From Answers to Questions: A Study on Backward Reasoning in Large Language Models

Anonymous ACL submission

Abstract

Multi-step reasoning through Chain-of-Thought (CoT) prompting has been extensively explored, highlighting the abilities of Large Language Models (LLMs) to generate answers derived from step-by-step reasoning. These studies focus the attention on LLMs' forward reasoning abilities manifested in a series of general premises leading to a final solution. In this paper, we take the reverse perspective by analyzing the backward reasoning abilities of LLMs, namely the inference that leads to the causal hypothesis. We conduct a study on Question Answering (QA) tasks, aimed at analyzing whether LLMs are able to reason about the conclusions and deliver the original question that leads to the final answer. We use three Multiple Choice Questions and six Math Word Problems QA tasks: (i) we observe a performance gap between standard and proposed approaches; hence (ii) we propose several methods to elicit the LLMs to generate the answer by considering the backward direction.

1 Introduction

Several techniques for in-context learning through prompting approaches (Brown et al., 2020; Min et al., 2022) enable pre-trained Large Language Models (LLMs) (Chowdhery et al., 2022; Touvron et al., 2023; OpenAI, 2023) to generalize well on out-domain tasks, demonstrating versatility in a variety of tasks such as sentence completion, multiple choices text comprehension, and mathematical reasoning, by providing multi-step forward responses. Earlier works have extensively studied these problems, adopting previous (Cobbe et al., 2021; Roy and Roth, 2015; Patel et al., 2022) and new (Gao et al., 2023; Zheng et al., 2023b) datasets to observe the performance of powerful LLMs comparatively.

Recently, Wei et al. (2022) have proposed the Chain-of-Thought (CoT) prompt for LLMs, which generates necessary explicit intermediate steps to reach the final answer. Specifically, each example in-context is complemented by several steps described in natural language. In inference, the verification question is added to the prompt and fed to an LLM, mimicking the in-context examples and delivering reasoning steps before the final result. Many works have recently been proposed to improve its effectiveness (Yu et al., 2023; Wang et al., 2023) and efficiency (Wu et al., 2023). Later, Qiao et al. (2023); Zhou et al. (2023) proposed an advancement through Self-Verification techniques. Different outputs delivered to CoT are sampled using temperature sampling (Ficler and Goldberg, 2017). Behind this passage, the one that receives the most votes is selected as the final response. Although these techniques show the reasoning abilities of LLMs, they are based on observations of generating forward, leaving unexplored the ability to infer a rule given the consequences.

This leads to the target research questions, which are the focus of this paper:

(RQ1) Can the well-known Question-answering benchmarks be employed to observe the reasoning abilities of LLMs with the purpose of studying the effect in the backward direction?

(RQ2) Do the different complexities of forward and backward reasoning observed in human minds also reflected in LLMs??

(RQ3) Could LLMs' reasoning abilities be empowered by using the structure of the prompt and the generated answers?

In this paper, we investigate whether LLMs are able to deliver answers by performing backward reasoning steps, which consist of developing hypotheses for a set of facts and deducing the most probable cause or the most plausible explanation. We propose to use Question-answering (QA) tasks, in particular, six Math Word Problem (MPW) and three Multiple Choice Question (MCQ), traditionally used to study forward generative abilities. Hence, we propose two different approaches

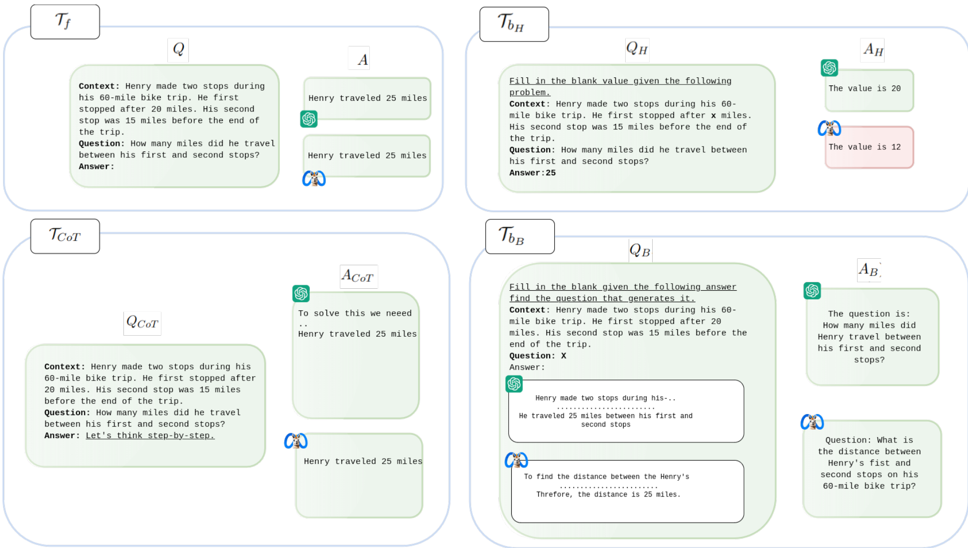

Figure 1: Overview of our proposed approaches.

(i.e., `Blanking` and `Hiding`) that revisit the standard prompts in order to elicit the LLMs to consider the initial question. In particular, in our approaches, we propose opposite versions of tasks by prompting the final answers as facts and eliciting the LLMs to reason about reconstructing the original question. We analyze whether different families of LLMs (GPT (OpenAI, 2023), Llama-2 (Touvron et al., 2023), Mistral (Jiang et al., 2023), and Orca2 (Mitra et al., 2023)) are able to deduce the original questions by proposing different prompting approaches.

Downstream of extensive analysis, we show a discrepancy regarding the performances obtained from forward and backward input-prompting. Therefore, we propose a series of approaches to stimulate LLMs to rephrase the problem by considering different shapes and achieving noticeable improvements.

Our contributions can be summarized as follows:

- Formalization of backward reasoning problem by proposing two different kinds of intervention in nine different benchmarks commonly used to test forward generative abilities of LLMs (Yuan et al., 2023; Ling et al., 2023).
- In-depth study of the divergences between forward reasoning obtained through standard prompting and backward reasoning obtained via our `Hiding` and `Blanking` approaches on different LLM families.
- Performance improvement via prompt-based approaches that elicit LLMs to reason about the input structures for the input problems.

2 Problem Formulation

A reasoning-based question-answering (QA) task is defined as a tuple $\mathcal{T}_f = (Q, O, A)$, where Q is the question, that could contain context C, such as the necessary background for answering a question; $O = (o_1, o_2, .., c_n)$ are answer choices if Q is a multiple choice (n) problem (C and O could be optional depending from the task); and A is the target answer. Given Q as input-prompt, Large Language Models (LLMs) generate the answer (output) that is a sequence of tokens $T_{out} = (t_1, t_2, ..., t_n)$. The generated answer is correct if and only if the $(t_i, .., t_m) \subseteq T$ matches the ground truth A. Recent works like Chain-of-Thought (CoT) (Wei et al., 2023) leverage prompt engineering in the context C to elicit LLMs to generate the intermediate reasoning process in T_{out}, which benefits their performance across diverse reasoning tasks. In this case, T_{out} consists of a set of m intermediate reasoning steps, which we denote as $S = (s_1, s_2, ..., s_m)$. Each step s_i can be represented by a subsequence of the generated tokens $s_i = (t_1, t_2, ...t_n) \subseteq T_{out}$. The generated solution is correct if the predicted final answer in s_i matches the ground truth A. Given the forward generative nature, the premise of C and Q, and the conclusion generated in the sequence T, it is possible to describe this as a deductive process (Huang and Chang, 2023; Ling et al., 2023).

