

Abstract

Chain-of-Thought (CoT) prompting has enhanced the performance of Large Language Models (LLMs) across various reasoning tasks. However, CoT still falls short in dealing with complex math word problems, as it usually suffers from three pitfalls: semantic misunderstanding errors, calculation errors, and step-missing errors. Prior studies involve addressing the calculation errors and step-missing errors, but neglect the semantic misunderstanding errors, which is the major factor limiting the reasoning performance of LLMs. To this end, we propose a simple-yet-effective method, namely *Deeply Understanding the Problems* (DUP), to improve the LLMs’ math problem-solving ability by addressing semantic misunderstanding errors. The core of our method is to encourage the LLMs to deeply understand the problems and extract the key problem-solving information used for better reasoning. Extensive experiments on 11 diverse reasoning benchmarks show that our DUP method consistently outperforms the other counterparts by a large margin. More encouragingly, DUP achieves a new SOTA result on the GSM8K benchmark, with a zero-shot accuracy of 97.1%.

1 Introduction

Despite the impressive performance of Large Language Models (LLMs) in diverse natural language understanding and generation tasks (Brown et al., 2020; Touvron et al., 2023; OpenAI, 2023; Bai et al., 2023), they often suffer from sub-optimal reasoning abilities, which cannot be overcome solely by simply scaling up the model size (Rae et al., 2021; Wang et al., 2023b). To tackle this limitation, Wei et al. (2022b) propose a few-shot Chain-of-Thought (CoT) prompting strategy, which prompts the LLMs to mimic the given step-by-step thought process a person might employ in solving a task. Such a simple strategy can significantly improve the reasoning ability of LLMs, and thus has attracted widespread attention in recent years.

Along this research line, many works focus on designing prompting strategies to enhance LLM’s reasoning ability, such as Zero-shot CoT (Kojima et al., 2022), Tree of Thought (Gao et al., 2023) and Plan-and-Solve (PS) prompting (Wang et al., 2023a). Although achieving remarkable progress, they still fall short in dealing with complex reasoning tasks, e.g., math word problems (Cobbe et al., 2021). As stated by Wei et al. (2022b), there are three main error types in the CoT-based reason-

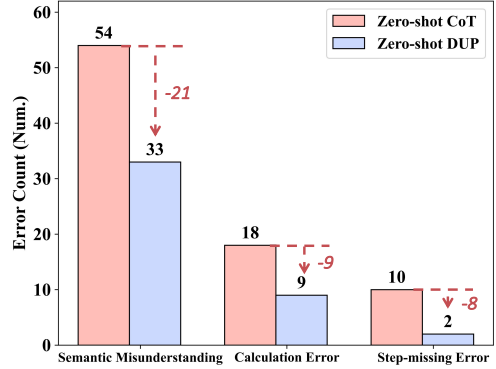


Figure 1: **Error analysis of GSM8K problems with incorrect answers returned by zero-shot CoT and our DUP using GPT-3.5 LLM.** We randomly sample 300 GSM8K problems, and follow Wei et al. (2022b) and Wang et al. (2023a) to assign the “Semantic Misunderstanding”, “Calculation Error” and “Step-missing Error” to each incorrect answer. The detailed prompts for error analysis are shown in Table 9. We see that our DUP effectively reduces the errors among all types.

ing: *semantic misunderstanding errors, calculation errors, and step-missing errors*. Specifically, semantic misunderstanding denotes the misunderstanding of the original problems or lack of commonsense concepts related to the problems. The step-missing error denotes the missing step during the calculation processes, while the calculation error denotes the operation errors. In our preliminary experiments (as shown in Figure 1), we found that CoT has major errors in semantic understanding¹, which is the main factor limiting LLMs’ reasoning performance. Prior studies (Wang et al., 2023a; Chen et al., 2023a) show that the carefully-designed prompting strategies can achieve much fewer calculation errors and step-missing errors, but still struggle to address the major semantic misunderstanding. Hence, there raises a question: *whether we can enhance the reasoning abilities of LLMs by reducing the semantic misunderstanding errors?*

