Exploring Backward Reasoning in Large Language Models

Leonardo Ranaldi **Giulia Pucci**
School of Informatics, University of Edinburgh, UK
Department of Computing Science, University of Aberdeen, UK
Idiap Research Institute, Martigny, Switzerland
University of Rome Tor Vergata, Italy
{first_name.last_name}@ed.ac.uk

Abstract

Multi-step reasoning through in-context learning strategies have 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 epitomised in a series of general premises leading to a final solution.

In this paper, by taking the reverse perspective, we study the backward reasoning abilities of LLMs, namely the inference that leads to the causal hypothesis. Behind formalising the backward problems, we analyse whether the LLMs are able to reason about the conclusion and reconstruct the original question that led to the delivery of the final answer. Operating with question-answering tasks involving symbolic reasoning, understanding, and commonsense abilities, we observe that the proposed models reveal robust comprehension capabilities managing different kinds of input; however, they are not always able to reason in the backward direction. Finally, to challenge this limitation, we demonstrate that instructing LLMs to generate the answer by reconsidering the structure of the problem allows for improved backward reasoning direction.

1 Introduction

Multi-step reasoning through Chain-of-Thought (CoT) *(et alia)* have been extensively explored, underlining the abilities of Large Language Models (LLMs) to solve problems in a step-wise manner. These strategies enable LLMs to generalise on out-of-domain tasks, demonstrating versatility in diverse assignments such as sentence completion, multiple-choice text comprehension, and mathematical reasoning by delivering multi-step forward responses. Specifically, each in-context demonstration is complemented by several steps represented in natural language. At inference time, the verification question is added to the prompt and fed to an LLM, mimicking the provided demonstrations and delivering reasoning steps before the final result. Many works have been proposed to improve its effectiveness and efficiency (Wu et al., 2023; Wang et al., 2023) in mono- and multi-lingual spaces (Ranaldi et al., 2024b,c). In parallel, multiple works study the impact of different in-context problem-solving frameworks such as Program-Aided Language Models (PAL) (Gao et al., 2023), or ensembling techniques, self-verification strategies (Qiao et al., 2023; Zhou et al., 2023) with the collective aim of achieving better results.

While these methods demonstrate considerable improvements across various benchmarks and display proficiency in generating linguistic patterns that mimic human logical reasoning, they remain constrained to a forward generative process. They forget to explore the potential of deriving underlying rules from given outcomes by adopting a backward reasoning perspective.

This leads to the target research questions:

RQ1 Can the well-known question-answering tasks be employed to observe the reasoning abilities of LLMs to study 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 using the structure of the inputs 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 from a set of facts and deducing the most probable cause or the most plausible explanation. We introduce question-answering tasks with Multiple Choice Questions (MCQ) structures and extend the evaluation to further Math Word Problem (MPW) task. Hence, we propose two different approaches (i.e., `Blanking` and `Hiding`).

Findings of the Association for Computational Linguistics: NAACL 2025, pages 6586–6601
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

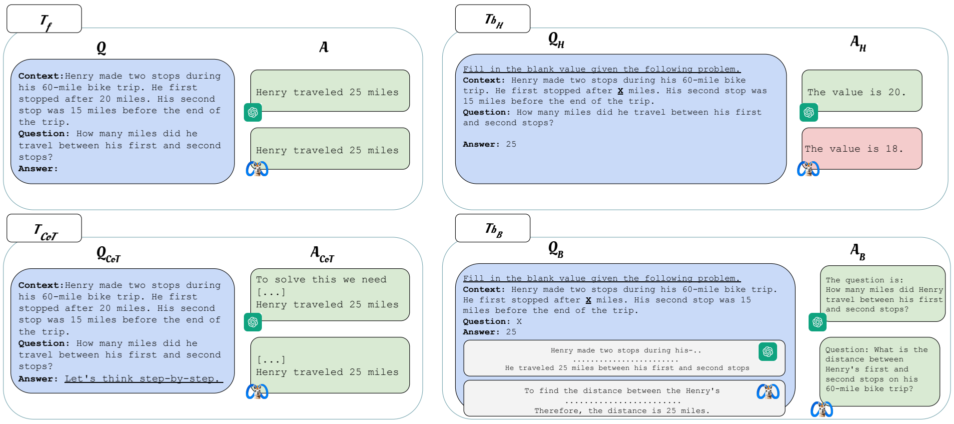

Figure 1: Overview of our proposed approaches.