In our work, we introduce $\mathcal{T}_b$ that is the opposite of $\mathcal{T}_f$. Starting from a QA task, given the answer A as evidence, we want to infer the rule (or, in our

Prompt: Multiple Choices Question $\mathcal{T}_f$
Question: <Question>
Choices:
a) <Option1>
b)...
Answer:
+ Let's think step by step (CoT Prompt)
generated answer $\mathcal{A}$ or $\mathcal{A}_{CoT}$

Prompt: Math Word Problem $\mathcal{T}_f$
Question: <Question>
Answer:
+ Let's think step by step (CoT Prompt)
generated answer $\mathcal{A}$ or $\mathcal{A}_{CoT}$

Table 1: Example of prompt for MCQs (left) and MWPs (right) Question Answering tasks.

Prompt: Hiding Approach $\mathcal{T}_{b_H}$
Fill in the blank value given the following problem.
Context: $t_1, t_2, \ldots, \underline{\mathbf{x}}, \ldots, t_{n-1}, t_n$
Question: <final question>
Answer: A
+ Let's think step by step (CoT Prompt)

Prompt: Blanking Approach $\mathcal{T}_{b_B}$
Fill in the blank given the following answer. Find the question that generates it.
Context: $t_1, t_2, \ldots, t_{n-1}, t_n$
Question: $\underline{\mathrm{x}}$
Answer: $\mathcal{A}$ or $\mathcal{A}_{CoT}$
+ Let's think step by step (CoT Prompt)

Table 2: Example of prompt for Hiding Approach $\mathcal{T}_{b_H}$ and Blanking Approach $\mathcal{T}_{b_H}$.

case, the question Q) that generated A. In particular, as described in Section 3, we propose two different versions of $\mathcal{T}_b$: in $\mathcal{T}_{b_H} = (Q_H, O, A_H)$, the relaxed version, we contextualize the generation of Q using Q_H, that is Q with a strategic hide part with a placeholder x and in a strict version $\mathcal{T}_{b_B} = (Q_B, O, A_B)$ we do not use Q or its derivates. Hence, in the first version, the final goal is to find out the x omitted from the prompt, and in the second one, the goal is to generate Q_B, as in the abductive reasoning process (Huang and Chang, 2023; Qiao et al., 2023).

In this scenario, we prompt the LLMs, as shown in Figure 1, in order to elicit them to reconstruct or generate the rule using the final evidence that is exemplified respectively by the question Q and answer A.

3 Method

In order to observe the backward abilities of LLMs, we propose a prompting intervention based on using the target answer A and the context provided by task C in order to deduce the original Q.

Hence, behind defined problem $\mathcal{T}_b$ in Section 2, we describe the construction of $\mathcal{T}_{b_B}$ (Section 3.2) and $\mathcal{T}_{b_H}$ (Section 3.1).

3.1 Hiding Approach

In order to elicit LLMs to retrieve the original Q by reasoning in a backward way, we propose $\mathcal{T}_{b_H} = (Q_H, O, A_H)$. In particular, we contextualize the generation of Q using Q_H, i.e., Q with an hide strategic part with a placeholder x. Consequently, we replace the target answer A_H with x. However, the hiding approach differs according to the nature of the question-answering task.

Math Word Problem The MWP tasks are characterized by a tuple (Q, A) where numerical values represent the strategic information. Following the approaches from the previous work (Deb et al., 2023), we mask the numerical value in the prompt with x (placeholder value). Hence, we produce the input-prompts using Q_H and A. Where Q_H is very close to Q, with the numerical value replaced by an x (detailed in Appendix B.1). Then, we evaluate the accuracy by performing a string matching between the generated answer and x (x used as a placeholder in the prompt).

Multiple Choices Question In the MCQ setting, it is more challenging to determine which strategic part to blank. The datasets introduced in Section 4.1 are characterized by tuples (Q, O, A). In each Q, a strategic concept S is presented that is generally provided in the dataset but is not used for the evaluation. We replace $S \in Q$ with x deriving Q_x (detailed in Appendix B.1). We evaluate the accuracy by performing a string matching between the generated answer and x.

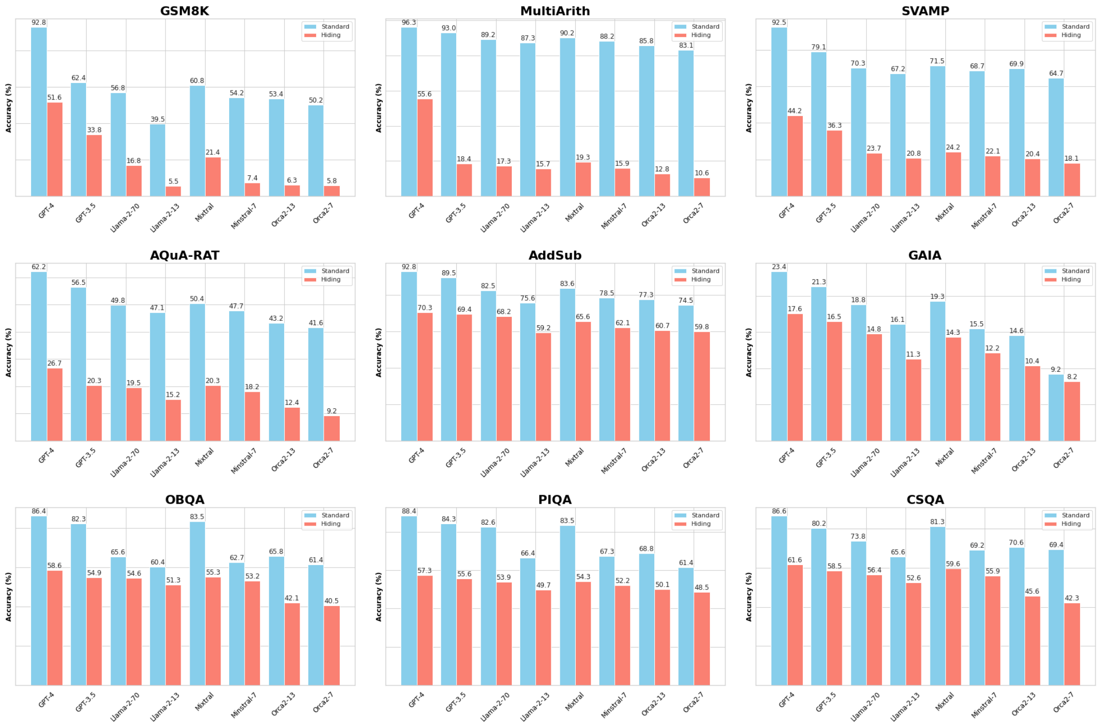

Figure 2: Accuracies (%) on Math Word Problem and Multiple Choices Questions proposed in Section 4.1 using `Standard` prompting approach (as shown in Table 8) and `Hiding` approach (Section 3.1).

3.2 Blanking Approach

Furthermore, we propose a stricter version of the tasks. Starting from $\mathcal{T}_b$ we propose $\mathcal{T}_{b_H} = (Q_B, O, A_B)$. We do not alter Q using the hiding approach, but blank entire Q, i.e., Q_B, reply with x. Consequently, the final target A, in our formulation A_B, is the original Q blanked with x. Then, we construct the input prompt, as shown in Figure 1 and in Table 2.

However, it is not possible to apply the Blanking approach directly to all tasks, for example, on MWPs that have only a numerical A target, and it is impossible to generate Q (or A_B) without having context. In order to solve this problem, we introduce $\mathcal{A}$ described in Section 3.3 for the Math Word Problem and the Multiple Choices Question tasks. Finally, we estimate the correctness of generated answers using BERTScore (Zhang et al., 2020) between the blanked question Q and the generated answer T_{out}.

3.3 Backward Answer

Behind proposing the $\mathcal{T}_{b_H}$ approach for constructing altered prompts to evaluate the abilities of LLMs, we introduce a Blanking approach, $\mathcal{T}_{b_B}$. However, LLMs need more context that targets A alone cannot supply. Therefore, we introduce $\mathcal{A}$ by constructing it by prompting the LLMs with input-prompts (as in Figure 1, Table 1, and Table 2). Moreover, we use the multi-step reasoning abilities by also proposing $\mathcal{A}_{CoT}$ that is based on the Chain-of-Thought prompt technique (Wei et al., 2023). Then, we use the generated answers, $\mathcal{A}$ and $\mathcal{A}_{CoT}$, as a component to produce $\mathcal{T}_{b_B}$ as Figure 1 (all passages are detailed in Appendix B.2).