Intuitively, since complex math word problems usually contain content irrelevant to solving the task, LLMs might fail to identify the core question and extract the relevant problem-solving information, thus leading to semantic misunderstanding and poor performance. This can be also proved by the findings in psychology, as prior studies (Hoyer et al., 1979; Pasolunghi et al., 1999) show that the irrelevant information may significantly decrease

¹ The discussion of the error analysis method can be found in Section 4.5.

some children’s and even adults’ problem-solving accuracy. Hence, this inspires us that, *it is crucial to enforce the LLMs to pay more attention to the core information and reduce the negative effects of irrelevant information.*

Motivated by this, we propose a simple-yet-effective method, namely *Deeply Understanding the Problems* (DUP), to improve the LLMs’ math problem-solving ability. The principle of our method is akin to the human learning process, *i.e.*, for human students who receive a complex math word problem, they will read and comprehend the text of the problem, identify the core question that needs to be answered, and finally solve it with relevant problem-solving information. Specifically, DUP consists of three stages: ❶ Revealing the core question of the input problem; ❷ Extracting the problem-solving information relevant to solving the core question; ❸ Generating and extracting the final answer by combining the core question with problem-solving information. By doing so, LLMs can filter out irrelevant information and achieve better math reasoning performance.

We conduct a series of experiments on 11 reasoning datasets across math, commonsense, and symbolic reasoning. The experimental results of GPT-3.5-Turbo (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023) show that: 1) DUP consistently outperforms the other counterparts across all datasets by a large margin; 2) Zero-shot DUP can even outperform the few-shot methods on most reasoning datasets; 3) More encouragingly, DUP achieves new SOTA results on the popular GSM8K (97.1%) and SVAMP (94.2%). To summarize, our contributions are three-fold: (1) We reveal the underlying causes of semantic misunderstanding errors, and propose a simple yet effective approach (DUP) to effectively address the semantic misunderstanding and boost LLMs’ math reasoning ability. (2) DUP is easy-to-implement and plug-and-play. It can be easily applied to various LLMs. (3) Extensive experiments show that DUP outperforms the other counterparts by a large margin, and achieves new SOTA results on GSM8K and SVAMP.

The rest of this paper is organized as follows. In Section 2, we briefly review the related works. In Section 3, we introduce our proposed DUP method in detail. Section 4 reports and analyzes our experimental results. Lastly, we conclude our study in Section 5.

2 Related Works

2.1 Reasoning with Large Language Models

Recently, we have witnessed numerous LLMs (Devlin et al., 2019; Brown et al., 2020; Chowdhery et al., 2022; Zhong et al., 2022; OpenAI, 2023; Touvron et al., 2023) that achieved tremendous success in the NLP community. Due to their emergent abilities (Wei et al., 2022a), these LLMs can achieve impressive few-shot and zero-shot performance in a variety of NLP tasks (Hendrycks et al.; Srivastava et al., 2023; Zhong et al., 2023). Specifically, as stated in (Wei et al., 2022a), with the scaling of model sizes, LLMs tend to show some emergent abilities, such as instruction-following (Wei et al.; Sanh et al.), program execution (Nye et al.), and model calibration (Kadavath et al., 2022). However, LLMs still struggle to provide stable and accurate answers when dealing with complex reasoning tasks (Zhang et al., 2023a), such as math reasoning (Cobbe et al., 2021; Patel et al., 2021; Ling et al., 2017; Hosseini et al., 2014), commonsense reasoning (Talmor et al., 2019; Geva et al., 2021) and symbolic reasoning (Wei et al., 2022b). Recent works (Yuan et al., 2023; Luo et al., 2023; Yu et al., 2023; Ho et al., 2023; Liu et al., 2023) have shown that reasoning-augmented LLMs tuning with mathematical data can relatively improve reasoning ability. However, even with such progress, these models still perform poorly in complex reasoning problems. This indicates that there is still significant room for improving the LLMs’ performance in complex reasoning tasks.

