Large Language Models Can Be Easily Distracted by Irrelevant Context

Freda Shi [1 2 *] **Xinyun Chen** [1 *] **Kanishka Misra** [1 3] **Nathan Scales** [1] **David Dohan** [1] **Ed Chi** [1]
Nathanael Schärli [1] **Denny Zhou** [1]

Abstract

Large language models have achieved impressive performance on various natural language processing tasks. However, so far they have been evaluated primarily on benchmarks where all information in the input context is relevant for solving the task. In this work, we investigate the *distractibility* of large language models, i.e., how the model problem-solving accuracy can be influenced by irrelevant context. In particular, we introduce Grade-School Math with Irrelevant Context (GSM-IC), an arithmetic reasoning dataset with irrelevant information in the problem description. We use this benchmark to measure the distractibility of cutting-edge prompting techniques for large language models, and find that the model performance is dramatically decreased when irrelevant information is included. We also identify several approaches for mitigating this deficiency, such as decoding with self-consistency and adding to the prompt an instruction that tells the language model to ignore the irrelevant information.[1]

1. Introduction

Prompting large language models performs decently well in a variety of domains (Brown et al., 2020; Chowdhery et al., 2022, *inter alia*). However, for most of theses evaluation benchmarks, all the information provided in the problem description is relevant to the problem solution, as the problems in exams. This is different from real-world situations, where problems usually come with several pieces of contextually related information, which may or may not be relevant to the problems that we want to solve. We have to identify what information is actually necessary during solving those problems. Studies in psychology have shown that irrelevant information may significantly decrease some children and even adults problem-solving accuracy (Hoyer et al., 1979; Pasolunghi et al., 1999; Marzocchi et al., 2002, *inter alia*).

Original Problem
Jessica is six years older than Claire. In two years, Claire will be 20 years old. How old is Jessica now?
Modified Problem
Jessica is six years older than Claire. In two years, Claire will be 20 years old. *Twenty years ago, the age of Claire's father is 3 times of Jessica's age.* How old is Jessica now?
Standard Answer 24

Table 1. An example problem from GSM-IC. An irrelevant sentence (*italic and underlined*) that does not affect the standard answer is added immediately before the question.

In this work, we study the *distractibility* of large language models for various prompting techniques; i.e., how is large language model prompting affected by irrelevant context, and what strategies can be used to improve performance? To measure distractibility, we construct the GSM-IC dataset, a grade-school math problem dataset derived from GSM8K (Cobbe et al., 2021) and introduce two different metrics. In contrast to prior work that derives benchmark variations by substituting sentences of the base problems with variations (Patel et al., 2021; Kumar et al., 2021, *inter alia*), we keep the base problem description and add to it one irrelevant sentence, while making sure that it does not affect the solution of the problem (Table 1).

We use Codex (`code-davinci-002`) and GPT-3.5 (`text-davinci-003`) in the GPT3 model family to evaluate state-of-the-art prompting techniques on GSM-IC,[2] including chain-of-thought prompting (CoT; Wei et al., 2022), zero-shot chain-of-thought prompting (0-CoT; Kojima et al., 2022), least-to-most-prompting (LtM; Zhou et al., 2022), and prompting with programs (Program;

Work done while FS and KM are student researchers at Google DeepMind. *Equal contribution [1]Google DeepMind [2]Toyota Technological Institute at Chicago [3]Purdue University. Correspondence to: Freda Shi <freda@ttic.edu>, Xinyun Chen <xinyunchen@google.com>, Denny Zhou <dennyzhou@google.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]Dataset is available at `https://github.com/google-research-datasets/GSM-IC`.

[2]`http://openai.com/api/`

Chowdhery et al., 2022).We find that their performance on GSM-IC greatly decreases compared to the original GSM8K (without irrelevant context). We then investigate several approaches to mitigate this weakness, including self-consistency (Wang et al., 2022c) and adding irrelevant information to the exemplars in the prompt. In addition to demonstrating how to handle irrelevant information via exemplars, we also investigate the usage of task-specific instructions (Wei et al., 2021; Sanh et al., 2021; Ouyang et al., 2022; Suzgun et al., 2022; Chung et al., 2022), where we prepend an instruction sentence *"feel free to ignore irrelevant information in the problem description"* to the exemplars. We summarize our key findings below:

1. All investigated prompting techniques are sensitive to irrelevant information in the problem description. In particular, among the original problems that can be solved by baseline prompts with greedy decoding, no more than 18% of them can be consistently solved for all types of irrelevant information, showing that the large language model is easily distracted and produces inconsistent predictions when adding a small amount of irrelevant information to the problem description.
2. Self-consistency improves the performance of all prompting techniques on GSM-IC. In particular, the recall rate of the correct answer for GSM-IC is as high as 99.7% with 20 samples per problem, i.e., at least one of the 20 solutions result in the correct final answer, which means that using multiple samples allows the model to almost always retrieve the correct answer.
3. Adding irrelevant information to the exemplars shown in the prompt consistently boosts the performance, and the same holds for adding an instruction to ignore irrelevant context. This suggests that language models are—to some extent—able to learn to ignore irrelevant information by following examples or instructions.
4. We identify different factors of the irrelevant information that affect the model's sensitivity to irrelevant context. Our breakdown analysis shows that varying the numbers in the irrelevant information does not notably change the model performance, while the degree of lexical overlap with the original problem description matters.

Filtering out irrelevant information is essential for handling real-world tasks. Our evaluation indicates that despite the strong performance on challenging reasoning problems, state-of-the-art language models still have fundamental weaknesses in context understanding and identifying the relevant information from the input. Our findings suggest that in order to gain a more holistic understanding of the reasoning capability of language models, future work should also consider the model sensitivity to irrelevant context, in addition to solving more challenging problems.

2. Related Work

Few-shot prompting. Few-shot prompting (Brown et al., 2020; Chowdhery et al., 2022, *inter alia*) has been significantly boosted with various techniques, including generating intermediate steps (Ling et al., 2017; Cobbe et al., 2021; Nye et al., 2021; Wei et al., 2022; Suzgun et al., 2022; Shi et al., 2022b, *inter alia*), problem decomposition (Zhou et al., 2022; Drozdov et al., 2022; Dohan et al., 2022; Khot et al., 2022; Press et al., 2022, *inter alia*), generating programs (Austin et al., 2021; Chowdhery et al., 2022; Gao et al., 2022; Chen et al., 2022, *inter alia*), marginalizing intermediate steps that share the same result (Wang et al., 2022c; Shi et al., 2022a), and ensemble (Wang et al., 2022b; Drozdov et al., 2022). In addition, Kojima et al. (2022) demonstrate that appropriate hint in prompts also leads to decent performance, even without any exemplar. In this work, we examine these cutting-edge prompting techniques (Wei et al., 2022; Zhou et al., 2022; Kojima et al., 2022; Wang et al., 2022c) on our benchmark, and demonstrate that they are sensitive to irrelevant input context.

Natural language benchmarks with input perturbations. There has been a long line of work on adding input perturbations for natural language tasks, including model-agnostic input transformations (Liang et al., 2022; Ravichander et al., 2022, *inter alia*) and adversarial example generation against individual models (Jia & Liang, 2017; Shi et al., 2018; Morris et al., 2020; Wang et al., 2021). In particular, prior work has constructed arithmetic reasoning benchmarks through paraphrasing or rewriting sentences in the base problems from clean datasets (Patel et al., 2021; Kumar et al., 2021). Meanwhile, Liang et al. (2022) evaluate various large language models under several metrics, including accuracy, robustness, fairness, etc. Specifically, the input transformations in their robustness evaluation include semantics-preserving and semantics-altering perturbations, such as injecting typos and modifying sentences to change the ground-truth classification labels. In contrast the above work where the meaning of problem descriptions may be changed with perturbations, we keep all sentences in the original problem description, and introduce an irrelevant sentence that is ensured not to affect the standard answer.