4 Experiments

In order to analyze the abductive reasoning abilities of Large Language Models (LLMs), we propose two backward reasoning approaches in Math Word Problem (MWP) and Multiple Choices Question (MCQ) tasks introduced in Section 4.1. Then, we systematically prompt different LLMs as described in Section 4.2 by evaluating the answers generated using Section 4.3's evaluation methods.

4.1 Data

We propose our experimental setup by adapting the method proposed in Section 3 to two typologies of Question-answering (QA) tasks:

QA Math Word Problem MPW tasks are characterized by a question (a mathematical problem) in natural language and a target answer, which in most cases is a number. We select five different datasets

Strategy	Model	GSM8K$_H$	SVAMP$_H$	MultiArith$_H$	AQua-RAT$_H$	AddSub$_H$	GAIA$_H$
Hiding (0-shot)	GPT-3.5	33.8$_{\pm.4}$	36.3$_{\pm.2}$	18.4$_{\pm.1}$	69.4$_{\pm.3}$	20.3$_{\pm.1}$	16.5$_{\pm.2}$
Hiding (5-shot)	GPT-3.5	35.4$_{\pm.3}$	38.4$_{\pm.4}$	20.5$_{\pm.3}$	70.6$_{\pm.4}$	22.1$_{\pm.3}$	18.6$_{\pm.3}$
CoT (5-shot)	GPT-3.5	34.5$_{\pm.4}$	35.3$_{\pm.4}$	19.5$_{\pm.1}$	70.2$_{\pm.3}$	19.4$_{\pm.5}$	15.9$_{\pm.1}$
Complex-CoT (0-shot)	GPT-3.5	40.5$_{\pm.1}$	39.9$_{\pm.1}$	21.7$_{\pm.2}$	73.7$_{\pm.3}$	24.5$_{\pm.6}$	21.2$_{\pm.4}$
Complex-CoT (5-shot)	GPT-3.5	43.5$_{\pm.2}$	41.3$_{\pm.2}$	26.4$_{\pm.2}$	76.6$_{\pm.3}$	24.8$_{\pm.2}$	26.3$_{\pm.4}$
Paraphrasing (2-shot)	GPT-3.5	50.2$_{\pm.3}$	45.8$_{\pm.4}$	36.8$_{\pm.3}$	79.2$_{\pm.4}$	26.7$_{\pm.2}$	29.8$_{\pm.2}$
	Llama-2-70	29.3$_{\pm.2}$	37.2$_{\pm.3}$	25.6$_{\pm.2}$	76.3$_{\pm.1}$	29.2$_{\pm.2}$	29.2$_{\pm.1}$
	Mixtral	28.9$_{\pm.2}$	31.5$_{\pm.1}$	30.1$_{\pm.2}$	69.9$_{\pm.1}$	29.0$_{\pm.0}$	30.0$_{\pm.1}$
Paraphrasing (5-shot)	GPT-3.5	56.7$_{\pm.1}$	50.3$_{\pm.1}$	41.9$_{\pm.4}$	**83.8**$_{\pm.2}$	32.1$_{\pm.1}$	**33.9**$_{\pm.4}$
	Llama-2-70	34.1$_{\pm.1}$	44.1$_{\pm.2}$	31.7$_{\pm.3}$	80.1$_{\pm.1}$	33.1$_{\pm.3}$	35.0$_{\pm.3}$
	Mixtral	33.9$_{\pm.1}$	38.9$_{\pm.2}$	33.3$_{\pm.1}$	73.8$_{\pm.4}$	33.7$_{\pm.1}$	36.2$_{\pm.5}$
Self-Refine (2-shot)	GPT-3.5	53.8$_{\pm.2}$	49.1$_{\pm.3}$	40.1$_{\pm.4}$	80.1$_{\pm.3}$	30.4$_{\pm.2}$	30.1$_{\pm.4}$
	Llama-2-70	34.1$_{\pm.4}$	40.1$_{\pm.1}$	31.7$_{\pm.3}$	78.2$_{\pm.3}$	30.1$_{\pm.3}$	33.2$_{\pm.3}$
	Mixtral	32.1$_{\pm.2}$	36.1$_{\pm.1}$	30.1$_{\pm.5}$	72.5$_{\pm.2}$	33.1$_{\pm.6}$	**32.1**$_{\pm.3}$
Paraphrasing +Self-Refine (2-shot)	GPT-3.5	**66.2**$_{\pm.3}$	**58.8**$_{\pm.1}$	**45.9**$_{\pm.3}$	82.6$_{\pm.4}$	**39.3**$_{\pm.1}$	32.9$_{\pm.2}$
	Llama-2-70	33.9$_{\pm.1}$	**42.3**$_{\pm.1}$	**35.9**$_{\pm.3}$	**78.7**$_{\pm.1}$	**36.5**$_{\pm.5}$	**36.1**$_{\pm.1}$
	Mixtral	**39.1**$_{\pm.5}$	44.3$_{\pm.1}$	**31.6**$_{\pm.4}$	**75.1**$_{\pm.2}$	**35.1**$_{\pm.5}$	31.3$_{\pm.2}$

Table 3: Improvements in accuracy with various prompting strategies in the Hiding approach. In Table 9 the results of other models.

with this type of structure: GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), MultiArith (Roy and Roth, 2015), AddSub (Hosseini et al., 2014) AQuA (Ling et al., 2017), GAIA (Mialon et al., 2023).

QA Multiple Choices Question MCQ tasks, unlike MWPs, have different structure. This type of task consists of a question, a context that is optional, and multiple choices. In our work, we select four resources: CommonSenseQA (Talmor et al., 2019) (CSQA) and OpenBookQA (Mihaylov et al., 2018) (OBQA) regarding commonsense reasoning, Physical Interaction Question Answering (Seo et al., 2018) (PIQA) regarding physical reasoning.

Finally, we systematically construct $\mathcal{T}_{b_H}$ and $\mathcal{T}_{b_B}$ (see Table 2), as described in Section 3 and detailed in Appendix B.

4.2 Models

In order to test the LLMs' abilities, we select different models by attempting to get at least two models from the same families, but differing in the number of parameters. In particular, we select: two GPT models (OpenAI, 2023) (GPT-4 and GPT-3.5-turbo), two Llama-2 models (Touvron et al., 2023) (Llama-2-70 and Llama-2-13), two Mistral models (Jiang et al., 2023) (Mixtral and Mistral-7b) and finally two Orca2 models (Mitra et al., 2023) (Orca2-7b and Orca2-13b). For more details on the parameters, see Appendix A.

4.3 Evaluation

We evaluate the performance of the LLMs introduced in Section 4.1 on the tasks defined in Section 4.2. The evaluation is conducted using the accuracy for the Hiding approach $\mathcal{T}_{b_H}$ and (F1-score) of BERTScore (Zhang et al., 2020) for the Blanking approach $\mathcal{T}_{b_B}$. We use BERTScore because the generation of the entire question could be correct, even if delivered with different terminology. In addition, in Appendix X we discuss an additional analysis performed with an LLM (GPT-4) as a judge.

5 Results & Discussion

Large Language Models (LLMs) are able to seek hypotheses that best approximate the explanation of a set of observations. In fact, LLMs generate answers when they are elicited to consider the fact that caused the final evidence. This statement can be demonstrated by the results shown in Figure 3. The proposed LLMs (Section 4.2) have inferred the initial question that generated the final answers in both Multiple Choices Questions and Math Word Problem tasks in the Blanking approach (proposed in Section 3.2).