2.2 Prompting Methods

Despite the remarkable performance, the aforementioned training-based approaches usually require collecting large amounts of data and expensive computational costs, and may cause LLMs’ universal ability to decrease. Hence, some works (Wei et al., 2022b; Kojima et al., 2022) attempt to use cheaper prompting methods to strengthen the LLMs’ reasoning abilities without additional training. Wei et al. (2022b) are the first to propose the few-shot CoT prompting, which elicits a series of intermediate natural language reasoning steps before giving the final answer. So far, CoT prompting has been proven to significantly improve the reasoning capability of LLMs. Along this research line, numerous works (Zhou et al., 2023; Wang et al., 2023a; Yao et al., 2023; Zhang et al., 2023b; Chen et al., 2023b; Xu et al., 2023; Tian

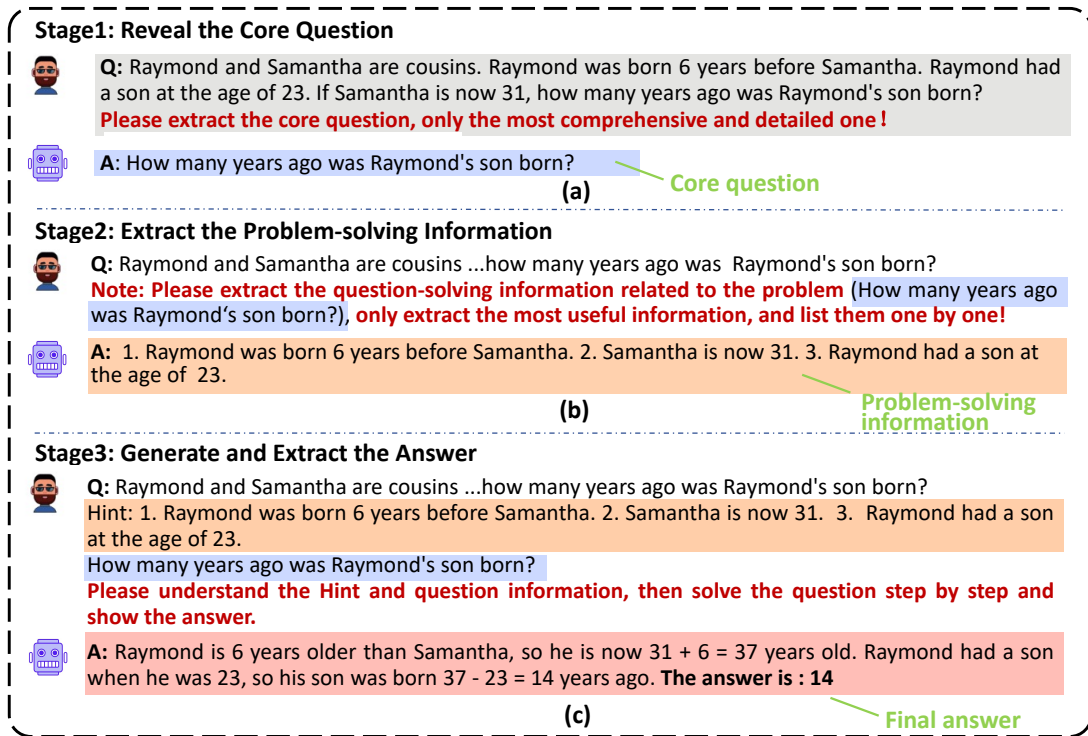


Figure 2: **Illustration of our DUP method**, containing three-stage processes: ❶ revealing the core question from the original input; ❷ extracting the problem-solving information based on the core question; ❸ generating and extracting the final answer via understanding the core question and problem-solving information.