Natural language benchmarks with irrelevant input context. Jia & Liang (2017) have shown that neural question answering systems are largely affected by adversarial distracting sentences, whereas follow up work (Khashabi et al., 2017; Ni et al., 2019) proposes learning strategies that mitigate the problem. Similar issues have been found for general-purpose pretrained language models, on the tasks of factual reasoning (Kassner & Schütze, 2020; Pandia & Ettinger, 2021; Misra et al., 2023; Li et al., 2022), code generation (Jones & Steinhardt, 2022), and syntactic generalization (Chaves & Richter, 2021). In particular, Li et al.

(2022) evaluated T5 (Raffel et al., 2020) and PaLM (Chowdhery et al., 2022) with few-shot prompts, and proposed knowledge-aware finetuning that finetunes the model on problems with counterfactual and irrelevant context, which strengthens the model robustness to noisy context. In our evaluation, we show that without training or finetuning, adding irrelevant context into demonstrations in the prompt also mitigates the distractibility of the underlying language model and significantly improves the model performance on our GSM-IC benchmark.

There exist some logical reasoning benchmarks that contain irrelevant content in task descriptions (Weston et al., 2015; Sinha et al., 2019; Clark et al., 2021; Han et al., 2022; Tafjord et al., 2020, *inter alia*). However, previous work largely focuses on designing models that require extra training, and prompting alone still hardly achieves the same level of performance as finetuned models for these tasks (Han et al., 2022; Creswell et al., 2022). In our work, we focus on arithmetic reasoning, where prompting techniques have achieved the state-of-the-art results, e.g., on GSM8K, while we show that adding a single irrelevant sentence into the problem description significantly degrades the performance.

Prompting with noisy ground truth. A line of work studies the model performance with incorrect prompting exemplars, i.e., the example problems are paired with wrong answers (Min et al., 2022; Kim et al., 2022). In addition, prior work has investigated the model sensitivity to other parts of the prompt, such as instruction tuning with misleading and irrelevant instructions (Webson & Pavlick, 2021) and wrong reasoning steps in the examples (Madaan & Yazdanbakhsh, 2022; Wang et al., 2022a). In particular, Madaan & Yazdanbakhsh (2022) conclude that the correctness of numbers and equations in chain-of-thought prompts does not play a key role in model performance, but using wrong entities and removing either equations or text explanation in the reasoning steps drastically hamper the performance. Different from this line of work, we always include correct answers to example problems in the prompt, and ensure that the irrelevant context added to the problem description does not change the ground truth answer. We show that the model performance significantly drops when presented with irrelevant context in problem descriptions, and different distributions of numbers and entities in the irrelevant context also lead to different levels of performance degradation.

3. The GSM-IC Dataset

In this section, we introduce the creation process of the GSM-IC dataset (§3.1) and the evaluation metrics (§3.2).

3.1. Dataset Creation

We randomly choose 1,000 problems from the GSM8K training set as a development set. To construct our base dataset, we then choose 100 problems from this development set that can be correctly solved by at least one of the prompting techniques mentioned in this paper;[3] that is, our base dataset is an "easy" subset of GSM8K (Table 2). Each base problem requires two to seven reasoning steps to solve.[4] Among the 100 base problems, 60 of them can be solved with two reasoning steps. The full dataset statistics can be found in Appendix A.

	COT	LTM	PROGRAM	0-COT
	95.0	94.0	83.0	44.0
+ SC	96.0	99.0	91.0	76.0

Table 2. Accuracy (×100) on the base 100-example dataset using `code-davinci-002`. See Table 3 for results with `text-davinci-003`.

Original Problem
Jeanne wants to ride the Ferris wheel, the roller coaster, and the bumper cars. The Ferris wheel costs 5 tickets, the roller coaster costs 4 tickets and the bumper cars cost 4 tickets. Jeanne has 5 tickets. [Irrelevant Sentence] How many more tickets should Jeanne buy?

Options for the Irrelevant Sentence Topic
In-Topic [ROLE] rides [NUMBER] kilometers to the bus station every day.
Off-Topic The shoe size of [ROLE] is [NUMBER].

Options for [ROLE]: Lexical Overlap with Original Characters?
Yes Jeanne's father, Jeanne's sister, Jeanne's neighbor…
No Ada, Jack, Mary, Tom…

Options for [NUMBER]
In-Range 5, 6, 7, 8…
Out-of-Range 100, 1000, 5000…

Figure 1. Illustration of the considered factors when creating the GSM-IC dataset. Best viewed in color.

We then generate the examples of our new dataset by adding to each base problem one sentence containing irrelevant information. We use a template-based method (Figure 1) to generate these sentences, which can be characterized by the following three factors:

- **Topic of the inserted sentence.** We write templates for both in-topic and off-topic sentences. In-topic sentences are closely related to the topic of the original problem, whereas off-topic sentences are about a different topic.
- **Role name overlap**. Most sentence templates contain some role name blanks, which can be filled with names that may or may not overlap with the role names that occur in the problem. For blank fillers that have overlap with original role names, we: (1) randomly pick a role name `A` from the original problem description and (2) create the blank fillers with template such as `A's father` and `A's sister`.

[3] We do not generate new examples or perform analysis on the test set to avoid potential tuning-on-test-set issues.

[4] The number of reasoning steps of a problem is given by the number of sentences in its standard answer (Cobbe et al., 2021).

- **Range of numbers**. Since we focus on arithmetic reasoning, most sentence templates also contain a number blank. We can choose to fill in the number blank with a number of similar or different magnitude to those in the original problem description. Concretely, for a number a, if there exists a number b in the original problem description or solution such that $\frac{1}{10} \leq \frac{a}{b} \leq 10$, we consider a as an in-range number, and otherwise an out-of-range number. Since the standard answer to GSM8K problems are all positive integers, we only consider positive integers as the number blank fillers.

We manually verify that (1) all the generated sentences are acceptable in English and that (2) adding them does not affect the standard solution of the base problem. Because the above factors are orthogonal, we generate for each base example a set of derived examples with different factor combinations. The full GSM-IC benchmark consists of 58,052 examples. More details about the dataset creation process can be found in Appendix A.

3.2. Evaluation Metrics

For a problem p, we denote its standard solution by $s(p)$, and the solution of method $\mathcal{M}$ by $\mathcal{M}(p)$. To evaluate the distractibility of $\mathcal{M}$, we consider the following two metrics:

- **Micro accuracy** $Acc_{micro}(\mathcal{M}; \mathcal{P})$ is the average accuracy of method $\mathcal{M}$ over all the test problems $\mathcal{P}$.

$$Acc_{micro}(\mathcal{M}; \mathcal{P}) = \frac{\sum_{p \in \mathcal{P}} \mathbb{1}\left[\mathcal{M}(p) = s(p)\right]}{|\mathcal{P}|}$$

  This means that the micro accuracy weighs all the individual test problems equally.

- **Macro accuracy** $Acc_{macro}(\mathcal{M}; \mathcal{B})$ is the average accuracy of method $\mathcal{M}$ over classes of test problems, where each class $\mathcal{P}(b)$ consists of the set of test examples derived from the base example $b \in \mathcal{B}$. We define $\mathcal{M}$'s prediction for a class $\mathcal{P}(b)$ to be correct if and only if $\mathcal{M}$'s prediction for all problems in this class are correct.

$$Acc_{macro}(\mathcal{M}; \mathcal{B}) = \frac{\sum_{b \in \mathcal{B}} \mathbb{1}\left[\bigwedge_{p \in \mathcal{P}(b)} \left[\mathcal{M}(p) = s(p)\right]\right]}{|\mathcal{B}|}$$

  This means that the macro accuracy is the fraction of base problems that can be consistently solved no matter what irrelevant sentence is being added.