However, although most LLMs perform well in the Blanking approach, we observe a different phenomenon in the Hiding approach. Figure 2 shows the accuracies obtained following standard zero-shot prompts and our Hiding Approach presented in Section 3.1. Although the task is quite different, there is a substantial gap between the performances. The nature of the differences between the final results in the Blanking approach (Section 5.1) and Hiding approach (Section 5.2) can be traced back to the structure of the prompt. Therefore, in Section 5.3 we propose techniques that improve the performance of the Hiding approach.

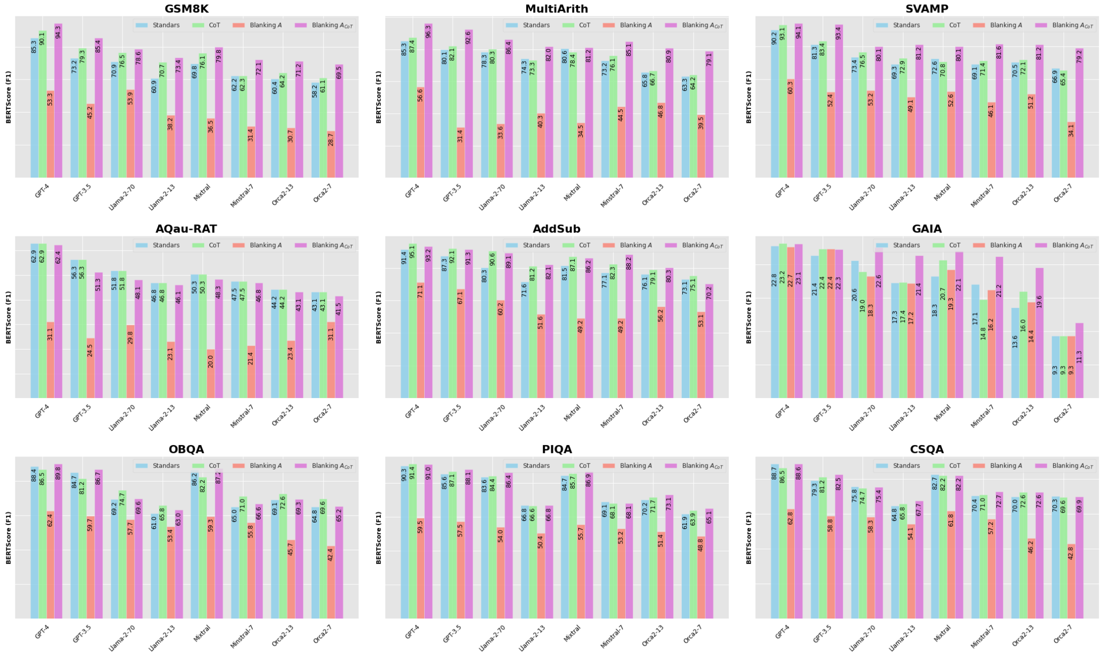

Figure 3: Performances (BERTScore F1) on Math Word Problem and Multiple Choices Questions proposed in Section 4.1 using Standard prompting approach (as shown in Table 1) and `Blanking` approach proposed in Section 3.2

Finally, if the prompt techniques proposed in Section 5.3 allow for improving the LLMs' abilities in the Hiding approach, in Section 5.4 we reconsider the Blanking approach by proposing the Cross-Blanking Test that stresses the LLMs' abilities by breaking the data contamination phenomenon.

5.1 Blanking Results

LLMs are able to reason about the evidence delivered in a multi-step way by reconstructing initial assumptions. As shown in Figure 3, the correctness of the Blanking approach (described in Section 3.2) is, on average, high when the input-prompts are formed with A_{CoT}, i.e., answers generated via CoT prompts (Wei et al., 2023). In order to have a term of comparison, we have reported the same evaluations, F1 BERTScore (Zhang et al., 2020), as well as the forward prompting approaches (described in Figure 1 and Table 2). Weak note for the Blanking approach that takes the answer A as evidence. Indeed, A alone is too context-poor to allow LLMs to reason about the prior blanked questions. Although the scores are, on average, high, motivation could lie in the presence of critical parts of the question in the evidence we provide in the input prompts. Consequently, this could be mistaken as a data contamination problem. In order to observe whether LLMs are able to reason in the opposite direction, we propose Cross-Blanking experiment in Section 5.4. Specifically, in this experiment, we provide as A_{CoT} the responses generated by other LLMs to perform the Cross-Blank evaluation (see Table 6).

5.2 Hiding Approach Results

LLMs fail to retrieve the hidden information in prompts. Table 8 shows the accuracies of different LLMs presented in Section 4.2. A clear difference can be seen between the standard prompts, where the models are prompted with a problem they should generate an answer, and the Hiding approach, where the models are asked to reconstruct the hidden part of the question. However, a significant difference can be observed because there is a smaller average gap in the MCQ tasks than in MWP (see Table 8 in the Appendix H). This phenomenon leads us to study the input composition, as we hypothesize that these average differences can be traced back to the present content. In fact, in the MCQ tasks, there is more context (e.g., the various choices) than in MWP, where the answer is entirely coincident.

5.3 Prompting Approaches

Manipulating the structure of the prompt leads LLMs to better reasoning in a backward direction. Table 3 shows the performance of the different techniques, in zero-shot and few-shot (In-context Learning (ICL) (Brown et al., 2020; Shin et al., 2022)), that made final improvements over those discussed in Section 5.2. Hence, we discuss the different approaches tested using GPT-3.5 and Llama-2-70 as base models.

CoT vs Complex-CoT CoT approaches in both zero-shot and few-shot scenarios do not contribute to substantially increasing baseline performances by highlighting the limitation of the input structure (Table 3 and Table 9). Moreover, we observe the same tendency for Complex-CoT (Fu et al., 2023). We hypothesize that these are the consequences of the LLMs' difficulty processing the input-prompt proposed in the Hiding approach (Section 3.1).

Paraphrasing Rephrasing the prompt helps LLMs understand the problem to be addressed. We detected a noticeable increase in downstream performances of the Paraphrasing technique (performances in Table 3). The method is described in Appendix C.

Self-Refine Although paraphrasing input prompts support LLMs in understanding the problem better, iteratively reconsidering the feedback until a predetermined condition is reached (Self-Refine) has overpowered all approaches. We notice conspicuous improvements in our experiments adapting the original Self-Refine to our Hiding approach (detailed description in Appendix E) (see Table 3 and Table 9).

5.4 Cross-Blanking Test

LLMs are able to reconstruct the initial problem and perform the reasoning in a backward direction by understanding the answers delivered by other LLMs. This is shown in Table 4. In particular, we have revisited the Blanking Approach from a Cross-perspective. More in detail, we construct the input-prompts as described in Section 3.2, but, instead of providing A_{CoT} generated by the evaluating LLM, we cross-reference the demonstrations (see Table 6 in Appendix G). We reproduce the experiments using one mathematical and one multiple-choice question task. From the results in Table 4, it is possible to observe an in-family phenomenon. The models of the same family seem to achieve similar performances, which is not observable in the out-family models. However, the models obtain sustainable performances.

5.5 Metrics Error Analysis & Limitations

The results discussed in Sections 5.1 demonstrate LLMs' ability to provide answers while considering backward-facing problems. Following the various techniques used to elicit generation in different scenarios, we qualitatively analyze the results obtained and the metrics behind them, highlighting limitations and strengths.

BERTScore vs LLMs-judge In the Blanking Task (Section 3.2), the evaluation metric used was BERTScore. However, this metric may have limitations, as there could be multiple valid questions for a given context and response, and it is not clear if BERTScore can distinguish between two semantically different questions with the same answer. For this reason, in Table 11, we discuss using GPT-4 as an evaluator judge, revealing that the results do not differ dramatically.

The Numerical Limitation On the side of the Hiding approach, we further consider the responses generated by different LLMs in the MWP tasks. Here, a potential limitation is associated with evaluating the generated placeholders. The placeholders generated could be numerical values but not in numeric format, rather nominal. To avoid this phenomenon, we (i) include the keyword [num] in the input prompts and (ii) implement a secondary check using a conversion function discussed in the Appendix.