et al., 2023; Zheng et al., 2023) attempt to carefully design more effective prompting strategies to improve the reasoning ability of LLMs. Unfortunately, these prompt methods achieve remarkable performance, but still fail to deal with complex reasoning tasks, e.g., math word problems. As stated by Wei et al. (2022b), the reasoning mistakes of LLMs can be classified into three categories: semantic misunderstanding errors, calculation errors, and step-missing errors. Some prior works (Wang et al., 2023a; Chen et al., 2023a) attempt to reduce these errors, and achieve some performance improvements. However, they mainly focus on the calculation errors and step-missing errors, but neglect the major semantic misunderstanding errors. That is, it is critical but under-explored to study how to address the semantic misunderstanding.

Novelty of our work. Our main contribution lies in discovering that CoT has major errors in semantic understanding, which is the main factor limiting LLMs’ reasoning performance. To address this issue, we are inspired by the human learning process and propose to enforce the LLMs to deeply understand the problems and pay more attention to the core information relevant to solving the problems. Although the method itself might not introduce

too many new technologies, we believe that our findings and approach will promote more related research in the future, which will help the development of this community. Additionally, our method can be easily applied to various LLMs to enhance their reasoning capabilities without introducing any training cost, demonstrating the practical application potential of our approach.

3 DUP Prompting

3.1 Overview

As mentioned in Section 1, semantic misunderstanding is the major error for limiting LLMs’ reasoning performance, which has not been well studied in prior works. To this end, we introduce a new zero-shot CoT prompting approach, called DUP prompting, which aims to improve the LLMs’ reasoning abilities by enforcing the LLMs to fully understand the problem. Figure 2 illustrates the process of our DUP method, which contains three-stage processes. Specifically, in stage 1, DUP reveals the **core question** from a complex and lengthy problem description. In stage 2, DUP further extracts the **problem-solving information** that is crucial for solving the core question from the same description. In stage 3, given the core ques-

tion and problem-solving information, DUP incorporates them into the original question to generate the detailed response, and then extracts the **final answer** from the generated text.

3.2 Stage 1: Reveal the Core Question

Understanding the goal of the question is the first step to solving it, even for humans. Unfortunately, LLMs might be confused by lengthy descriptions of complex reasoning questions, leading to inaccurate understanding and poor performance. In response to this problem, we encourage LLMs to explicitly extract the core question from the original input before reasoning. Specifically, we design a core question extraction prompt “*Please extract core question, only extract the most comprehensive and detailed one!*”, which is appended to the end of question. We then use GPT-3.5-turbo (Ouyang et al., 2022) to extract the core question from the input. As a result, the output of this step will be a shorter and clearer question that will be used to help LLMs focus on the goal of input questions in subsequent steps.

3.3 Stage 2: Extract the Problem-solving Information

In addition to clarifying the goal, it is also important to find the information required to solve the problem. Without fully understanding and utilizing the information provided by the question, reasoning cannot be correctly proceeded. Moreover, it is difficult for LLMs to take full advantage of this information. Therefore, we design a problem-solving information extraction prompt to help solve this problem, *i.e.*, “*Note: Please extract the problem-solving information related to the core question [Core Question info], only extract the most useful information, list them one by one!*”. The slot [Core Question info] contains the core question extracted in Stage 1. The output of this step is a list of information, which is useful in reasoning.

3.4 Stage 3: Generate and Extract the Answer

Given the extracted core question and problem-solving information extracted in previous stages, we incorporate them into the original input by the template “*Hint: [Problem-Solving Info]\n[Core Question]\n Please understand the Hint and question information, then solve the problem step by step and show the answer.*”, where the input slots refer to the corresponding

outputs in previous steps. This prompt is beneficial to improve LLMs’ understanding of the question by explicitly pointing out the goal and necessary information to solve the question. Lastly, following the prior work (Wang et al., 2023a), we enforce the LLMs to extract the final numerical answer from the generated long reasoning text. Compared with rule-based matching methods, using LLMs to extract the final answer is more robust and accurate in practice.