- **Normalized accuracy** measures how a method is affected by the distractors, considering its accuracy on base problems. For a micro or macro accuracy $a_{\mathcal{M}}$ achieved by method $\mathcal{M}$, we calculate its corresponding normalized accuracy by

$$norm(a_{\mathcal{M}}; \mathcal{M}) = \frac{a_{\mathcal{M}}}{n_{\mathcal{M}}},$$

  where $n_{\mathcal{M}}$ denotes the base problem accuracy of method $\mathcal{M}$ (Table 2).

4. Investigated Solutions

In the following section, we review the investigated prompting techniques (§4.1), present the formats of our prompts (§4.2), and introduce instructed prompting (§4.3).

4.1. Base Techniques

Chain-of-thought prompting (CoT; Wei et al., 2022) is a prompting technique that guides the language models to solve a problem in a step-by-step manner. By presenting exemplars that solve the corresponding problems with intermediate reasoning steps in the prompts, CoT significantly improves the reasoning performance over direct answer prediction without such intermediate reasoning steps.

Zero-shot chain-of-thought prompting (0-CoT; Kojima et al., 2022) is a variation of CoT where the prompt does not contain any exemplar. Instead, the model is prompted directly with the problem of interest followed by the instruction "*Let's think step by step:*".

Least-to-most prompting (LtM; Zhou et al., 2022) teaches language models to (1) break down a problem into subproblems, and (2) solve those subproblems sequentially using CoT. The final answer is that to the last subproblem.

Program prompts (Program; Chowdhery et al., 2022) represent the arithmetic reasoning process as a program. Following prior work on solving GSM8K problems with code (Chowdhery et al., 2022; Gao et al., 2022; Chen et al., 2022), we include a Python program as the problem solution in the prompt, and execute the generated Python code using an external Python interpreter to obtain the final answer.

Self-consistency (SC; Wang et al., 2022c; Shi et al., 2022a) may further boost the reasoning performance by marginalizing over intermediate reasoning steps that share the same final result. In practice, SC can be implemented by (1) sampling several solutions from the large language model and (2) taking the majority vote. Note that SC is orthogonal to above techniques, and can be combined with any of them.

4.2. Prompt Design

We present some example prompts used in our experiments (Figure 2). For few-shot prompting techniques (i.e., CoT, LtM and Program), the input prompt includes exemplar problems and their solutions before the problem of interest. In order to keep simplicity and avoid over-fitting in prompt engineering, we follow Zhou et al. (2022) on exemplar creation; that is, we only use one simple exemplar for our main experiments. This exemplar is either based on the [Original Problem] or the [Problem with Irrelevant Context], which

[Original Problem] =
Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together?

[Problem with Irrelevant Context] =
Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. *Liz has 4 peaches.* How many apples do they have together?

[CoT Solution] =
A: Anna has 2 more apples than Elsa, so Anna has 2 + 5 = 7 apples. Elsa and Anna have 5 + 7 = 12 apples together. The answer is 12.

[LtM Solution] =
A: Let's break down this problem: 1. How many apples does Anna have? 2. How many apples do Elsa and Anna have together?
1. Anna has 2 more apples than Elsa. So Anna has 2 + 5 = 7 apples.
2. Elsa and Anna have 5 + 7 = 12 apples together.

[Program Solution] =
A: Let's solve the problem by a Python program:
Elsa_apples = 5
Anna_apples = 2 + Elsa_apples
Elsa_Anna_apples = Elsa_apples + Anna_apples
print(Elsa_Anna_apples)

[Problem of Interest] =
Q: Jeanne wants to ride the Ferris wheel, the roller coaster, and the bumper cars. The Ferris wheel costs 5 tickets, the roller coaster costs 4 tickets and the bumper cars cost 4 tickets. Jeanne has 5 tickets. Jeanne's neighbor rides 8 kilometers to the bus station every day. How many more tickets should Jeanne buy?

CoT Prompt

[Original Problem]
[CoT Solution]
Q: [Problem of Interest]
A:

0-CoT Prompt (No Exemplar Problem)

Q: [Problem of Interest]
A: Let's think step by step:

LtM Prompt

[Original Problem]
[LtM Solution]
Q: [Problem of Interest]
A: Let's break down this problem:

Program Prompt

[Original Problem]
[Program Solution]
Q: [Problem of Interest]
A: Let's solve the problem by a Python program:

Instructed CoT Prompt

Solve grade school math problems. Feel free to ignore irrelevant information given in the questions.

[Original Problem]
[CoT Solution]
Q: [Problem of Interest]
A:

Figure 2. Prompt formats for the investigated techniques on the right, which are constructed from building blocks on the left (best viewed in color). The [Problem with Irrelevant Context] is obtained by adding an irrelevant sentence (*italic and underlined*) to the original problem description and it can be used as an alternative to the [Original Problem] in the prompts on the right. In these prompts, identifiers highlighted and wrapped by brackets (e.g., [Problem of Interest]) are replaced by the contents of the corresponding building blocks. The prompts for all settings can be found in Appendix C.

allows us to investigate the effect of irrelevant information in the prompt exemplar. For 0-CoT, we adhere to Kojima et al. (2022) and directly present the problem of interest followed by "*A: Let's think step by step:*".

4.3. Instructed Prompting

In addition to presenting irrelevant information in the exemplars, we also investigate whether natural language instructions help language models ignore irrelevant context and become less distracted. Extending the line of work (Suzgun et al., 2022; Sanh et al., 2021; Ouyang et al., 2022) that includes a general task description before exemplars, we add the sentence *"Solve grade school math problems. Feel free to ignore irrelevant information given in the questions."* before our exemplars in the prompt (Figure 2), which explicitly *instructs* the language model to ignore irrelevant information in the problem description.

5. Experiments

Being mindful of the experiment costs, we uniformly sample 4,000 examples from the GSM-IC dataset (denoted by GSM-IC-4K)[5] for evaluation and analysis purposes throughout this paper. Unless otherwise specified, we mainly use `code-davinci-002` in our experiments, and we also evaluate `text-davinci-003` which is a model trained with RLHF to better follow instructions (Ouyang et al., 2022). For experiments without self-consistency decoding, we use greedy decoding (i.e., temperature $\tau = 0$); for self-consistency experiments that require multiple samples for a problem, we sample 20 responses with temperature $\tau = 0.7$ following Wang et al. (2022c).

5.1. Main Results on GSM-IC

We compare the performance of different prompting techniques on GSM-IC-4K (Table 3), in terms of both micro

[5] Our sampled GSM-IC-4K covers all 100 base problems.