Error Analysis Paraphrasing the prompt has its benefits. As shown in Appendix C, the approach proposed in Section 3.1 appears to work in the case of a few-shot scenario reinforced with a self-refined approach, while it seems to lead to misleading and incorrect responses when the approaches are employed alone (see Table 3).

6 Related Work

Question Answering Problem Question-answering tasks are generally characterized by a natural language description that can be a question in the case of Multiple Choice Questions (MCQ) tasks or a mathematical problem in the case of Math Word Problems (MWP) tasks (Lu et al., 2023). The description expresses the relations between various entities or quantities followed by a query for an unknown quantity in the case of MWP, and a known quantity in the case of MCQ. One must represent the relationship between entities and quantities to respond to the query. The resolution of MWP, but especially that of MCQ, requires a semantic understanding of the natural language description. The initial works (Koncel-Kedziorski et al., 2015; Roy and Roth, 2018) for solving these tasks propose to parse the description using statistical learning techniques to identify suitable models for generating answers. Behind the advent of sequence-to-sequence (Seq2Seq) models (Sutskever et al., 2014) for automatic translation, the approaches for solving MCQ and MWP tasks diverge. For MWP (Wang et al., 2017; Shen et al., 2021; Jie et al., 2022), they propose encoder-decoder frameworks to translate the natural language description of MWPs into equations directly. In MCQ (Multiple Choice Questions), many studies have proposed methods for retrieving answers from Knowledge bases (Banerjee et al., 2019) or generating the answer using prior knowledge (Abujabal et al., 2018).

Large Language Models Recently, Large Language Models (LLMs) such as GPTs (Brown et al., 2020; OpenAI, 2023), Llamas (Touvron et al., 2023), PaLM (Chowdhery et al., 2022) have been achieving outstanding performance in both MWPs and MCQs tasks without the use of external knowledge bases or further analysis methods. These models use the ability of context-based instances via a few-shot iteration and use prompting methods such as CoT (Wei et al., 2023), all without demanding any parameter modifications. Several approaches (Welleck et al., 2022; Madaan et al., 2023) that use LLMs involve verifying the response provided by the Language Model, either using the model itself or external verifiers like compilers or proof checkers. If the response is incorrect, the model is re-prompted, potentially with suggestions to improve its output. This querying process continues until the model generates the correct output. Different techniques, such as Progressive Hint Prompting (Zheng et al., 2023a), iteratively pass the model's previous responses to itself as hints. Iterative querying techniques like those in (Weng et al., 2023) do not use a verifier; instead, they sample multiple hypotheses from the model and select the answer via majority voting.

Reasoning Direction As described in the introduction to our paper, we focus on a precise case of abductive reasoning with a single answer. Abductive reasoning (Qin et al., 2020; Thayaparan et al., 2021; Zhao et al., 2023) consists of inferring which of several explanations is the most plausible. Previous work on abductive reasoning has mainly focused on textual reasoning under constraints. In arithmetic reasoning tasks, Weng et al. (2023) used abductive reasoning to improve the accuracy of forward reasoning. Our work, on the other hand, addresses backward reasoning as an independent problem. We are inspired by what Deb et al. (2023) proposed and take it further by extending us to more tasks and scaling the tests to different models with softer and stricter formalization. Our primary interest lies in analyzing the inherent complexities of reasoning and creating more effective solutions to deal with it.

7 Conclusion

This paper explores the abilities of Large Language Models (LLMs) in forward and backward generative ways. We introduce two novel approaches, namely Hiding and Blanking, to challenge LLMs to infer the original question from the answers given. Our experiments reveal interesting insights into the LLMs' abilities. While LLMs show proficiency in forward reasoning, their performances in backward reasoning vary significantly. The Hiding approach, which partially obscures the original question, demonstrates that LLMs could, to some extent, reconstruct missing elements. Moreover, the Blanking approach, which presents a more challenging scenario by completely removing the original question, highlights the effective abilities. Our research also delves into various prompting techniques to empower the LLMs' performance in these tasks to elicit the LLM to understand and approach the problems better. Our study opens new avenues for understanding and improving the reasoning abilities of LLMs. It also raises important questions about the future directions of LLMs development, particularly in areas requiring complex, multi-directional reasoning abilities.

Limitations

In our work, we analyzed the abilities of Large Language Models (LLMs) in solving reverse question-answering and math word problems. Specifically, starting from the original settings where a question is provided and the LLM is required to generate an answer, we examined the reverse task. This analysis reveals the strengths and weaknesses of LLMs in generating reverse reasoning. Potentially, reverse reasoning could be useful when faced with evidence and one wishes to trace back to the phenomenon that caused them by reasoning backward. In this work, we used the BERTScore and the judgment-based assessment of GPT-4 as judgment metrics. In future work, we will study the effect of additional metrics in order to improve the evaluative aspect.

Ethics Statement

In our work, ethical topics were not addressed. The data comes from open-source benchmarks, and statistics on language differences in commonly used pre-training data were obtained from official sources without touching on gender, sex, or race differences.

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2018. Never-ending learning for open-domain question answering over knowledge bases. Proceedings of the 2018 World Wide Web Conference.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Pratyay Banerjee, Kuntal Kumar Pal, Arindam Mitra, and Chitta Baral. 2019. Careful selection of knowledge to solve open book question answering. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6120–6129, Florence, Italy. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Aniruddha Deb, Neeva Oza, Sarthak Singla, Dinesh Khandelwal, Dinesh Garg, and Parag Singla. 2023. Fill in the blank: Exploring and enhancing llm capabilities for backward reasoning in math word problems.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In Proceedings of the Workshop on Stylistic Variation, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In Findings of the Association for Computational Linguistics: ACL 2023, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Zhanming Jie, Jierui Li, and Wei Lu. 2022. Learning to reason deductively: Math word problem solving as complex relation extraction. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5944–5955, Dublin, Ireland. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. Transactions of the Association for Computational Linguistics, 3:585–597.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. A survey of deep learning for mathematical reasoning. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14605–14631, Toronto, Canada. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. MetaICL: Learning to learn in context. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2791–2809, Seattle, United States. Association for Computational Linguistics.

MistralAI. Mistral-7b-instruct-v0.2. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agrawal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching small language models how to reason.

OpenAI. 2023. Gpt-4 technical report.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2080–2094, Online. Association for Computational Linguistics.

Pruthvi Patel, Swaroop Mishra, Mihir Parmar, and Chitta Baral. 2022. Is a question decomposition unit all we need? In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 4553–4569, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5368–5393, Toronto, Canada. Association for Computational Linguistics.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 794–805, Online. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. Transactions of the Association for Computational Linguistics, 6:159–172.

Minjoon Seo, Tom Kwiatkowski, Ankur P. Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Phrase-indexed question answering: A new challenge for scalable document comprehension.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2269–2279, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, and Nako Sung. 2022. On the effect of pretraining corpora on in-context learning by a large-scale language model. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5168–5186, Seattle, United States. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2021. Explainable inference over grounding-abstract chains for science questions. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1–12, Online. Association for Computational Linguistics.

TheBloke. a. Llama-2-70b-chat-gptq. https://huggingface.co/TheBloke/Llama-2-13B-Chat-GPTQ.

TheBloke. b. Llama-2-70b-chat-gptq. https://huggingface.co/TheBloke/Llama-2-70B-Chat-GPTQ.

TheBloke. c. Mixtral-8x7b-moe-rp-story-gguf. https://huggingface.co/TheBloke/Mixtral-8x7B-v0.1-GPTQ.

TheBloke. d. Orca-2-13b-gguf. https://huggingface.co/TheBloke/Orca-2-13B-GGUF. Accessed: [your access date].

TheBloke. e. Orca-2-7b-gguf. https://huggingface.co/TheBloke/Orca-2-7B-GGUF.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. Towards understanding chain-of-thought prompting: An empirical study of what matters. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models.

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.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin

Choi. 2022. Generating sequences by learning to self-correct.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification.