The study of the backward process, i.e., reconstructing the questions from the outcomes, delivers evidence of the ability to understand the process and profitability of LLMs that are systematically posed to different elicitation approaches. Thus, to have a comprehensive overview, we operate on different versions of the best-known LLMs exemplified by GPT (OpenAI, 2023), Llama-2 (Touvron et al., 2023), Mistral (Jiang et al., 2023) and Orca2 (Mitra et al., 2023).

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

Our contributions can be outlined as follows:

- Formalization of the backward reasoning problem and proposal of two intervention approaches in nine benchmarks commonly used to test forward generative abilities of LLMs.
- Study about divergences between forward reasoning obtained through standard prompting and backward way via our `Hiding` and `Blanking` approaches on different models.
- Demonstration of performance improvement via prompt operation approaches that elicit LLMs to reason about the input structures for the given 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 on the task); and A is the target answer. Given Q as input, 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 case, the question Q) that generated A. As described in §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, the goal is to generate Q_B, as in the backward reasoning process

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: $t_1, t_2, \ldots,$ **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: 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 $\mathcal{T}_{b_H}$.

(Huang and Chang, 2023; Qiao et al., 2023).

In this scenario, we prompt the LLMs, as shown in Figure 1, 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

To observe LLMs' backward abilities, we propose a prompting intervention based on deducing the original Q using the target answer A and the context provided by task C. Hence, we define the general problem $\mathcal{T}_b$ in §2, and the applications $\mathcal{T}_{b_B}$ (§3.2) and $\mathcal{T}_{b_H}$ (§3.1).

3.1 Hiding Approach

To elicit LLMs to retrieve the original Q by reasoning in a backward way, we propose $\mathcal{T}_{b_H} = (Q_H, O, A_H)$. We restrict the generation of Q using Q_H, i.e., Q with an hide part with a placeholder x. We replace the target 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., 2024), we mask the numerical value in the prompt with x (placeholder value). Hence, we produce the 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 §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.

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 the entire Q, i.e., Q_B, and 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. To solve this problem, we introduce $\mathcal{A}$ described in §3.3 for the Math Word Problem