Method	Micro Accuracy				Macro Accuracy			
	2 Steps	>2 Steps	Overall	*Norm*	2 Steps	>2 Steps	Overall	*Norm*
Prompting Exemplar w/o Irrelevant Context, `code-davinci-002`								
CoT	73.5	70.8	72.4	*76.2*	8.3	2.5	6.0	*6.3*
CoT + Inst.	79.0	76.0	77.8	*81.8*	20.0	7.0	15.0	*15.8*
0-CoT	29.0	29.1	29.0	*65.9*	1.7	0.0	1.0	*2.3*
0-CoT +Inst.	31.6	28.8	30.5	*69.3*	1.7	0.0	1.0	*2.3*
LtM	74.9	**81.5**	77.5	*82.4*	16.7	20.0	18.0	*19.1*
LtM + Inst.	**80.1**	81.3	**80.6**	***85.7***	**18.3**	**35.0**	**25.0**	***26.6***
Program	59.1	47.4	54.4	*65.5*	6.7	2.5	5.0	*6.0*
Program + Inst.	60.6	50.9	56.7	*68.3*	6.7	5.0	6.0	*7.2*
CoT + SC	87.6	90.1	88.1	*91.8*	29.0	28.3	30.0	*31.3*
0-CoT + SC	61.6	68.4	64.3	*84.6*	0.0	2.5	1.0	*1.3*
LtM + SC	**92.4**	**94.8**	**93.4**	***94.3***	**51.6**	**35.0**	**45.0**	***45.5***
Program + SC	73.5	76.1	74.6	*82.0*	16.7	7.5	13.0	*14.3*
Prompting Exemplar w/o Irrelevant Context, `text-davinci-003`								
CoT	69.3	66.9	68.4	*85.4*	10.0	7.5	9.0	*11.3*
CoT + Inst.	72.0	70.3	71.3	*89.1*	**11.7**	**12.5**	**12.0**	***15.0***
LtM	78.0	**73.6**	76.3	*94.2*	5.0	0.0	5.0	*6.2*
LtM + Inst.	**80.5**	70.9	**76.7**	***94.7***	5.0	0.0	5.0	*6.2*
Prompting Exemplar w/ Irrelevant Context, `code-davinci-002`								
CoT	79.8	72.4	76.8	*80.8*	16.7	10.0	14.0	*14.7*
CoT + Inst.	80.5	74.4	78.1	*82.2*	20.0	12.0	17.0	*17.9*
LtM	78.1	84.6	80.7	*85.9*	**23.3**	**35.0**	**28.0**	***29.8***
LtM + Inst.	**81.0**	**85.4**	**82.8**	***88.1***	**23.3**	**35.0**	**28.0**	***29.8***
Program	67.0	55.0	62.2	*74.9*	11.7	5.0	9.0	*10.8*
Program + Inst.	68.8	54.8	63.2	*76.1*	15.0	7.5	12.0	*14.5*

Table 3. Micro and macro accuracies ($\times 100$) on the GSM-IC-4K dataset. SC denotes self-consistency. *Norm* is the overall accuracy normalized by the fraction of solved base problems (Table 2), which is a measure for robustness w.r.t. irrelevant information. For `text-davinci-003`, the base problem accuracy with CoT is 80.0, and the base problem accuracy with LtM is 81.0. The best numbers in each column for each section (i.e., whether using `code-davinci-002` or `text-davinci-003`, whether using exemplar with irrelevant context or not, and whether using self-consistency or not) are in **boldface**.

and macro accuracies, as well as their corresponding normalized accuracies. Overall, we observe significant performance drop for both models with all prompting techniques. The drop on macro accuracy is especially large, showing that fewer than 30% of the base problems are consistently solved after adding distractors. Comparing the results of two models, `text-davinci-003` achieves better normalized micro accuracy than `code-davinci-002`, though its macro accuracy is mostly worse. In Figure 3, we present a GSM-IC-4K example where a single irrelevant sentence causes different types of errors in investigated prompting techniques. One common error type is wrongly using the number in the irrelevant sentence, as shown in the LtM prediction and other examples in Appendix B. Even if the model does not directly use the irrelevant number for numerical calculation, the presence of the irrelevant sentence in the reasoning steps alone can still cause a wrong prediction, as shown in the CoT prediction.

LtM is generally the most robust technique to irrelevant context. In terms of micro accuracy, LtM outperforms all other prompting methods across models. Using `code-davinci-002`, LtM achieves about double macro accuracy of CoT. Interestingly, with `text-davinci-003`, despite that LtM outperforms CoT on the micro accuracy, its macro accuracy is lower. Specifically, `text-davinci-003` is highly susceptible to irrelevant context with role overlap; e.g., such irrelevant sentences decrease the macro accuracy to 0 on problems with more than 2 reasoning steps. See Table 4 for the breakdown performance on different types of irrelevant context.

Selecting exemplars with distractors mitigates the distractibility. For few-shot prompts, we find that using exemplars with distractors (i.e., including problems with irrelevant context) consistently outperforms using the original exemplars without distractors across prompting techniques. While prior work has shown that training and fine-tuning with different types of problems improves model robustness (Li et al., 2022), our results show that prompting with exemplars that demonstrate how to ignore irrelevant context also results in significant robustness improvement. In Table 5, we further show that using exemplars with distractors does not cause a performance drop on the original GSM8K dataset,

Modified Problem

Maria buys a large bar of French soap that lasts her for 2 months. She spends $8.00 per bar of soap. *Every 10 months, Maria's neighbor buys a new shampoo and moisturizer for Maria's neighbor.* If Maria wants to stock up for the entire year, how much will she spend on soap?

Correct CoT Output to the Original Problem

Maria spends $8.00 per bar of soap. She buys a bar of soap every 2 months. There are 12 months in a year. Maria will spend $8.00 x 12 / 2 = $48.00 on soap for the year. The answer is $48.00.

Output to the Modified Problem

CoT

Maria buys a large bar of French soap that lasts her for 2 months. She spends $8.00 per bar of soap. Every 10 months, Maria's neighbor buys a new shampoo and moisturizer for Maria's neighbor. If Maria wants to stock up for the entire year, she will spend $8.00 x 12 = $96.00 on soap. The answer is $96.00.

LtM

Let's break down this problem: 1. How much does Maria spend on soap every 10 months? 2. How much does Maria spend on soap every year?
1. Maria spends $8.00 per bar of soap. She buys a new bar of soap every 10 months. So Maria spends $8.00 * 10 = $80.00 every 10 months.
2. Maria spends $80.00 every 10 months. So Maria spends $80.00 * 12 = $960.00 every year.
The answer is $960.00.

Program

```
Soap_price = 8.00
Soap_per_year = 2 * 12
Soap_total = Soap_price * Soap_per_year
print(Soap_total)
```

Output: 192.0

Figure 3. Example problem and corresponding outputs by different prompting techniques (best viewed in color). The CoT answer to the original problem is highlighted in green. The added irrelevant sentence is in *italic and highlighted in red*, which causes different errors (highlighted in yellow) for all prompting techniques. More examples of model predictions can be found in Appendix B.

indicating that such a prompt design can be beneficial in achieving better accuracy and robustness simultaneously.

Self-consistency significantly reduces the distractibility. Taking the majority vote from 20 samples,[6] SC improves the overall micro accuracy by more than 11 percentage points. This means that in addition to improving model performance on clean arithmetic reasoning tasks (Wang et al., 2022c), SC also substantially reduces the distractibility of large language models to irrelevant context. The gain on micro accuracy is notably large on 0-CoT (35.5 percentage points). Furthermore, the correct answer for 99.7% of the problems is in the 20 sampled answers for both CoT and LtM. Even for 0-CoT, the recall of correct solutions within 20 samples is 96.5%. Despite these improvements, the best macro accuracy among all prompting techniques is only 45%, suggesting that for more than half of the base problems, SC fails to prevent the model from being distracted by different variants of irrelevant information. These results imply that a better algorithm may be developed to further reduce the distractibility based on a few sampled solutions.

[6] If there is a tie, we take a random top-tier result for evaluation, following Wang et al. (2022c) and Shi et al. (2022a).

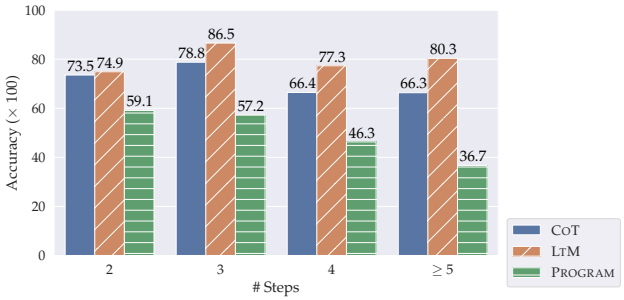

Figure 4. Micro accuracies on GSM-IC-4K with respect to the number of required reasoning steps.