Dingjun Wu, Jing Zhang, and Xinmei Huang. 2023. Chain of thought prompting elicits knowledge augmentation. In Findings of the Association for Computational Linguistics: ACL 2023, pages 6519–6534, Toronto, Canada. Association for Computational Linguistics.

Fangyi Yu, Lee Quartey, and Frank Schilder. 2023. Exploring the effectiveness of prompt engineering for legal reasoning tasks. In Findings of the Association for Computational Linguistics: ACL 2023, pages 13582–13596, Toronto, Canada. Association for Computational Linguistics.

Zhangdie Yuan, Songbo Hu, Ivan Vulić, Anna Korhonen, and Zaiqiao Meng. 2023. Can pretrained language models (yet) reason deductively?

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

Wenting Zhao, Justin T. Chiu, Claire Cardie, and Alexander M. Rush. 2023. Abductive commonsense reasoning exploiting mutually exclusive explanations.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023a. Progressive-hint prompting improves reasoning in large language models.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification.

A Model and Hyperparameters

As introduced in Section 4.2, we used:

- two models from the GPT family (OpenAI, 2023): GPT-4 and GPT-3.5-turbo (GPT-3.5) used via API.
- two models from the Llama-2 family (Touvron et al., 2023): Llama-2-70b and Llama-2-13b using versions of the quantized to 4-bit models using GPTQ (TheBloke, a,b).
- two models of the Orca2 family (Mitra et al., 2023): Orca2-7b (TheBloke, e) and Orca2-13b (TheBloke, d).
- two models of the MistralAI family: Mistral-7b and Mixtral using official version on huffingface (MistralAI) versions of the quantized to 4-bit models using GPTQ (TheBloke, c).

We use closed-source API or the 4-bit GPTQ quantized version of the model on two 48GB NVIDIA RTXA600 GPUs for all experiments performed only in inference. All experiments use a generation temperature of [0, 0.5] for (mostly) deterministic outputs, with a maximum token length of 256. The other parameters are left unchanged as recommended by the official resources. We will release the code and the dataset upon acceptance of the paper.

B Dataset Construction

We use six different Math Word Problem datasets: GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), MultiArith (Roy and Roth, 2015), AddSub (Hosseini et al., 2014), AQuA (Ling et al., 2017), MathQA (Amini et al., 2019). We describe the generation methodology of the final composition of $\mathcal{T}_{b_H}$ in Section B.1 and $\mathcal{T}_{b_B}$ in Section B.2. Downstream of the generation methodologies, we filtered the original datasets by removing the examples we could not parse optimally (see Table 10).

B.1 Generation for Hiding Approach

Math Word Problems As introduced in Section 3.1, in $\mathcal{T}_{b_H} = (Q_H, A_H)$ (in MWP there are not O), we construct Q_H from Q. For each question of $Dataset$:

$$\{(Q_i, A_i)\}_{i=1}^{n} | Q_i \in \Sigma^*, A_i \in \mathbb{R}\}$$

We propose a method to create $Dataset'_k$:

$$\{(Q'_i, A_i, (H_i^0, \ldots, H_i^k))\}_{i=1}^{n} | Q'_i \in \Sigma^*, H_i^j \in \mathbb{R}\}$$

Generator	Task	Evaluator					
		GPT-4	GPT-3.5	Llama-2-70	Llama-2-13b	Mixtral	Mistral-7b
GPT-4	GSM8K	**94.3**$_{\pm.1}$	92.5$_{\pm.3}$	84.4$_{\pm.6}$	83.3$_{\pm.3}$	78.2$_{\pm.2}$	76.3$_{\pm.2}$
	CSQA	88.6$_{\pm.5}$	87.4$_{\pm.4}$	75.6$_{\pm.1}$	74.5$_{\pm.2}$	67.9$_{\pm.3}$	66.3$_{\pm.2}$
GPT-3.5	GSM8K	90.9$_{\pm.2}$	85.4$_{\pm.5}$	72.3$_{\pm.2}$	69.4$_{\pm.4}$	67.3$_{\pm.3}$	65.2$_{\pm.2}$
	CSQA	81.9$_{\pm.3}$	82.5$_{\pm.3}$	71.9$_{\pm.1}$	68.5$_{\pm.3}$	64.7$_{\pm.2}$	63.6$_{\pm.3}$
Llama-2-70	GSM8K	76.1$_{\pm.3}$	75.6$_{\pm.5}$	78.6$_{\pm.3}$	78.5$_{\pm.2}$	62.9$_{\pm.4}$	60.9$_{\pm.1}$
	CSQA	65.3$_{\pm.3}$	65.8$_{\pm.5}$	75.4$_{\pm.3}$	74.3$_{\pm.2}$	61.9$_{\pm.2}$	59.4$_{\pm.2}$
Llama-2-13	GSM8K	81.4$_{\pm.3}$	80.6$_{\pm.2}$	75.3$_{\pm.4}$	73.4$_{\pm.2}$	60.9$_{\pm.1}$	59.2$_{\pm.4}$
	CSQA	82.2$_{\pm.3}$	81.9$_{\pm.3}$	70.9$_{\pm.3}$	67.7$_{\pm.1}$	59.1$_{\pm.5}$	58.2$_{\pm.2}$
Mixtral	GSM8K	83.8$_{\pm.3}$	81.6$_{\pm.5}$	68.3$_{\pm.2}$	65.8$_{\pm.3}$	79.8$_{\pm.1}$	77.9$_{\pm.3}$
	CSQA	74.8$_{\pm.2}$	72.3$_{\pm.3}$	65.3$_{\pm.4}$	63.2$_{\pm.3}$	82.2$_{\pm.3}$	81.3$_{\pm.2}$
Mistral-7b	GSM8K	78.7$_{\pm.3}$	77.9$_{\pm.3}$	67.5$_{\pm.3}$	66.6$_{\pm.1}$	73.9$_{\pm.4}$	72.1$_{\pm.1}$
	CSQA	69.4$_{\pm.4}$	67.8$_{\pm.1}$	62.3$_{\pm.2}$	61.8$_{\pm.4}$	76.4$_{\pm.4}$	72.7$_{\pm.3}$

Table 4: Performances Cross-Blanking test. In this test, we elicit the models to generate the Blanked question (Section 3.2) using the A delivered from other LLMs. "Generator" refers to the model that generates the A. "Evaluator" refers to the model that is prompted to generate the initial question (example shown in Appendix G).

To convert Q in Q_H and extract the numerical subparts $H_i^0, \ldots, B_i^k$, we split Q_H into its constituent tokens. Hence, we consider all numeric tokens as tokens that encode a number. Numeric tokens may be alphanumeric, such as 150 or 2.23, or alphabetic, such as three, twice, or half. Using this heuristic for numeric tokens, we ignore the first numeric token and extract the following k tokens sequentially. We skip that question-and-answer pair if we cannot extract k tokens. It is worth noting that for the datasets we use, $k = 1$, we only consider the problem of backwardly inferring one missing number in the question, given the answer. To simplify the process and better adapt it to the subsequent Blanking approach as well, when possible, we differentiate the main question of the problem (structurally defined by the "?" character that ends the sentence or sub-sentence) by splitting the $Question$ and the $Concept$ as shown in Figure 1.

Multiple Choice Question As introduced in Section 3.1, MCQ tasks do not always have easily maskable symbols, such as numerical values. Here, our contribution is different. Given $\mathcal{T}_{b_H} = (Q_H, A_H)$, we construct Q_H from Q. For each question of $Dataset$:

$$\{(Q_i, A_i)\}_{i=1}^{n} | Q_i \in \Sigma^*, A_i \in \mathcal{C}\}$$

where $\mathcal{C}$ represents the set of choice options in MCQs. We propose a method to create $Dataset'_k$:

$$\{(Q'_i, A_i, (P_i^0, \ldots, P_i^k))\}_{i=1}^{n} | Q'_i \in \Sigma^*, P_i^j \in \Sigma^*\}$$

To convert Q in Q_H and extract the noun subparts $P_i^0, \ldots, P_i^k$, we split Q_H into its constituent tokens and perform part-of-speech (POS) tagging. We specifically identify nouns, which may be subjects or objects, as our primary tokens of interest. These tokens are processed and tagged using a POS tagging algorithm. We sequentially extract the first k identified noun tokens for each question. We skip that question-and-answer pair if we cannot extract k noun tokens. Again, we use $k = 1$, meaning we focus on the challenge of inferring a single missing noun in the question, given the answer.