4 Experiments

4.1 Experimental Setup

Tasks and Datasets As stated in Section 1, the core of our DUP is to address the semantic misunderstanding errors in the math word problems. Thus, we mainly evaluate our DUP method on some arithmetic benchmarks. Specifically, we conduct extensive experiments on seven **Arithmetic Reasoning** benchmarks, including 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), SingleEq (Koncel-Kedziorski et al., 2015) and MathQA (Amini et al., 2019). Moreover, to investigate the universality of DUP, we also evaluate it on several reasoning tasks in the other domains, *i.e.*, two **Commonsense Reasoning** benchmarks (CommonsenseQA (Talmor et al., 2019), StrategyQA (Geva et al., 2021)) and two **Symbolic Reasoning** benchmarks (Last Letter (Wei et al., 2022b), Coin Flip (Wei et al., 2022b)). The details of all evaluated datasets are shown in Table 2.

Compared Methods Since our DUP is a zero-shot prompting method, we mainly compare it with other zero-shot methods. For references, two typical few-shot prompting methods are also used as the baselines.

- Zero-shot CoT (Kojima et al., 2022) simply adds a prompt “Let’s think step by step” before each answer.
- Least-to-Most (Zhou et al., 2023) aims to break down a complex problem into a series of simpler sub-problems and then solve them in sequence.
- Plan-and-Solve (Wang et al., 2023a) devises a plan to divide the entire task into smaller sub-tasks, and then carries out the sub-tasks according to the plan. Specifically, we adopt

No.	Template	Reasoning tasks
1	<p>Extract core question: Please extract core question, only the most comprehensive and detailed one!</p> <p>Extract problem-solving information : Note: Please extract the problem-solving information related to the core question [<i>Core Question</i>], only extract the most useful information, list them one by one!</p> <p>Generate the answer: Hint: [<i>Problem-solving Info</i>], \n[<i>Core Question</i>]. \n Please understand the Hint and question information, then solve the question step by step and show the answer.</p>	GSM8K, AddSub, SVAMP, MultiArith, SingleEq, AQuA, MathQA, StrategyQA, Coin Flip, CommonsenseQA
2	<p>Prompt: Please accurately understand the question useful information and solve the question step by step.</p>	Last Letter

Table 1: **Reasoning prompt templates of DUP for all reasoning tasks.** Notably, [*Core Question*] indicates the extracted core question, and [*Problem-solving Info*] indicates the extracted problem-solving information.

Dataset	Domain	# Samples	Answer Type
GSM8K	Math	1,319	Number
MultiArith	Math	600	Number
AddSub	Math	395	Number
SVAMP	Math	1,000	Number
SingleEq	Math	508	Number
AQuA	Math	254	Option
MathQA	Math	2,985	Option
Last Letters	Symbolic	500	String
Coin Flip	Symbolic	500	Yes / No
StrategyQA	Commonsense	2,290	Yes / No
CSQA	Commonsense	1,221	Option

Table 2: **Details of all evaluated datasets.** “Math”, “Symbolic” and “Commonsense” denote the arithmetic, symbolic and commonsense reasoning, respectively. CSQA refers to the CommonsenseQA benchmark.

the more sophisticated Plan-and-Solve (PS+) prompting method in this work.

- R^3 prompting (Tian et al., 2023) instructs LLMs to perform key sentence extraction, variable declaration and answer prediction, *i.e.*, a thought process of reviewing, rephrasing and resolving².
- Manual-CoT (Wei et al., 2022b) is the first CoT method that proposes to use a few CoT demonstrations as exemplars in prompting.
- Auto-CoT (Zhang et al., 2023b) improves the vanilla CoT via sampling questions with diversity and generating reasoning chains to construct demonstrations.

Implementation Details In the main experiments, we use the API-based GPT-3.5-Turbo

² Notably, since the R^3 prompting (Tian et al., 2023) is mainly designed for the math word problems, *i.e.*, involving translating the sentences into equations, we only evaluate it on Arithmetic Reasoning benchmarks.