5.2. Break-Down Analysis

5.2.1. Factors of the Irrelevant Context

We analyze the performance of CoT, LtM and Program with respect to the considered factors (§3.1) of the irrelevant sentences (Table 4). For both models, we find that (1) in-topic sentences with (2) role name overlap and (3) in-range numbers are generally more challenging, which is examplified by Figure 3. For LtM, the latter two factors do not have a large effect on the micro accuracy. The difference is more significant for the macro accuracy and, as an anomaly, using distractors with in-range numbers turns out to be less challenging than out-of-range numbers when using irrelevant context in the exemplar. Again, with `code-davinci-002`, LtM outperforms CoT and Program on all investigated sub-categories. On the other hand, using `text-davinci-003`, LtM outperforms CoT in terms of the micro accuracy, but the macro accuracy is much lower on all sub-categories.

5.2.2. Break-Down Accuracies w.r.t. # Steps

We analyze the break-down accuracies for problems with respect to the reasoning steps (Figure 4). While we see a significant drop for CoT and Program on problems that require four or more steps in the reasoning process, the performance of LtM is fairly consistent across difficulty. In addition to the advantage of LtM on clean problems for complicated reasoning (Zhou et al., 2022), our results show that LtM is also less sensitive to irrelevant context for complicated problems that require more steps to solve.

5.3. Instructed Prompting Improves Robustness to Irrelevant Context

We have shown that using exemplars with distractors improves robustness to irrelevant context. We also compare the performance of instructed prompting and that of the prompts without instructions in Table 3. Adding instructions to CoT, LtM, and Program consistently improves their performance. Surprisingly, instructed prompting with

Method	Micro Accuracy						Macro Accuracy					
	Topic		Role Overlap		Num. Range		Topic		Role Overlap		Num. Range	
	In	Off	Yes	No	In	Out	In	Off	Yes	No	In	Out
Prompting Exemplar w/o Irrelevant Context (`code-davinci-002`)												
CoT	63.1	80.7	68.3	76.6	70.2	74.6	10.2	33.0	10.3	22.2	11.0	19.0
LtM	**70.8**	**83.4**	**77.0**	**78.2**	**77.2**	**77.8**	**23.5**	**45.0**	**25.8**	**35.4**	**27.0**	**29.0**
Program	44.1	63.5	50.7	58.4	54.3	54.5	4.1	24.0	9.3	16.2	7.0	11.0
Prompting Exemplar w/o Irrelevant Context (`text-davinci-003`)												
CoT	63.3	72.9	68.7	68.1	67.2	69.6	**16.3**	**36.0**	**17.5**	**20.2**	**19.0**	**22.0**
LtM	**75.4**	**76.9**	**75.6**	**76.8**	**75.3**	**77.2**	6.1	7.0	6.2	9.1	6.0	6.0
Prompting Exemplar w/ Irrelevant Context (`code-davinci-002`)												
CoT	70.2	82.7	73.6	80.2	76.1	77.7	18.4	43.0	21.6	32.3	22.0	26.0
LtM	**73.0**	**87.5**	**81.4**	**80.2**	**80.0**	**81.4**	**28.6**	**58.0**	**37.1**	**42.4**	**41.0**	**35.0**
Program	52.9	70.5	60.2	64.5	61.5	62.8	10.2	37.0	14.4	23.2	15.0	17.0

Table 4. Breakdown accuracies ($\times 100$) w.r.t. the factors of the added irrelevant sentence. Lower accuracy indicates the model is more fragile to the corresponding type of irrelevant contexts. Note that the macro average accuracies are higher than the corresponding ones reported in Table 3, as we only include a subset of created problems (i.e., those corresponding to the appropriate factor) here to compute the metric. The best result in each column is in **boldface**.

original exemplars reaches comparable or even better performance than uninstructed prompting that uses exemplars with distractors for both CoT and LtM. Note that adding the instruction *"Solve grade school math problems."* alone does not significantly improve the performance, and it is the instruction *"Feel free to ignore irrelevant information given in the questions."* that makes the difference. Similar to the instruction *"Let's think step by step."* employed by 0-CoT, this shows that language models are—to some extent—able to follow natural language instructions in a way that dramatically changes their problem solving behavior, suggesting that such instructions may be useful for guiding the behavior of language models on more tasks.

On the original GSM8K development set (Cobbe et al., 2021; Zhou et al., 2022), we do not observe a drop in accuracy when using exemplars with irrelevant information, adding natural language instructions, or both (Table 5). The same holds for SVAMP (Patel et al., 2021), an arithmetic reasoning benchmark constructed by applying different types of variations to math problems from existing clean datasets, e.g., changing sentence structures, asking different questions with the same information, etc. This is impressive because the results on GSM-IC show that prompt exemplars with irrelevant information and instructed prompting both improve robustness. For the Program prompt, we find that using exemplars with distractors even increases performance on SVAMP.

Method	Exemplar w/ IrrCtx?	Accuracy GSM8K Dev.	Accuracy SVAMP Test
CoT	✓	59.3	79.1
	✗	<u>60.3</u>	<u>77.6</u>
CoT + Inst.	✓	59.3	79.1
	✗	58.8	78.7
LtM	✓	61.9	76.9
	✗	<u>59.8</u>	<u>76.6</u>
LtM + Inst.	✓	60.9	76.2
	✗	60.3	76.3
Program	✓	58.6	80.0
	✗	<u>59.8</u>	<u>77.3</u>
Program + Inst.	✓	59.2	77.9
	✗	61.1	77.8

Table 5. Accuracies ($\times 100$) on the GSM8K development set and the SVAMP test set. IrrCtx denotes irrelevant contexts, and +Inst. denotes instructed prompting. The baseline results (i.e., those using the simplest exemplars without irrelevant context and without instructions) are <u>underlined</u>.

5.4. Complicated Prompts May Hurt the Robustness to Irrelevant Context

We compare our 1-exemplar CoT prompt (Figure 2) to a 4-exemplar prompt (Appendix D of Zhou et al., 2022), which is reported as the best-performing CoT prompt on GSM8K, on GSM-IC (Table 6). Note that the 1-exemplar CoT prompt only includes a problem with a 2-step solution, while the 4-exemplar prompt includes problems that require more reasoning steps. While the 4-exemplar prompt leads to better performance on the original GSM8K development set, the 4-exemplar prompt is surprisingly more susceptible to the distraction provided by the irrelevant context. In particular, the 4-exemplar prompt is consistently worse than the 1-exemplar prompt on problems with more than 2 intermediate steps. Even for 2-step prompts, the accuracy

Method	#Prompting Exemplars	GSM8K Dev.	GSM-IC-4K 2 Steps	GSM-IC-4K > 2 Steps
CoT	1	60.3	73.6	70.8
	4	66.3	78.0	69.4
CoT + Inst.	1	58.8	79.0	**76.0**
	4	**66.5**	**79.2**	70.6

Table 6. Micro accuracies (×100) on the GSM8K development set and GSM-IC-4K. # Prompting exemplars denotes the number of exemplars used in the prompt. The best number in each column is in **boldface**.

Method	code-davinci-002	text-davinci-003
CoT	67.4	68.2
CoT + Inst.	68.9	69.9
LtM	73.4	70.2
LtM + Inst.	**74.4**	**72.8**

Table 7. Accuracies (×100) on the football split of DROP (Dua et al., 2019) benchmark.

improvement from adding more exemplars is almost negligible when using instructions (79.0 vs 79.2). Overall, this finding indicates that adding more exemplars can make the prompt less robust as it leads to some overfitting.

5.5. Extension to DROP

In addition to GSM-IC, we extend our evaluation to the DROP dataset (Dua et al., 2019), where the task is to answer a question according to a long passage that naturally contains irrelevant context. We show an example about football games in Table 8.