B.2 Generation for Blanking Approach

As introduced in the section 3.1, in $\mathcal{T}_{b_B} = (A_B, O, Q_B)$, we replicate Q with x as shown in Table 2. However, to contextualize the generation, we substitute the A with $\mathcal{A}$ or $\mathcal{A}_{CoT}$ for the target generated via the CoT prompt. We propose this approach for both task types.

C Paraphrasing Prompting

To test if prompting approaches could infer the final answer, our initial strategy concerns transforming the problem through paraphrasing, as also proposed by (Deb et al., 2023). This method simplifies the complex reasoning challenge into a more suitable forward reasoning task. As a result, we apply the LLM to this more manageable, rephrased forward reasoning problem rather than grappling with the more arduous backward reasoning task.

In the case of a $\mathcal{T}_{b_H} = (A_H, O, Q_H)$, we prompt the language model to generate a different prompt P. This rephrased prompt integrates the forward answer A_H into the original question Q_H, altering the goal from discovering the answer A_H to determining the value of the blank. We then direct the language model to address this rephrased problem P, bypassing the initial problem.

The results, as illustrated in Table 3 and Table 9, reveal that changing the problem and changing the problem by posing the value of x and instructing the LLM to ascertain the value of x, as illustrated in Table 7, yields better results than classic prompting strategies.

D Self-Refine

Moreover, we utilize the Self-Refine framework proposed by Madaan et al. (2023). This approach is also employed in Self-Verification prompting by (Weng et al., 2023). This iterative prompting technique alternates between refinement and feedback until a predefined condition is met. We have modified the technique to perform backward reasoning on our tasks as done in (Deb et al., 2023).

E Paraphrased Self-Refine Prompting

To test whether prompting approaches can infer the final answer, our initial strategy involves transforming the problem through paraphrasing. This method simplifies the complex challenge of abductive reasoning into a simpler deductive reasoning task. Consequently, we apply the LLM to this more manageable and reformulated reasoning problem instead of tackling the more arduous abductive reasoning task.

Hence, we propose a further experiment by including paraphrase and self-consistency to obtain higher accuracy (Table 3 and Table 9).

F GPT-4 as a Judge

In Section X, we used BERTScore to evaluate the performances achieved by different models in the Blanking task introduced in Section Y. In this additional experiment, we replicate the Cross-Blanking test using GPT-4 as the judge, which, given the original question and the question generated by the LLM under test, will produce a positive or negative judgment that we will define as accuracy.

In Table X, where we have reported the accuracies obtained, we can observe no sensible differences compared to Table Y. Therefore, even though the two metrics are not directly comparable, BERTScore approximates the accuracy of a GPT-4 evaluator well.

G Prompting Approaches

Prompt: MCQ $\mathcal{T}_f$ to $\mathcal{M}_1$	*Prompt:* MCQ $\mathcal{T}_f$ to $\mathcal{M}_2$
Question: <Question>	**Question:** <Question>
Choices:	**Choices:**
a) <Option1>	a) <Option1>
b)...	b)...
Answer:	**Answer:**
+ <u>Let's think step by step</u> (CoT Prompt)	+ <u>Let's think step by step</u> (CoT Prompt)
generated answer $\mathcal{M}_1$ ($\mathcal{A}'$ or $\mathcal{A}'_{CoT}$)	*generated answer* $\mathcal{M}_2$ ($\mathcal{A}''$ or $\mathcal{A}''_{CoT}$)

Table 5: Example of input-prompt for Cross-Blanking Task.

Prompt: Cross-Blanking Approach on $\mathcal{M}_1$	*Prompt:* Cross-Blanking Approach on $\mathcal{M}_2$
Fill in the blank given the following answer find the question that generates it.	Fill in the blank given the following answer find the question that generates it.
Context: $t_1, t_2, \ldots, t_{n-1}, t_n$	**Context:** $t_1, t_2, \ldots, t_{n-1}, t_n$
Question: <u>x</u>	**Question:** <u>x</u>
Answer: $\mathcal{A}''$ or $\mathcal{A}''_{CoT}$	**Answer:** $\mathcal{A}'$ or $\mathcal{A}'_{CoT}$

Table 6: Example of Cross-Blanking Task where we provide to $\mathcal{M}_1$ the $\mathcal{A}''_{CoT}$ generated from $\mathcal{M}_2$, and vice versa.

Paraphrase Prompting
Question: A grove has 15 trees. Today, grove workers will add x trees. What will be the total number of trees after this addition? **Answer:** 21 **Paraphrased:** A grove has 15 trees. Grove workers added x trees today. The total becomes 21 trees. Calculate the value of x. **Answer:** Originally, there are 15 trees. After planting, the total is 21 trees. Therefore, x = 21 - 15 = 6 trees. The solution is 6. **Question:** he parking lot currently holds 3 cars. If x additional cars arrive, what is the total number of cars in the parking lot? Answer: 5 Paraphrased: There are 3 cars in the parking lot initially, and x additional cars arrive, making a total of 5 cars. Determine x. Answer: Initially, there are 3 cars. After x cars arrive, 3 + x = 5, hence x = 5 - 3 = 2. The solution is 2. **Question:** <Question> **Answer:** <Answer> **Paraphrasis:**

Table 7: Paraphrasis prompting.

H Detailed Results

Dataset	Approach	GPT-4	GPT-3.5	Llama-2-70	Llama-2-13	Mixtral	Minstral-7	Orca2-13	Orca2-7
				Math Word Problem					
GSM8k	`Standard`	92.8 ±.2	62.4 ±.1	56.8 ±.3	39.5 ±.2	60.8 ±.4	54.2 ±.2	53.4 ±.4	50.2 ±.1
	`Hiding`	51.6 ±.2	33.8±.4	16.8±.3	5.5±.3	21.4±.3	7.4±.3	6.3±.4	5.8±.1
SVAMP	`Standard`	92.5±.4	79.1±.3	70.3±.2	67.2±.2	71.5±.2	68.7±.1	69.9±.2	64.7±.4
	`Hiding`	44.2±.3	36.3±.2	23.7±.3	20.8±.2	24.2±.2	22.1±.1	20.4±.2	18.1±.3
MultiArith	`Standard`	96.3±.4	93.0±.4	89.2±.2	87.3±.1	90.2±.2	88.2±.3	85.8±.2	83.1±.2
	`Hiding`	55.6±.3	18.4±.1	17.3±.3	15.7±.4	19.3±.2	15.9±.1	12.8±.2	10.6±.3
AddSub	`Standard`	92.8±.3	89.5±.2	82.5±.2	75.6±.4	83.6±.2	78.5±.1	77.3±.2	74.5±.3
	`Hiding`	70.3±.3	69.4±.3	68.2±.2	59.2±.2	65.6±.2	62.1±.1	60.7±.2	59.8±.4
AQuA-RAT	`Standard`	62.2±.3	56.5±.2	49.8±.4	47.1±.4	54.4±.2	47.7±.1	43.2±.2	41.6±.2
	`Hiding`	26.7±.2	20.3±.1	19.5±.2	15.2±.3	20.3±.2	18.2±.4	12.4±.2	9.2±.3
GAIA	`Standard`	23.4±.2	21.3±.2	18.8±.4	16.1±.2	19.3±.3	15.5±.2	14.6±.1	9.2±.1
	`Hiding`	17.6±.4	16.5±.2	14.8±.3	11.3±.2	14.3±.4	12.2±.1	10.4±.2	8.2±.4
				Multiple Choices Question					
CSQA	`Standard`	86.6±.1	80.2±.2	73.8±.4	65.5±.2	81.3±.3	69.2±.2	70.6±.3	69.4±.2
	`Hiding`	61.6±.2	58.5±.1	56.4±.4	52.6±.4	59.6±.2	55.9±.3	45.6±.2	42.3±.2
OBQA	`Standard`	86.4±.2	82.3±.2	65.6±.2	60.4±.1	83.5±.2	62.7±.4	65.8±.2	61.4±.3
	`Hiding`	58.6±.3	54.9±.1	54.6±.4	51.3±.2	55.3±.2	53.2±.4	42.1±.2	40.5±.3
PIQA	`Standard`	88.4±.2	84.3±.1	82.6±.2	66.4±.4	83.5±.3	67.3±.2	68.8±.3	61.6±.2
	`Hiding`	57.3±.4	55.6±.4	53.9±.3	47.7±.2	54.3±.1	52.2±.1	50.1±.2	48.5±.4

Table 8: Accuracies (%) on dataset proposed in Section 4.1 using `Standard` and `Hiding` approaches.