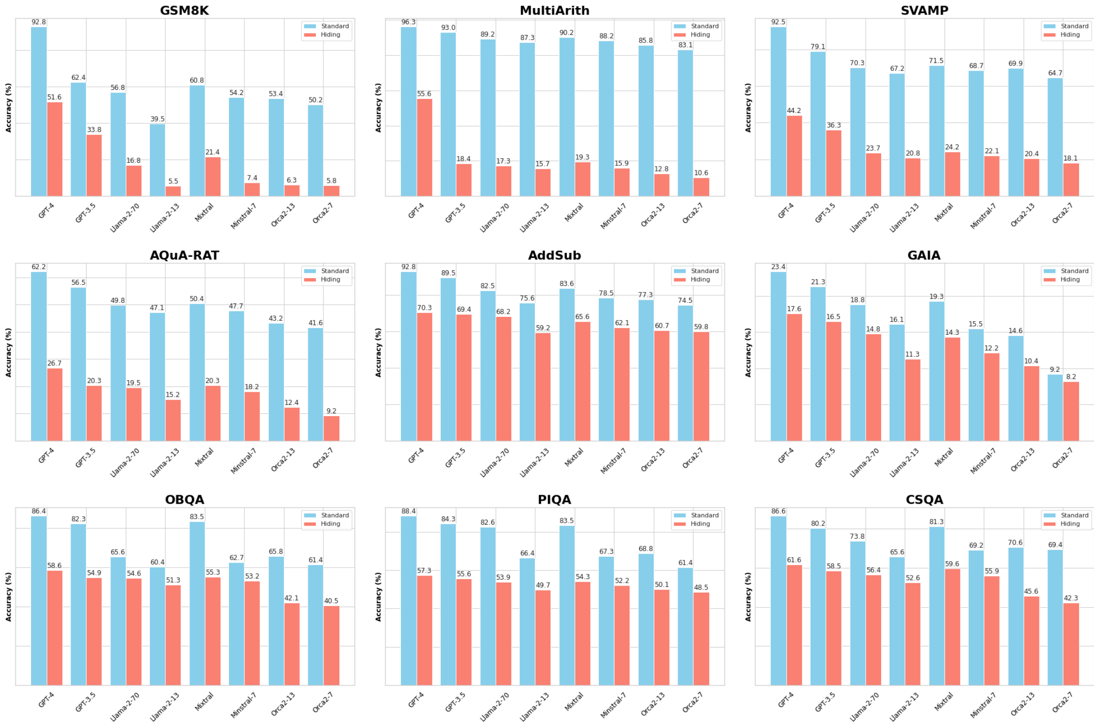

Figure 2: Accuracies (%) on Math Word Problem and Multiple Choices Questions proposed in §4.1 using `Standard` prompting approach and `Hiding` approach (§3.1).

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 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 shown in Figure 1 (all passages are detailed in Appendix B.2).

4 Experiments

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

4.1 Data

We propose our experimental setup by adapting the method proposed in §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 with this type of structure. Following Deb et al. (2024) we use GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), MultiArith (Roy and Roth, 2015); and Jiang et al. (2024) we use AddSub (Hosseini et al., 2014) AQuA (Ling et al., 2017), GAIA (Mialon et al., 2023).

QA Multiple Choices Question In contrast to the previous works, we have introduced additional tasks. These are exemplified by MCQ tasks that, unlike MWPs, have different structures. 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 reason-

Strategy	Model	GSM8K$_H$	SVAMP$_H$	M.Arith$_H$	AQua-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
Hiding (5-shot)	GPT-3.5	35.4±.3	38.4±.4	20.5±.3	70.6±.4	22.1±.3	18.6±.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
Complex-CoT (0-sh.)	GPT-3.5	40.5±.1	39.9±.1	21.7±.2	73.7±.3	24.5±.6	21.2±.4
Complex-CoT (5-sh.)	GPT-3.5	43.5±.2	41.3±.2	26.4±.2	76.6±.3	24.8±.2	26.3±.4
Paraphrasing (2-sh.)	GPT-3.5	50.2±.3	45.8±.4	36.8±.3	79.2±.4	26.7±.2	29.8±.2
	Llama2-70	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-sh.)	GPT-3.5	56.7±.1	50.3±.1	41.9±.4	**83.8**±.2	32.1±.1	**33.9**±.4
	Llama2-70	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-sh.)	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
	Llama2-70	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

Table 3: Improvements in accuracy with various prompting strategies in the `Hiding` approach.

ing, 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 §3 and detailed in Appendix B.

4.2 Models

To produce a complete analysis, we test different LLMs. We select different models by attempting to get at least two models from the same families but with differing 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 -13), two Mistral models (Jiang et al., 2023) (Mixtral and Mistral-7b) and finally two Orca2 models (Mitra et al., 2023) (Orca2-7b and -13b). For more details on the parameters, see Appendix A.

4.3 Evaluation

We evaluate the performance of the LLMs introduced in §4.1 on the tasks defined in §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 entire question can be generated correctly, even if it is delivered using different terminology. In addition, in Appendix H, we discuss an additional analysis performed with an LLM (GPT-4) as a judge.

5 Results

Large Language Models (LLMs) are able to seek hypotheses that best approximate the explanation of a set of observations; indeed, they deliver answers when elicited to consider the fact that caused the final evidence as observed in the `Blanking` experiments in Figure 3. On the other side of the coin, the same behaviour does not emerge when the nature of the task is related to a more in-depth understanding of the context in the prompt, as occurs in the `Hiding` experiments in Figure 2. The nature of the differences between the final results of the `Blanking` and `Hiding` (§6.1) approaches can be traced back to the structure of the prompt. Therefore, in §6, we analyse the role of in-context examples and how they impact the answers delivered by the models. Therefore, we show that input paraphrasing techniques might benefit, aid comprehension, and have a positive impact on the tasks analysed (§6.2). Finally, we observe that these findings also emerge when the nature of in-context demonstrations varies (`Cross-Blanking` in §6.3).