We use the CoT and LtM prompts in (Zhou et al., 2022) as the baselines, and we evaluate the prompt variants with the instruction *"Solve following questions. Feel free to ignore irrelevant information given in the questions."* added before the exemplars. Note that by adding a problem reduction step in the exemplar solution, the least-to-most prompt implicitly leads the model to come up with relevant subproblems to solve the given problem. Again, we observe that the instruction consistently improves the performance of both CoT and LtM prompting (Table 7).

Paragraph
The Seahawks played the San Francisco 49ers. In the first quarter, the Hawks RB Julius Jones got a 27-yard TD run, along with DT Craig Terrill returning a fumble 9 yards for a touchdown. In the third quarter, the 49ers almost rallied as RB H. J. Torres made a 12-yard TD pass to Lucas Nelly, along with Mare kicking a 32-yard field goal. In the final quarter, Julius Jones got another 11-yard TD.
Question
How many yards do the shortest touchdown run and the longest touchdown pass combine for?
LtM solution
To answer the question "How many yards do the shortest touchdown run and the longest touchdown pass combine for?", we need to know: "How many yards was the shortest touchdown run?", "How many yards was the longest touchdown pass?".
Q: How many yards was the shortest touchdown run?
A: All the touchdown runs are: a 27-yard touchdown run, a 9-yard touchdown run, a 11-yard touchdown run. The smallest number between 27, 9, 11 is 9. The shortest touchdown run was 9 yards. The answer is 9.
Q: How many yards was the longest touchdown pass?
A: All the touchdown passes are: a 12-yard touchdown pass. The longest touchdown pass was 12 yards. The answer is 12.
Q: How many yards do the shortest touchdown run and the longest touchdown pass combine for?
A: We know that the shortest touchdown run was 9-yard. We also know that the longest touchdown pass was 12-yard. 9 + 12 = 21. The answer is 21.

Table 8. A DROP example about football games.

6. Conclusion and Discussion

In this work, we introduce GSM-IC, a dataset that supports comprehensive study of the distractibility of large language models when performing arithmetic reasoning in presence of irrelevant contexts. We examine a variety of prompting techniques on GSM-IC, and demonstrate that they are all sensitive to the irrelevant information in the problems. Among the studied techniques, self-consistency (Wang et al., 2022c) leads to a substantial improvement in robustness to irrelevant context across the board, and presenting example problems with irrelevant context in the prompt also consistently improves the performance. Similarly, we find that simply adding an instruction to ignore irrelevant information brings notable performance gains on our benchmark.

Despite the improvement achieved by these methods, the fundamental issue remains: a single piece of irrelevant information can distract the models and substantially degrade their performance, even on problems whose clean versions they correctly solve. We encourage researchers to also prioritize improving on this fundamental limitation when developing new training and prompting techniques. We leave further investigation on the distractibility for other tasks and different language models for future work.

Acknowledgement

We would like to thank Dale Schuurmans, Olivier Bousquet and Jack Nikodem for helpful discussion and feedback.

References

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *NeurIPS*, 2020.

Chaves, R. P. and Richter, S. N. Look at that! BERT can be easily distracted from paying attention to morphosyntax. In *Proceedings of the Society for Computation in Linguistics 2021*, pp. 28–38, Online, February 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.scil-1.3.

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Clark, P., Tafjord, O., and Richardson, K. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 3882–3890, 2021.

Cobbe, K., Kosaraju, V., Bavarian, M., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/pdf/2110.14168.

Creswell, A., Shanahan, M., and Higgins, I. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

Dohan, D., Xu, W., Lewkowycz, A., Austin, J., Bieber, D., Lopes, R. G., Wu, Y., Michalewski, H., Saurous, R. A., Sohl-Dickstein, J., Murphy, K., and Sutton, C. Language model cascades. *arXiv preprint arXiv:2207.10342*, 2022.

Drozdov, A., Schärli, N., Akyürek, E., Scales, N., Song, X., Chen, X., Bousquet, O., and Zhou, D. Compositional semantic parsing with large language models. *arXiv preprint arXiv:2209.15003*, 2022.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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)*, 2019.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.

Han, S., Schoelkopf, H., Zhao, Y., Qi, Z., Riddell, M., Benson, L., Sun, L., Zubova, E., Qiao, Y., Burtell, M., et al. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.

Hoyer, W. J., Rebok, G. W., and Sved, S. M. Effects of varying irrelevant information on adult age differences in problem solving. *Journal of gerontology*, 34(4):553–560, 1979.

Jia, R. and Liang, P. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

Jones, E. and Steinhardt, J. Capturing failures of large language models via human cognitive biases. *arXiv preprint arXiv:2202.12299*, 2022.

Kassner, N. and Schütze, H. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7811–7818, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.698. URL https://aclanthology.org/2020.acl-main.698.

Khashabi, D., Khot, T., Sabharwal, A., and Roth, D. Learning what is essential in questions. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 80–89, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1010. URL https://aclanthology.org/K17-1010.

Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., and Sabharwal, A. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

Kim, J., Kim, H. J., Cho, H., Jo, H., Lee, S.-W., Lee, S.-g., Yoo, K. M., and Kim, T. Ground-truth labels matter: A deeper look into input-label demonstrations. *arXiv preprint arXiv:2205.12685*, 2022.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

Kumar, V., Maheshwary, R., and Pudi, V. Adversarial examples for evaluating math word problem solvers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2705–2712, 2021.

Li, D., Rawat, A. S., Zaheer, M., Wang, X., Lukasik, M., Veit, A., Yu, F., and Kumar, S. Large language models with controllable working memory. *arXiv preprint arXiv:2211.05110*, 2022.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. 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)*, pp. 158–167, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1015. URL https://aclanthology.org/P17-1015.

Madaan, A. and Yazdanbakhsh, A. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*, 2022.

Marzocchi, G. M., Lucangeli, D., De Meo, T., Fini, F., and Cornoldi, C. The disturbing effect of irrelevant information on arithmetic problem solving in inattentive children. *Developmental neuropsychology*, 21(1):73–92, 2002.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

Misra, K., Rayz, J., and Ettinger, A. COMPS: Conceptual minimal pair sentences for testing robust property knowledge and its inheritance in pre-trained language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023.

Morris, J. X., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., and Qi, Y. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*, 2020.

Ni, J., Zhu, C., Chen, W., and McAuley, J. Learning to attend on essential terms: An enhanced retriever-reader model for open-domain question answering. 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)*, pp. 335–344, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1030. URL https://aclanthology.org/N19-1030.

Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., Sutton, C., and Odena, A. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. URL https://arxiv.org/abs/2203.02155.

Pandia, L. and Ettinger, A. Sorting through the noise: Testing robustness of information processing in pre-trained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1583–1596, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.119. URL https://aclanthology.org/2021.emnlp-main.119.

Pasolunghi, M. C., Cornoldi, C., and De Liberto, S. Working memory and intrusions of irrelevant information in a group of specific poor problem solvers. *Memory & Cognition*, 27:779–790, 1999.

Patel, A., Bhattamishra, S., and Goyal, N. Are nlp models really able to solve simple math word problems? In *NAACL-HLT*, 2021. URL https://aclanthology.org/2021.naacl-main.168.pdf.

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., and Lewis, M. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Ravichander, A., Gardner, M., and Marasović, A. Condaqa: A contrastive reading comprehension dataset for reasoning about negation. *arXiv preprint arXiv:2211.00295*, 2022.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Shi, F., Fried, D., Ghazvininejad, M., Zettlemoyer, L., and Wang, S. I. Natural language to code translation with execution. In *EMNLP*, 2022a.

Shi, F., Suzgun, M., Freitag, M., Wang, X., Srivats, S., Vosoughi, S., Chung, H. W., Tay, Y., Ruder, S., Zhou, D., Das, D., and Wei, J. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022b. URL https://arxiv.org/pdf/2210.03057.