(0613) (Ouyang et al., 2022) and GPT-4 (0613) (OpenAI, 2023) as the test LLMs. Furthermore, to verify the universality of DUP, we additionally evaluate DUP on more LLMs, covering two proprietary LLMs (*i.e.*, GPT-4o-mini (Hurst et al., 2024) and Gemini-1.5-pro (Team et al., 2024)) and two open-source LLMs (*i.e.*, LLaMA2-Chat-70B (Touvron et al., 2023) and CodeLLaMA-Instruct-34B (Roziere et al., 2023)). To implement our DUP, we show the reasoning prompt templates for all reasoning tasks in Table 1. Notably, since the Last Letter (Wei et al., 2022b) is a simple symbolic reasoning task that does not rely on the problem-solving information, we skip the stage-2 process of DUP and use a simplified prompt. In this work, all GPT models are employed via OpenAI’s API, and we adopt the greedy decoding strategy with the temperature setting of 0 across all experiments. For the few-shot prompting baselines, we keep the recommended number of demonstration examples specified in their original papers.

4.2 Main Results

Arithmetic Reasoning Table 3 presents the main results of Arithmetic Reasoning benchmarks. As seen, compared to the vanilla zero-shot CoT, our DUP method brings consistent and significant performance gains across all reasoning benchmarks. Specifically, in GPT-3.5-turbo settings, DUP improves the accuracy by an average of 4% over Zero-shot CoT. When using GPT-4, our DUP even achieves new state-of-the-art results on **GSM8K (97.1%)** and **SVAMP (94.2%)**, indicating its superiority. Moreover, we also report the results of few-shot counterparts. Due to the high cost of GPT-4 API, we use the more affordable GPT-3.5-turbo as the responder for few-shot methods. Generally, the performance of zero-shot methods tends to be

Model	Method	Arithmetic Reasoning							Score	
		SVAMP	GSM8K	AddSub	MultiArith	AQuA	SingleEq	MathQA	Avg.	Δ
<i>Performance of Zero-shot Methods</i>										
GPT-3.5-Turbo	Zero-shot CoT	79.3	78.9	85.8	95.3	53.0	93.5	63.7	<u>78.5</u>	-
	Least-to-Most	80.9	77.5	91.3	95.5	57.4	93.5	66.0	<u>80.3</u>	+1.8
	Zero-shot PS+	80.7	79.3	86.5	92.0	55.9	93.0	66.8	<u>79.2</u>	+0.7
	R ³ prompting	81.6	79.4	92.0	95.5	59.4	94.6	64.4	<u>81.0</u>	+2.5
	DUP (Ours)	82.5	82.3	92.1	97.8	60.2	94.9	68.1	<u>82.6</u>	+4.1
GPT-4	Zero-shot CoT	90.4	94.6	92.4	97.8	72.8	95.0	82.1	<u>89.3</u>	-
	Least-to-Most	90.3	92.1	92.1	97.1	71.6	95.0	82.4	<u>88.7</u>	-0.6
	Zero-shot PS+	92.6	94.3	93.1	98.1	75.5	95.3	83.4	<u>90.3</u>	+1.0
	R ³ prompting	93.4	92.1	94.5	98.1	75.1	94.8	81.7	<u>90.0</u>	+0.7
	DUP (Ours)	94.2	97.1	95.1	98.1	77.1	96.0	84.1	<u>91.7</u>	+2.4
<i>Performance of Few-shot Methods</i>										
GPT-3.5-Turbo	Manual-CoT	78.5	81.6	90.6	95.6	55.9	94.2	64.2	<u>80.1</u>	+1.6
	Auto-CoT	82.9	80.2	89.9	99.0	54.3	94.6	64.8	<u>80.8</u>	+2.3

Table 3: **Results on Arithmetic Reasoning benchmarks.** The best results in the zero-shot setting are in bold. “ Δ ” denotes the average performance **improvement** or **decline** of various methods compared to Zero-shot CoT.