6 Analysis & Discussion

The results obtained downstream of the proposed approaches (i.e., `Blanking` and `Hiding`) reveal that LLMs are able to understand the given task and deliver reasoned answers by solving the input problem. However, an in-depth analysis of the results highlights divergences as discussed in §6.1. However, the discrepancies seem to be related to the understanding of the task. In fact, through manipulation of the prompt in specific paraphrasing of the input, different scenarios emerge, as shown in §6.2. Finally, in §6.3, we reconsider the Blanking approach by proposing the Cross-Blanking Test that stresses the LLMs' understanding abilities.

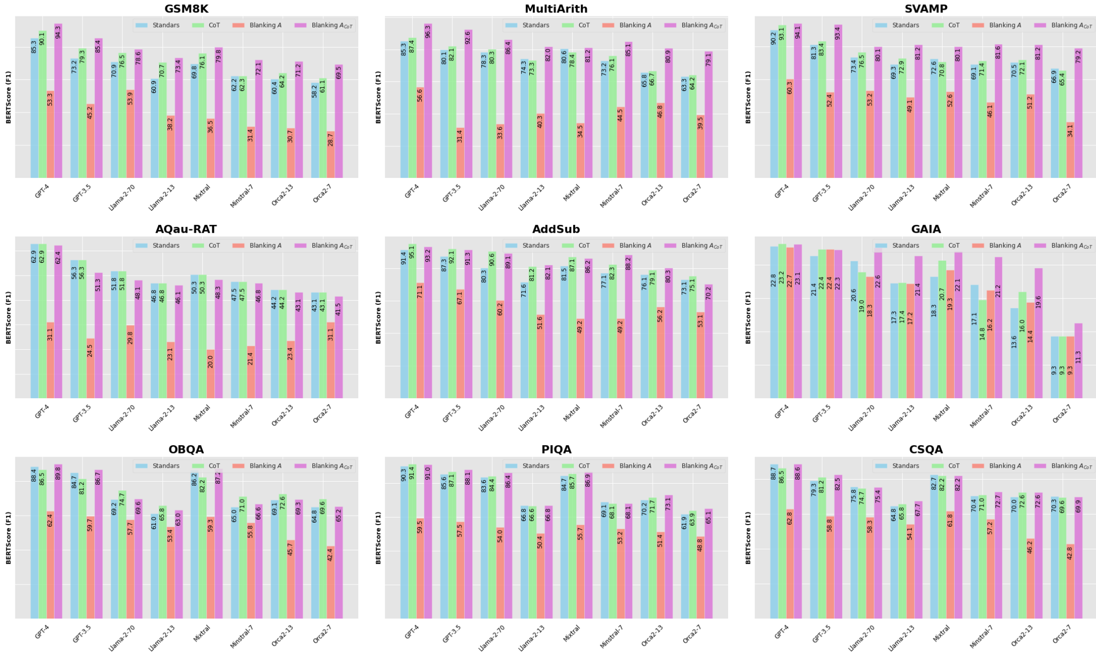

Figure 3: Performances (BERTScore F1) on Math Word Problem and Multiple Choices Questions proposed in §4.1 using Standard prompting approach (as shown in Table 1) and `Blanking` approach proposed in §3.2

6.1 Blanking & Hiding Results

Blanking 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 (§3.2) is, on average, high when the prompts are formed with A_{CoT}, i.e., answers generated via CoT (Wei et al., 2023). 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). On the other hand, the Blanking approach version constructed using the answer A as evidence does not have the same results. 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 inputs.

Although this result could be expected or mistaken as a data contamination problem, we introduce a cross-evaluation to better study the in-context comprehension abilities displayed by LLMs. To observe whether LLMs can reason in the opposite direction, we introduce *Cross-Blanking* experiment in §6.3. Specifically, we deliver as A_{CoT} the responses generated by other LLMs to perform the `Cross-Blank` evaluation as in Table 7.

Hiding LLMs fail to retrieve the hidden information in prompts. Table 3 shows the accuracies of different LLMs presented in §4.2. A difference emerges between the standard prompts, where models are prompted with a problem to generate an answer, and the Hiding approach, where the models are asked to reconstruct the hidden part of the question. A significant difference emerges because there is a smaller gap in the MCQ tasks than in MWP. This phenomenon leads to the study of the input composition, as we hypothesize that these average differences can be traced back to the present content. In the MCQ tasks, there is more context (e.g., the choices) than in MWP, where the answer is coincident.

