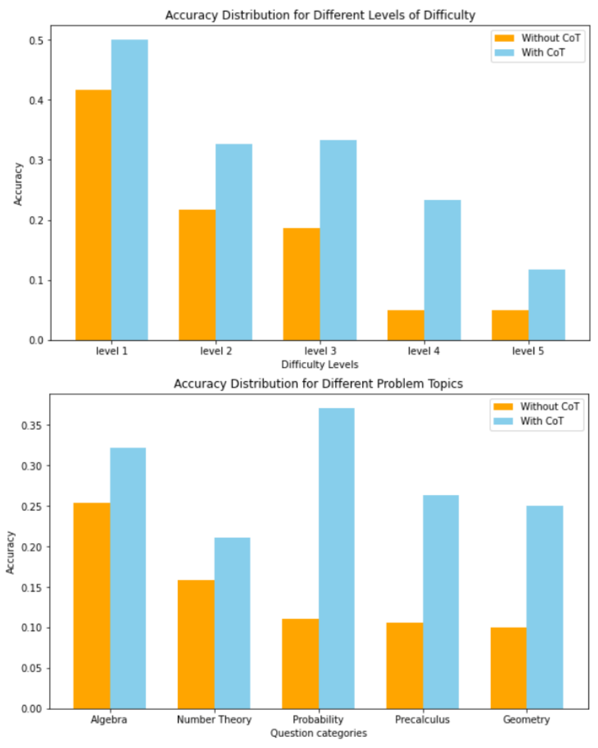


Figure 7: Baseline and CoT comparison on MATH

We first compared the model's results on MATH, both with and without Chain-of-Thoughts prompting, displayed in Figure 7. Generally, an increase in difficulty results in a decrease in the model's accuracy, while the implementation of Chain-of-Thoughts prompting enhances performance across all difficulty levels. As shown in Figure 7, CoT prompting demonstrates superior effectiveness in tasks requiring complex reasoning (Probability) compared to those requiring more calculation (Algebra) or complicated definitions (Number Theory).

To further analyze how we can improve over the current result, we manually inspect all the wrong CoT responses and collect the percentages of different kinds of errors: problem understanding, calculation, reasoning, or answer reach max length. Figure 8 shows an example of CoT response generated by the model, and Figure 9 shows the pie plot for the error distribution.

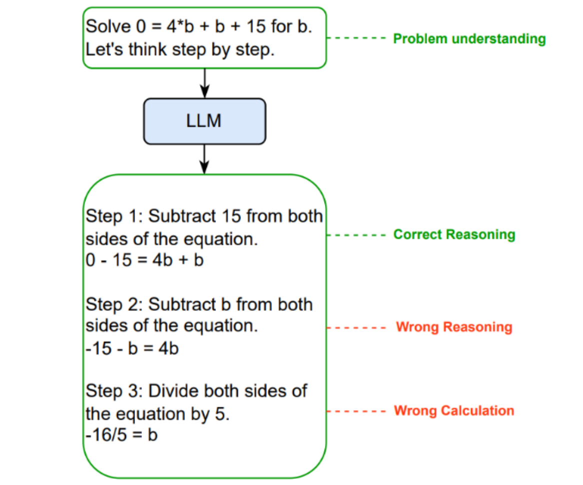

Solve 0 = 4*b + b + 15 for b.
Let's think step by step.
Problem understanding
LLM
Step 1: Subtract 15 from both sides of the equation.
0 - 15 = 4b + b
Correct Reasoning
Step 2: Subtract b from both sides of the equation.
-15 - b = 4b
Wrong Reasoning
Step 3: Divide both sides of the equation by 5.
-16/5 = b
Wrong Calculation

Figure 8: Chain of Thoughts Example

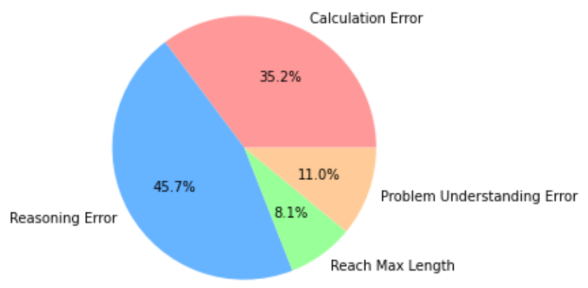


Figure 9: Error Distribution of CoT Prompting

Based on problem understanding, reasoning, and calculation errors, we use three different prompting methods to improve the performance. Due to the limited time and large human efforts in evaluating accuracy on MATH dataset, we performed the following experiments on Mathematics dataset where we can use code to compute the accuracy.

Few-Shot (8) Prompting (Figure 10) In this method, we randomly select eight examples from different subcategories and compose detailed, step-by-step Chain of Thoughts solutions for each. These examples are then appended at the beginning of each input question to provide in-context learning examples. We wish to use this approach to minimize errors from wrong reasoning path.

Problem Explanation Prompting (Figure 11) In this method, we ask the model to first interpret the original input questions and then output the answer. We wish to direct the model's attention towards critical points and details of the questions, thus fostering a comprehensive understanding of

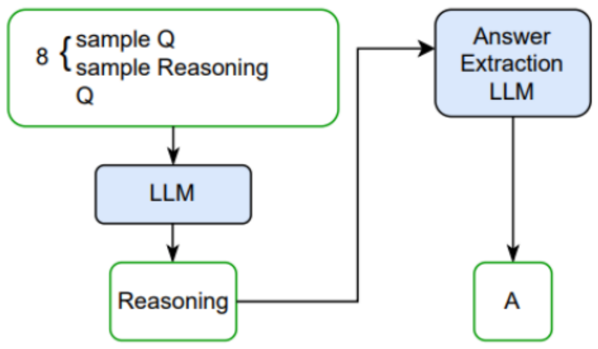


Figure 10: Few Shot Prompting

the problem and reducing errors due to misunderstandings.