Strategy	Model	$\mathrm{GSM8K}_H$	SVAMP_H	$\mathrm{MultiArith}_H$	$\mathrm{AQua\text{-}RAT}_H$	AddSub_H	GAIA_H
Hiding (0-shot)	`GPT-3.5`	33.8±.4	36.3±.2	18.4±.1	69.4±.3	20.3±.1	16.5±.2
	`Llama-2-70`	16.8±.3	23.7±.3	17.3±.2	68.2±.2	19.5±.3	14.8±.2
	`Mixtral`	21.4±.3	24.2±.2	19.3±.3	65.6±.2	20.3±.1	14.3±.4
Hiding (5-shot)	`GPT-3.5`	35.4±.3	38.4±.4	20.5±.3	70.6±.4	22.1±.3	18.6±.3
	`Llama-2-70`	20.3±.4	24.3±.3	18.9±.2	70.3±.3	20.6±.3	16.5±.2
	`Mixtral`	22.5±.2	25.6±.2	20.5±.2	66.6±.4	23.0±.1	16.3±.3
CoT (5-shot)	`GPT-3.5`	34.5±.4	35.3±.4	19.5±.1	70.2±.3	19.4±.5	15.9±.1
	`Llama-2-70`	15.9±.1	24.2±.3	14.6±.3	68.4±.2	18.2±.1	15.1±.3
	`Mixtral`	20.8±.3	22.1±.3	20.2±.3	64.9±.1	21.4±.3	15.1±.3
Complex-CoT (0-shot)	`GPT-3.5`	40.5±.1	39.9±.1	21.7±.2	73.7±.3	24.5±.6	21.2±.4
	`Llama-2-70`	20.9±.2	28.4±.1	16.9±.3	69.8±.4	22.3±.2	20.1±.4
	`Mixtral`	21.2±.3	23.1±.3	20.6±.1	65.0±.2	24.1±.1	18.2±.1
Complex-CoT (5-shot)	`GPT-3.5`	43.5±.2	41.3±.2	26.4±.2	76.6±.3	24.8±.2	26.3±.4
	`Llama-2-70`	22.4±.3	30.5±.1	17.2±.2	70.2±.1	22.3±.2	23.0±.2
	`Mixtral`	22.3±.1	24.5±.4	22.6±.1	65.8±.3	24.6±.1	20.2±.2
Paraphrasing (2-shot)	`GPT-3.5`	50.2±.3	45.8±.4	36.8±.3	79.2±.4	26.7±.2	29.8±.2
	`Llama-2-70`	29.3±.2	37.2±.3	25.6±.2	76.3±.1	29.2±.2	29.2±.1
	`Mixtral`	28.9±.2	31.5±.1	30.1±.2	69.9±.1	29.0±.0	30.0±.1
Paraphrasing (5-shot)	`GPT-3.5`	56.7±.1	50.3±.1	41.9±.4	**83.8**±.2	32.1±.1	**33.9**±.4
	`Llama-2-70`	34.1±.1	44.1±.2	31.7±.3	80.1±.1	33.1±.3	35.0±.3
	`Mixtral`	33.9±.1	38.9±.2	33.3±.1	73.8±.4	33.7±.1	36.2±.5
Self-Refine (2-shot)	`GPT-3.5`	53.8±.2	49.1±.3	40.1±.4	80.1±.3	30.4±.2	30.1±.4
	`Llama-2-70`	34.1±.4	40.1±.1	31.7±.3	78.2±.3	30.1±.3	33.2±.3
	`Mixtral`	32.1±.2	36.1±.1	30.1±.5	72.5±.2	33.1±.6	**32.1**±.3
Paraphrasing +Self-Refine (2-shot)	`GPT-3.5`	**66.2**±.3	**58.8**±.1	**45.9**±.3	82.6±.4	**39.3**±.1	32.9±.2
	`Llama-2-70`	33.9±.1	**42.3**±.1	**35.9**±.3	**78.7**±.1	**36.5**±.5	**36.1**±.1
	`Mixtral`	**39.1**±.5	44.3±.1	**31.6**±.4	**75.1**±.2	**35.1**±.5	31.3±.2
(Deb et al., 2023)							
custom Prompt "CW"	`GPT-3.5`	41.8	49.7	51.1	-	-	-
Ensemble	`GPT-3.5`	65.3	**66.7**	92.6	-	-	-

Table 9: Improvements in accuracy with various prompting strategies in the `Hiding` approach.

Name	URL	total examples	used examples
GSM8k	https://huggingface.co/datasets/gsm8k	1320	1270
AddSub	https://huggingface.co/datasets/allenai/lila/viewer/addsub	109	105
MultiArith	https://huggingface.co/datasets/ChilleD/MultiArith	420	350
AQuA-RAT	https://huggingface.co/datasets/aqua_rat	360	316
SVAMP	https://huggingface.co/datasets/MU-NLPC/Calc-svamp	1000	1000
GAIA	https://huggingface.co/datasets/gaia-benchmark/GAIA	466	195
CSQA	https://huggingface.co/datasets/commonsense_qa	1100	1100
OBQA	https://huggingface.co/datasets/openbookqa	500	500
PIQA	https://huggingface.co/datasets/piqa	3000	2000

Table 10: We report the sources where we download the datasets used in our work. For each dataset containing many instances, we randomly composed a subset.

Generator	Task	Evaluator					
		GPT-4	GPT-3.5	Llama-2-70	Llama-2-13b	Mixtral	Mistral-7b
GPT-4	GSM8K	95.3	94.3	87.2	84.5	81.6	79.8
	CSQA	92.3	89.5	79.7	78.9	71.3	69.6
GPT-3.5	GSM8K	92.1	89.2	75.6	72.3	70.6	69.8
	CSQA	82.3	84.1	73.3	70.2	69.7	69.3
Llama-2-70	GSM8K	77.6	78.7	81.3	80.5	66.7	62.1
	CSQA	66.4	67.2	78.4	76.3	62.9	62.3
Llama-2-13	GSM8K	83.2	81.7	76.8	76.4	63.1	61.3
	CSQA	83.4	82.6	72.3	69.1	61.3	60.4
Mixtral	GSM8K	84.3	85.6	71.4	67.9	82.3	79.3
	CSQA	76.3	74.5	66.2	66.9	83.4	85.3
Mistral-7b	GSM8K	79.4	80.1	69.5	68.6	75.5	73.5
	CSQA	71.3	69.6	66.4	64.3	77.9	76.8

Table 11: Performances Cross-Blanking test using GPT-4 as a judge. In this test, we elicit the models to generate the Blanked question (Section 3.2) using the A delivered from other LLMs. "Generator" refers to the model that generates the A. "Evaluator" refers to the model that is prompted to generate the initial question (example shown in Appendix G). Unlike Table 4, we use GPT-4 as the judge (accuracy) instead of the previously used BERTScore in this experiment.