Backward beyond standard benchmarks The performances in the tasks proposed in §4.1 can also be similarly observed on lesser-known tasks, such as reasoning in medical question-answering, as discussed in Section 7.

6.2 Prompting Approaches

Manipulation of the prompt structure 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)), that made final improvements over those discussed in §6.1. We discuss different approaches tested via GPT-3.5 and Llama-2-70 as models.

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 (§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).

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 (Tables 3). 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 prompt proposed in the Hiding approach (§3.1).

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

Self-Refine Although paraphrasing prompts support LLMs in understanding the problem, iteratively reconsidering the feedback until a predetermined condition is reached (Self-Refine) has overpowered all approaches. We notice improvements by adapting the original Self-Refine to our Hiding approach (Tables 3).

6.3 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. We have revisited the Blanking Approach from a Cross-perspective. Hence, we construct the prompts as described in §3.2, but instead of providing A_{CoT} generated by the evaluating LLM, we cross-reference the demonstrations (see Table 7 in Appendix G). We reproduce the experiments using one mathematical and one multiple-choice question task. From the results in Table 4 it emerges 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.

6.4 Metrics Error Analysis & Limitations

The results in §6.1 demonstrate LLMs' ability to provide answers while considering backward-facing problems. Following the various techniques used to elicit generation in different scenarios, we qualitatively analyse the results obtained and the metrics behind them, highlighting limitations and strengths.

BERTScore vs LLMs-judge In the Blanking Task (§3.2), we employ 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. In Table 10, we discuss using GPT-4 as an evaluator judge, revealing that the results do not differ.

Numerical Limitation On the side of the Hiding approach, we consider the responses generated by different LLMs in the MWP tasks. 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 described in Appendix B.

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

7 Application and Future Work

Our contribution was to analyse the reasoning abilities of different LLMs. In particular, by proposing variants of the original tasks, we aimed to test the LLMs' understanding and generative abilities. The tests in the main contribution were conducted on nine benchmarks widely used to assess various types of LLM capabilities (mathematical, symbolic, and commonsense reasoning abilities).

Application However, the application of our findings goes beyond just benchmarking tests. Table 5 shows the application of our tests to tasks concerning medical-reasoning QA (Jin et al., 2020), where backward comprehension abilities support the choice of the final diagnosis.

8 Related Work

Question Answering Problem QA tasks are generally defined by a natural language description that can be a question in the case of Multiple Choice Questions (MCQ) or a mathematical problem (MWP) tasks (Lu et al., 2023). The description expresses the relations between various entities or quantities followed by a query. To respond to the query, one must represent the relationship between entities and quantities. The resolution of the problem requires a semantic understanding of the natural language description. Koncel-Kedziorski et al. (2015); Roy and Roth (2018) parse the description using statistical learning techniques to identify suitable models for generating answers. Behind the advent of sequence-to-sequence models (Sutskever et al., 2014), for automatic translation, the approaches for solving these tasks diverge. For MWP, Wang et al. (2017); Jie et al. (2022) propose

Model		forward	Hiding	Blanking
GPT-4	-	93.5	67.9	62.8
	CoT	96.2	75.3	**90.2**
	Paraphrasing	-	79.5	-
	Para+Self	-	**82.6**	-
GPT-3.5	-	82.3	61.8	56.6
	CoT	**86.4**	**65.4**	74.9
	Paraphrasing	-	76.1	-
	Para+Self	-	**79.7**	-
Llama-2-70	-	58.2	43.2	24.2
	CoT	**62.4**	**46.8**	**47.8**
	Paraphrasing	-	50.2	-
	Para+Self	-	**55.8**	-
Llama-2-13	-	48.2	24.6	19.6
	CoT	**47.8**	**32.3**	**36.8**
Mixtral8x7	-	51.8	36.7	20.6
	CoT	**52.6**	**38.2**	**43.2**
	Paraphrasing	-	44.3	-
	Para+Self	-	**49.8**	-
Mistral-7	-	50.2	18.6	16.8
	CoT	**49.4**	**22.3**	**46.3**
Orca2-13	-	51.8	23.8	17.1
	CoT	52.6	27.2	**44.8**
Orca2-7	-	50.2	16.8	13.4
	CoT	49.4	23.4	**42.3**