Method	CSQA	StrategyQA	Avg.	Δ
Zero-shot CoT	72.3	66.1	<u>69.2</u>	-
Least-to-Most	71.9	61.5	<u>66.7</u>	-2.5
Zero-shot PS+	68.8	62.8	<u>65.8</u>	-3.4
DUP (Ours)	74.5	68.5	<u>71.5</u>	+2.3
Few-shot Manual-CoT	76.5	64.8	<u>70.8</u>	+1.6
Few-shot Auto-CoT	74.2	62.5	<u>68.3</u>	-0.9

Table 4: **Results of GPT-3.5-Turbo on Commonsense Reasoning benchmarks.**

Method	Last Letter	Coin Flip	Avg.	Δ
Zero-shot CoT	60.8	94.4	<u>77.6</u>	-
Least-to-Most	83.2	82.8	<u>83.0</u>	+2.4
Zero-shot PS+	60.6	95.4	<u>78.0</u>	+0.4
DUP (Ours)	81.2	97.6	<u>89.4</u>	+11.8
Few-shot Manual-CoT	74.4	98.2	<u>86.3</u>	+8.7
Few-shot Auto-CoT	81.2	98.6	<u>89.9</u>	+12.3

Table 5: **Results of GPT-3.5-Turbo on Symbolic Reasoning benchmarks.**

lower than that of few-shot methods. However, with the help of DUP, GPT-3.5 can even achieve remarkable zero-shot performance that is higher than few-shot methods. These results prove the effectiveness of DUP.

Commonsense and Symbolic Reasoning Table 4 shows the performance on Commonsense Reasoning datasets. Considering the experimental cost, we only used GPT-3.5-turbo as the test LLM. Compared to zero-shot methods, DUP consistently outperforms all counterparts. In comparison with few-shot methods, DUP also achieves comparable or even better performance. Table 5 lists the results on Symbolic Reasoning datasets. On Last Letters,

Model	Method	GSM8K	AddSub	Avg.
GPT-4o-mini	Zero-shot CoT	92.4	94.4	<u>93.4</u>
	DUP (Ours)	93.8	95.7	<u>94.8</u>
Gemini-1.5-pro	Zero-shot CoT	94.3	95.6	<u>95.0</u>
	DUP (Ours)	95.6	98.0	<u>96.8</u>
LLaMA2-70B*	Zero-shot CoT	53.9	75.6	<u>64.7</u>
	DUP (Ours)	56.4	87.8	<u>72.1</u>
CodeLLaMA-34B*	Zero-shot CoT	39.1	81.2	<u>60.1</u>
	DUP (Ours)	43.5	86.0	<u>64.7</u>

Table 6: **Results of more LLMs using different prompting methods on GSM8K and Addsub.** “*” refers to the chat version of open-source LLMs.

zero-shot DUP (81.2%) is marginally worse than Zero-shot Least-to-Most (83.2%), on par with few-shot Auto-CoT (81.2%), but significantly exceeds other zero-shot methods and few-shot Manual-CoT (74.4%). On Coin Flip, zero-shot DUP (97.6%) is slightly worse than few-shot Manual-CoT (98.2%) and few-shot Auto-CoT (98.6%), but significantly outperforms other zero-shot baseline methods. In general, we can basically conclude that DUP outperforms other zero-shot methods, and has great potential to beat the few-shot methods.