Shi, H., Mao, J., Xiao, T., Jiang, Y., and Sun, J. Learning visually-grounded semantics from contrastive adversarial samples. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 3715–3727, 2018.

Sinha, K., Sodhani, S., Dong, J., Pineau, J., and Hamilton, W. L. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4506–4515, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1458. URL https://aclanthology.org/D19-1458.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., and Wei, J. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Tafjord, O., Mishra, B. D., and Clark, P. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*, 2020.

Wang, B., Xu, C., Wang, S., Gan, Z., Cheng, Y., Gao, J., Awadallah, A. H., and Li, B. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*, 2021.

Wang, B., Min, S., Deng, X., Shen, J., Wu, Y., Zettlemoyer, L., and Sun, H. Towards understanding chain-of-thought prompting: An empirical study of what matters. *arXiv preprint arXiv:2212.10001*, 2022a.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., and Zhou, D. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747*, 2022b.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022c.

Webson, A. and Pavlick, E. Do prompt-based models really understand the meaning of their prompts? *arXiv preprint arXiv:2109.01247*, 2021.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022. URL https://openreview.net/pdf?id=_VjQlMeSB_J.

Weston, J., Bordes, A., Chopra, S., Rush, A. M., Van Merriënboer, B., Joulin, A., and Mikolov, T. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Bousquet, O., Le, Q., and Chi, E. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022. URL https://arxiv.org/pdf/2205.10625.

A. GSM-IC Details

Each of the 100 base problem require two to seven steps to solve (Figure 5).

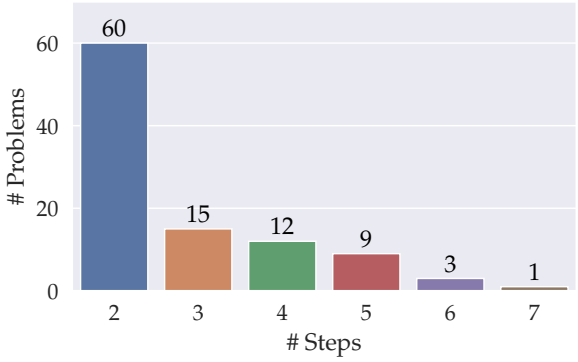

Figure 5. Base problem distribution of GSM-IC with respect to the number of reasoning steps in the ground truth problem solution.

Starting from the base problems, we follow the protocols below to create GSM-IC (§3.1).

1. **Irrelevant sentence template**.
   (a) For in-topic sentences, we manually write templates within the topic that is close to the original problem description. We are particularly careful about the shareable stuff, for example, money is sometimes considered shareable between family members. In such cases, we make sure that the added do not change the amount of shareable stuff to ensure that the final standard answer is not affected.
   (b) For off-topic sentences, we use general templates (Table 9) for all problems unless some of them can be considered as in-topic sentences for some problems—for example, the sentence "*The height of* {*role*} *is* {*number*} *feet.*" is considered as an in-topic sentence for problems about heights of people.

The shoe size of `[ROLE]` is `[NUMBER]`.
`[ROLE]` is `[NUMBER]` years old.
The height of `[ROLE]` is `[NUMBER]` feet.
`[ROLE]` bought `[NUMBER]` tomatoes from the grocery store.
`[ROLE]` has read `[NUMBER]` books in the past year.

Table 9. Off-topic sentence templates for GSM-IC.

   (c) We make sure that all sentences derived by each template are grammatical English sentences.
   (d) We write four in-topic and choose four off-topic distractor sentence templates for each problem.

2. **Blank fillers: role names**.
   (a) We randomly choose a role name `X`, and use `X's father`, `X's mother`, `X's brother`, `X's sister` and `X's neighbor` as the overlapped role names.
   (b) We choose from the name set {`Ada`, `David`, `Emma`, `Jack`, `John`, `Mary`, `Max`, `Tom`} for non-overlapped role names.
   (c) We write five names that have overlap with the original character, and five names that do not have overlap for each problem.

3. **Blank fillers: numbers**.
   (a) For in-range numbers, we randomly sample positive integers in the range of $[\frac{\ell}{10}, 10r]$, where ℓ and r denote the smallest and the largest number that appear in the problem description and standard solution, respectively.

Original Problem	Kim plants 80 cherry pits. 25% of them sprout and Kim sells 6 of the saplings. How many cherry saplings does she have left?
Added Sentence	Kim's mother plants 20 more potatoes.
Ambiguous Problem	Kim plants 80 cherry pits. 25% of them sprout and Kim sells 6 of the saplings. Kim's mother plants 20 more potatoes. How many cherry saplings does she have left?
Fixed Problem	Kim plants 80 cherry pits. 25% of them sprout and Kim sells 6 of the saplings. Kim's mother plants 20 more potatoes. How many cherry saplings does Kim have left?

Table 10. An example that adding irrelevant contexts causes ambiguity: after adding the sentence, it is unclear whether she refers to Kim or Kim's mother. To ensure that the standard answer is unchanged, we modify the final question to make it clear and faithful to the original problem.

(b) For out-of-range numbers, we choose from the range of $[2, +\infty)\backslash[\frac{\ell}{10}, 10r]$. For very few problems that ℓ is relatively large (i.e., $\ell > 10^5$) where we choose out-of-range numbers from the range of $[2, \frac{\ell}{10}]$; for other problems we choose out-of-range numbers $n = a \times 10^b$ from the range $[10r, \infty)$, where a and b are both non-negative integers.

(c) We write four in-range numbers and four out-of-range numbers for each problem.

4. Finally, if adding the irrelevant sentence causes ambiguity (e.g., Table 10), we fix the question to ensure that the standard solution to the generated problem remain the same as the base problem.

B. Sample Predictions on GSM-IC

In addition to the example outputs shown in Figure 3, we include more example problems and the predictions by different techniques (Tables 11 and 12).

C. Full prompts in experiments

We list the prompts for all experiments in Tables 13 and 14.

Method	Original GSM8K Problem	Modified Problem (w/ Irrelevant Sentence)
	Jessica is six years older than Claire. In two years, Claire will be 20 years old. How old is Jessica now?	Jessica is six years older than Claire. In two years, Claire will be 20 years old. Twenty years ago, the age of Claire's father is 3 times of Jessica's age. How old is Jessica now?
CoT	In two years, Claire will be 20 years old. So, Claire is 18 years old now. Jessica is six years older than Claire, so Jessica is 18 + 6 = 24 years old now. The answer is 24.	Jessica is six years older than Claire, so Jessica is 20 - 6 = 14 years old. Twenty years ago, the age of Claire's father is 3 times of Jessica's age, so 20 years ago, the age of Claire's father is 3 * 14 = 42 years old. The answer is 14.
LtM	Let's break down this problem: 1. How old is Claire now? 2. How old is Jessica now? 1. In two years, Claire will be 20 years old. So Claire is 20 - 2 = 18 years old now. 2. Jessica is six years older than Claire. So Jessica is 18 + 6 = 24 years old now. The answer is 24.	A: Let's break down this problem: 1. How old is Claire now? 2. How old is Jessica now? 1. In two years, Claire will be 20 years old. So Claire is 18 years old now. 2. Twenty years ago, the age of Claire's father is 3 times of Jessica's age. So 20 years ago, Claire's father is 3 * Jessica's age. So Claire's father is 3 * Jessica's age now. So Claire's father is 3 * Jessica's age - 20 years old now. So Claire's father is 3 * Jessica's age - 20 + 18 = 3 * Jessica's age - 2 years old now. So Claire's father is 3 * Jessica's age - 2 + 6 = 3 * Jessica's age + 4 years old now. So Claire's father is 3 * Jessica's age + 4 - 18 = 3 * Jessica's age - 14 years old now. So Claire's father is 3 * Jessica's age - 14 + 20 = 3 * Jessica's age + 6 years old now. So 3 * Jessica's age + 6 = 3 * Jessica's age - 2 + 6 = 3 * Jessica's age + 4. So 3 * Jessica's age + 6 = 3 * Jessica's age + 4. So 6 = 4. So Jessica's age = 2. So Jessica is 2 years old now. The answer is 2.
Program	Claire_age = 20 - 2 Jessica_age = Claire_age + 6 print(Jessica_age)	Claire_age = 20 - 2 Jessica_age = Claire_age - 6 Claire_father_age = 3 * Jessica_age print(Jessica_age)

Table 11. Output example #1 for different prompting techniques, on a GSM8K problem and a corresponding GSM-IC problem. While all techniques give correct answers to the original problem, they output different different results for the modified GSM-IC problem.