Table 5: Performances on MedQA (Jin et al., 2020) accuracies for `Hiding` and BertScoreF1 for `Blanking` approaches. Moreover, we evaluate additional strategies as in Table 3. (* we called (**Para+Self**) the approach (**Paraphrasing+Self-Refine**).

encoder-decoder frameworks that translate the natural language description of MWPs into equations. In MCQ, Banerjee et al. (2019); Abujabal et al. (2018) propose methods for retrieving or generating answers from knowledge bases.

Large Language Models (LLMs) Recently LLMs (OpenAI, 2023; Touvron et al., 2023) achieved outstanding performance in both MWPs and MCQs tasks without using external knowledge bases or additional methods. They employ the ability to create context-based instances via a few-shot iteration and prompting methods.Welleck et al. (2022); Madaan et al. (2023) use LLMs involve verifying the response provided by the LM, either using the model itself or external verifiers like compilers or proof checkers (Zheng et al., 2023; Weng et al., 2023).

Reasoning Direction We focus on a precise case of backward reasoning with a single answer (Qin et al., 2020; Thayaparan et al., 2021) consists of inferring which of the explanations is the most plausible. Previous works have mainly focused on textual reasoning under constraints. In arithmetic tasks, Weng et al. (2023) used abductive reasoning to improve the accuracy of forward reasoning by involving the backwards. Our work, on the other hand, addresses backward reasoning as an independent problem. Following foundational work, we extend the study to tasks beyond math problems and scale the tests. Our interest is analysing the inherent complexities of reasoning and creating practical solutions to deal with them.

9 Future Works

The study of LLMs' reasoning capabilities is an active area. In parallel contributions to this work, we have studied and proposed techniques for aligning reasoning capacities between models in both English (teacher-student paradigms (Ranaldi and Freitas, 2024a,b)) and multilingual settings (Ranaldi and Pucci, 2023). In the future, we would like to use abductive methods to enhance the deductive abilities of LLMs.

On the other side of the coin, we study topics relevant to data contamination (Ranaldi et al., 2024a), memorisation (Ranaldi et al., 2023; Ranaldi and Zanzotto, 2024) and effect to persuasion or better defined as sycophancy (Ranaldi and Pucci, 2024) of LLMs. In this case, one objective is to use the generations of LLMs as an analysis tool by expanding the Cross-Banking task proposed in §6.3.

10 Conclusion

This paper explores Large Language Models (LLMs) behaviour in forward and backward generative ways. By operating via two approaches (Hiding and Blanking), we challenge LLMs to infer the original question from the answers. The experiments reveal insights into the LLMs' abilities; while they show proficiency in forward reasoning, their performances in backward ways vary significantly. The Hiding approach, which partially obscures the original question, demonstrates that LLMs could reconstruct missing elements. Instead, the Blanking approach, which presents a challenging scenario by completely removing the original question, highlights the practical abilities. Our research delves into various prompting techniques to empower the LLMs' performance by eliciting the LLM to understand and challenge 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 LLM development, particularly in areas requiring complex, multi-directional reasoning abilities.

Limitations

We analysed 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.

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.

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.

Aniruddha Deb, Neeva Oza, Sarthak Singla, Dinesh Khandelwal, Dinesh Garg, and Parag Singla. 2024. Fill in the blank: Exploring and enhancing llm capabilities for backward reasoning in math word problems.

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

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models.

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.

Weisen Jiang, Han Shi, Longhui Yu, Zhengying Liu, Yu Zhang, Zhenguo Li, and James T. Kwok. 2024. Forward-backward reasoning in large language models for mathematical verification.

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.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. ArXiv, abs/2009.13081.

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.

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.

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.

Federico Ranaldi, Elena Sofia Ruzzetti, Dario Onorati, Leonardo Ranaldi, Cristina Giannone, Andrea Favalli, Raniero Romagnoli, and Fabio Massimo Zanzotto. 2024a. Investigating the impact of data contamination of large language models in text-to-SQL translation. In Findings of the Association for Computational Linguistics: ACL 2024, pages 13909–13920, Bangkok, Thailand. Association for Computational Linguistics.