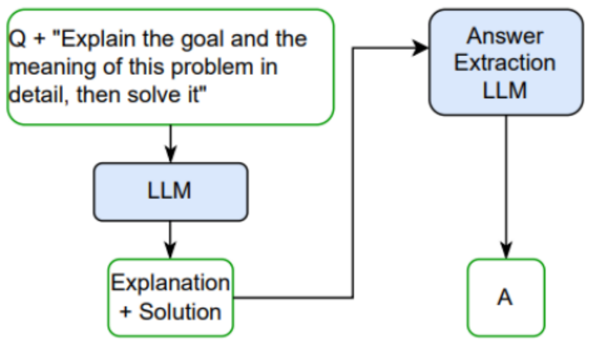

Q + "Explain the goal and the meaning of this problem in detail, then solve it"
LLM
Explanation + Solution
Answer Extraction LLM
A

Figure 11: Problem Explanation Prompting

Code Prompting (Figure 12) In this method, we ask the model to produce Python code that returns the answer to the original input questions. If the generated code is executable, we then employ the exec() function to obtain the answer; otherwise, we resort to the Chain of Thoughts response. We wish to use this approach to reduce errors associated with mathematical calculations.

The results are shown in table 4. All three prompting methods outperform the basic CoT by 3% to 7%, and the code prompting performed the best, reached an accuracy of 47.35%.

Method	Accuracy
Original input	26.79%
CoT	40.5%
Few-shot (8) CoT	45.79%
Explanation	43.61%
Code	**47.35%**

Table 4: Accuracy on Mathematics dataset

Long CoT error During our experimentation, we observed cases where the GPT model produced

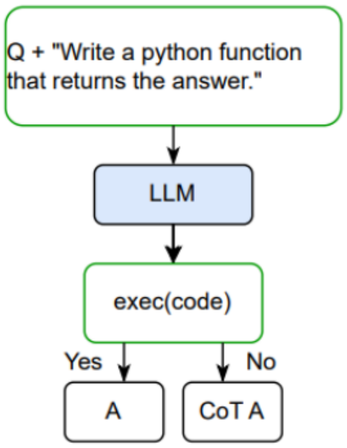


Figure 12: Code Prompting

correct answers without reasoning, yet produced wrong answers when reasoning was involved. This phenomenon typically occurred when the reasoning chain was very long or complex. To illustrate this, we designed a simple experiment where the model was instructed to count the number of 1s in a 01 sequence. We evaluated model accuracy on sequences ranging in length from 8 to 20. For each sequence length, we generated 100 sample questions and calculated the mean accuracy. The comparative performance of the model with and without reasoning is presented in Figure 13. For shorter sequences, the Chain-of-Thoughts reasoning method tended to perform better. However, with longer sequences, it was prone to errors. We assumed that when the question requires long reasoning, the model loses track of its previous outputs, resulting in flawed reasoning.

```
Count the number of 1s in the following
sequence: 0 1 0 1 0 1 1 1
Output the answer (arabic numerals) directly.
```

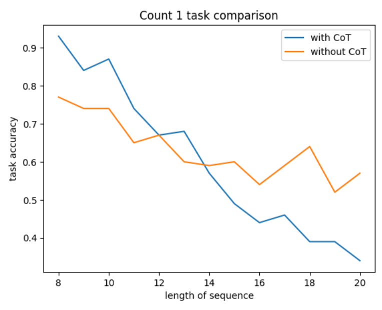


Figure 13: Count 1s task performance

4 Limitations

In this report, we only tested on ChatGPT 3.5 turbo model due to limited time. Moreover, for each dataset, we selected approximately 300 questions for testing. We acknowledge that this limited sample size could potentially lead to biased results. Additionally, our experimentation varied across different datasets, implying that certain experiments may not be generalizable to other datasets.

As we move forward, we will expand our experimentation with various prompting methods across a broader range of LLMs. We can also incorporate the complete datasets into our future testing process. This will help us in developing a general pipeline to improve the model's performance across a spectrum of tasks.

5 Conclusion

From these datasets, we have gathered that generally ChatGPT is better at solving knowledge based and coding questions than math and complex reasoning questions. We were able to improve the performance by providing relevant contextual information. We saw that we were able to confuse the model by adding confusion words and rephrasing questions with the same meaning, showing that it is not robust. For math questions, we could improve the performance by refining the problem's description, correcting the reasoning steps, and ensuring the calculation steps are accurate. After analyzing the responses by ChatGPT, we found that it is a sensitive model and making the prompts as clear as possible is important. It also is not great at long and complex Chain of Thoughts, and does better on shorter and more succinct queries. To improve the model's performance in tackling long and complex reasoning problems, we suggest implementing a working memory system in the future to mitigate the recurring issue of the model losing its track during the reasoning.

References

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners.

Chenxiao Liu and Xiaojun Wan. 2021. Codeqa: A question answering dataset for source code comprehension. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2618–2632.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3?

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *ArXiv*, abs/1904.01557.

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts?

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.