Method	Original GSM8K Problem	Modified Problem (w/ Irrelevant Sentence)
	Lucy has $65 in the bank. She made a $15 deposit and then followed by a $4 withdrawal. What is Lucy's bank balance?	Lucy has $65 in the bank. She made a $15 deposit and then followed by a $4 withdrawal. Lucy's mother's monthly rent is $10. What is Lucy's bank balance?
CoT	Lucy has $65 in the bank. She made a $15 deposit, so she has $65 + $15 = $80. She then made a $4 withdrawal, so she has $80 - $4 = $76. The answer is $76.	Lucy has $65 in the bank. She made a $15 deposit and then followed by a $4 withdrawal. Lucy's mother's monthly rent is $10. Lucy's bank balance is $65 + $15 - $4 - $10 = $66. The answer is $66.
LtM	Let's break down this problem: 1. How much money did Lucy deposit? 2. How much money did Lucy withdraw? 3. How much money does Lucy have in the bank? 1. Lucy deposited $15. 2. Lucy withdrew $4. 3. Lucy has $65 + $15 - $4 = $76 in the bank. The answer is $76.	Let's break down this problem: 1. How much money did Lucy deposit? 2. How much money did Lucy withdraw? 3. How much money does Lucy have in the bank? 1. Lucy deposited $15. 2. Lucy withdrew $4. 3. Lucy has $65 + $15 - $4 - $10 = $66 in the bank. The answer is $66.
Program	Lucy_balance = 65 Lucy_balance = Lucy_balance + 15 Lucy_balance = Lucy_balance - 4 print(Lucy_balance)	Lucy_bank = 65 Lucy_bank = Lucy_bank + 15 Lucy_bank = Lucy_bank - 4 Lucy_bank = Lucy_bank - 10 print(Lucy_bank)

Table 12. Output example #2 for different prompting techniques, on a GSM8K problem and a corresponding GSM-IC problem. While all techniques give correct answers to the original problem, they output the same incorrect result for the modified GSM-IC problem.

Method	Example w/ IrrCtx?	Prompt
CoT	✗	Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together? A: Anna has 2 more apples than Elsa, so Anna has 2 + 5 = 7 apples. Elsa and Anna have 5 + 7 = 12 apples together. The answer is 12. Q: [Problem of Interest] A:
CoT	✓	Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. Liz has 4 peaches. How many apples do they have together? A: Anna has 2 more apples than Elsa, so Anna has 2 + 5 = 7 apples. Elsa and Anna have 5 + 7 = 12 apples together. The answer is 12. Q: [Problem of Interest] A:
LtM	✗	Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together? A: Let's break down this problem: 1. How many apples does Anna have? 2. How many apples do Elsa and Anna have together? 1. Anna has 2 more apples than Elsa. So Anna has 2 + 5 = 7 apples. 2. Elsa and Anna have 5 + 7 = 12 apples together. Q: [Problem of Interest] A: Let's break down this problem:
LtM	✓	Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. Liz has 4 peaches. How many apples do they have together? A: Let's break down this problem: 1. How many apples does Anna have? 2. How many apples do Elsa and Anna have together? 1. Anna has 2 more apples than Elsa. So Anna has 2 + 5 = 7 apples. 2. Elsa and Anna have 5 + 7 = 12 apples together. Q: [Problem of Interest] A: Let's break down this problem:
0-CoT	N/A	Q: [Problem of Interest] A: Let's think step by step:
Program	✗	Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together? A: Let's solve the problem by a Python program: Elsa_apples = 5 Anna_apples = 2 + Elsa_apples Elsa_Anna_apples = Elsa_apples + Anna_apples print(Elsa_Anna_apples) Q: [Problem of Interest] A: Let's solve the problem by a Python program:
Program	✓	Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. Liz has 4 peaches. How many apples do they have together? A: Let's solve the problem by a Python program: Elsa_apples = 5 Anna_apples = 2 + Elsa_apples Elsa_Anna_apples = Elsa_apples + Anna_apples print(Elsa_Anna_apples) Q: [Problem of Interest] A: Let's solve the problem by a Python program:

Table 13. Prompts used for all investigated techniques, without instruction. The placeholder [Problem of Interest] is substituted for each problem at the test time.

Method	Example w/ IrrCtx?	Prompt
CoT	✗	Solve grade school math problems. Feel free to ignore irrelevant information given in the questions. Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together? A: Anna has 2 more apples than Elsa, so Anna has 2 + 5 = 7 apples. Elsa and Anna have 5 + 7 = 12 apples together. The answer is 12. Q: [Problem of Interest] A:
CoT	✓	Solve grade school math problems. Feel free to ignore irrelevant information given in the questions. Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. Liz has 4 peaches. How many apples do they have together? A: Anna has 2 more apples than Elsa, so Anna has 2 + 5 = 7 apples. Elsa and Anna have 5 + 7 = 12 apples together. The answer is 12. Q: [Problem of Interest] A:
LtM	✗	Solve grade school math problems. Feel free to ignore irrelevant information given in the questions. Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together? A: Let's break down this problem: 1. How many apples does Anna have? 2. How many apples do Elsa and Anna have together? 1. Anna has 2 more apples than Elsa. So Anna has 2 + 5 = 7 apples. 2. Elsa and Anna have 5 + 7 = 12 apples together. Q: [Problem of Interest] A: Let's break down this problem:
LtM	✓	Solve grade school math problems. Feel free to ignore irrelevant information given in the questions. Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. Liz has 4 peaches. How many apples do they have together? A: Let's break down this problem: 1. How many apples does Anna have? 2. How many apples do Elsa and Anna have together? 1. Anna has 2 more apples than Elsa. So Anna has 2 + 5 = 7 apples. 2. Elsa and Anna have 5 + 7 = 12 apples together. Q: [Problem of Interest] A: Let's break down this problem:
0-CoT	N/A	Solve grade school math problems. Feel free to ignore irrelevant information given in the questions. Q: [Problem of Interest] A: Let's think step by step:
Program	✗	Solve grade school math problems. Feel free to ignore irrelevant information given in the questions. Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together? A: Let's solve the problem by a Python program: Elsa_apples = 5 Anna_apples = 2 + Elsa_apples Elsa_Anna_apples = Elsa_apples + Anna_apples print(Elsa_Anna_apples) Q: [Problem of Interest] A: Let's solve the problem by a Python program:
Program	✓	Solve grade school math problems. Feel free to ignore irrelevant information given in the questions. Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. Liz has 4 peaches. How many apples do they have together? A: Let's solve the problem by a Python program: Elsa_apples = 5 Anna_apples = 2 + Elsa_apples Elsa_Anna_apples = Elsa_apples + Anna_apples print(Elsa_Anna_apples) Q: [Problem of Interest] A: Let's solve the problem by a Python program:

Table 14. All prompts with instructions. The placeholder [Problem of Interest] is substituted for each problem at the test time.