Leonardo Ranaldi and Andre Freitas. 2024a. Aligning large and small language models via chain-of-thought reasoning. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1812–1827, St. Julian's, Malta. Association for Computational Linguistics.

Leonardo Ranaldi and Andre Freitas. 2024b. Self-refine instruction-tuning for aligning reasoning in language models. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 2325–2347, Miami, Florida, USA. Association for Computational Linguistics.

Leonardo Ranaldi and Giulia Pucci. 2023. Does the English matter? elicit cross-lingual abilities of large language models. In Proceedings of the 3rd Workshop on Multi-lingual Representation Learning (MRL), pages 173–183, Singapore. Association for Computational Linguistics.

Leonardo Ranaldi and Giulia Pucci. 2024. When large language models contradict humans? large language models' sycophantic behaviour.

Leonardo Ranaldi, Giulia Pucci, Barry Haddow, and Alexandra Birch. 2024b. Empowering multi-step reasoning across languages via program-aided language models. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 12171–12187, Miami, Florida, USA. Association for Computational Linguistics.

Leonardo Ranaldi, Giulia Pucci, Federico Ranaldi, Elena Sofia Ruzzetti, and Fabio Massimo Zanzotto. 2024c. A tree-of-thoughts to broaden multi-step reasoning across languages. In Findings of the Association for Computational Linguistics: NAACL 2024, pages 1229–1241, Mexico City, Mexico. Association for Computational Linguistics.

Leonardo Ranaldi, Elena Sofia Ruzzetti, and Fabio Massimo Zanzotto. 2023. PreCog: Exploring the relation between memorization and performance in pre-trained language models. In Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing, pages 961–967, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Leonardo Ranaldi and Fabio Zanzotto. 2024. HANS, are you clever? clever hans effect analysis of neural systems. In Proceedings of the 13th Joint Conference on Lexical and Computational Semantics (*SEM 2024), pages 314–325, Mexico City, Mexico. 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.

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.

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, 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.

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

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

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 §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.
- two models of the Orca2 family (Mitra et al., 2023): Orca2-7b and Orca2-13b.
- two models of the MistralAI family: Mistral-7b and Mixtral using official version on huffingface versions of the quantized to 4-bit models using GPTQ.

For all experiments performed only in inference, we use a closed-source API or the 4-bit GPTQ quantized version of the model on two 48GB NVIDIA RTX A6000 GPUs. 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 §B.1 and $\mathcal{T}_{b_B}$ in §B.2. Downstream of the generation methodologies, we filtered the original datasets by removing the examples we could not parse optimally (see Table 9).

B.1 Generation for Hiding Approach

Math Word Problems As introduced in §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 $Dataset'_k$:

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

To convert Q in Q_H and extract the numerical subparts $H_i^0, \dots, B_i^k$, 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 §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 $Dataset'_k$:

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

To convert Q in Q_H and extract the noun subparts $P_i^0, \dots, P_i^k$, 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 §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 **??**, 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 8, 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 **??**).

F GPT-4 as a Judge

In §6.3, we discuss the results obtained using BERTScore to evaluate the performances achieved by different models in the Blanking task introduced in §3.2. In this additional experiment, we replicate the Cross-Blanking test using GPT-4 as the judge. Given the original question and the question generated by the LLM under test, GPT -4 will produce a positive or negative judgment that we will define as accuracy.

Table 10 reports the accuracies obtained. Hence, we can observe no sensible differences compared to Table 4. Therefore, even though the two metrics are not directly comparable, BERTScore approximates the accuracy of a GPT-4 evaluator well in this scenario.

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:**
+ Let's think step by step (CoT Prompt)	+ Let's think step by step (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 6: 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: x	**Question:** x
Answer: $\mathcal{A}''$ or $\mathcal{A}''_{CoT}$	**Answer:** $\mathcal{A}'$ or $\mathcal{A}'_{CoT}$

Table 7: 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 8: Paraphrasis prompting.

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

H Cross-Blanking test using LLM as a judge

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 10: Performances Cross-Blanking test using GPT-4 as a judge. In this test, we elicit the models to generate the Blanked question (§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.