Results of More Proprietary and Open-source LLMs To verify the universality of DUP, we further evaluate our method on 4 cutting-edge LLMs, covering two proprietary LLMs (*i.e.*, GPT-4o-mini (Hurst et al., 2024) and Gemini-1.5-pro (Team et al., 2024)) and two open-source LLMs (*i.e.*, LLaMA2-Chat-70B (Touvron et al., 2023) and CodeLLaMA-Instruct-34B (Roziere et al., 2023)). As seen in Table 6, in both cases of proprietary and open-source LLMs, our DUP can still outper-

Stage 1	Stage 2	Stage 3	GSM8K	AQuA	Avg.
✗	✗	✗	76.5	51.2	<u>63.8</u>
✓	✗	✗	78.9	53.1	<u>66.0</u>
✗	✓	✗	80.6	55.1	<u>67.8</u>
✗	✗	✓	80.3	54.7	<u>67.5</u>
✓	✓	✗	79.9	57.0	<u>68.4</u>
✓	✗	✓	80.8	56.2	<u>68.5</u>
✗	✓	✓	81.7	58.2	<u>69.9</u>
✓	✓	✓	82.3	60.2	<u>71.2</u>

Table 7: **Ablation study for different variations of DUP.** We report the results of GPT-3.5-turbo on GSM8K and AQuA. Notably, Stage 1 involves extracting core questions, Stage 2 focuses on extracting problem-solving information, and Stage 3 entails solving the problem step by step.

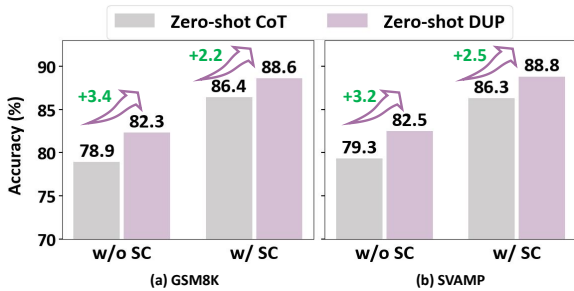


Figure 3: **Results of DUP Prompting with and without self-consistency(SC) using GPT-3.5-turbo LLM on GSM8K and SVAMP.**

form the baseline zero-shot CoT by a large margin on GSM8K and AddSub benchmarks. This also proves the DUP can work well in the other model scenarios.

4.3 Ablation Study

In this part, we conduct a series of ablation experiments to investigate the impact of each stage in our DUP, where Stage 1 of DUP involves extracting core questions, Stage 2 focuses on extracting problem-solving information, and Stage 3 entails solving the problem step by step. In Table 7, we report the results of various combinations of the three stages in our DUP. Specifically, we conduct the ablation experiments on GPT-3.5-turbo and present the results of GSM8K and AQuA benchmarks. As seen, removing each stage results in performance degradation, and the combination of all stages achieves the best performance on both benchmarks. These results demonstrate the importance of each stage in our DUP.

Template
Perform the following processes:
1. Extract core question from following question delimited by triple backticks with 1 sentence, only the most comprehensive and detailed one!
2. Extract the the most useful information related to the core question from the same question delimited by triple backticks and list them one by one!
3. Answer the question delimited by triple backticks with fully and comprehensively considering the core question and useful information.
4. Output a json object that contains the following keys: "final_answer: <intfloat>"
Use the following format:
Text: <question to answer>
Core Question: <extracted core question>
Hints: <extracted useful information, listed one by one>
Reason Steps: <the reasoning steps to get the final answer>
Output Json: <json with the final answer>
Text: [<i>Input Question</i>]

Table 8: **Details of our simplified DUP-s Prompts.** The slot [*Input Question*] denotes the original problem.

4.4 Discussion

Compatibility with Self-consistency We employ an innovative decoding strategy with self-consistency (SC) (Wang et al., 2023b) as a substitute for the conventional greedy decoding approach, which initially samples N reasoning paths rather than only opting for the greedy approach. Subsequently, choosing the most consistent answer as the answer. Existing works (Wang et al., 2023a; Xu et al., 2023) indicate that adopted SC notably enhances the performance of chain-of-thought prompting. Here, to verify whether using SC can further enhance the performance of DUP, we conduct experiments on GSM8K and SVAMP using GPT-3.5-Turbo, setting the temperature to 0.7 and N to 10. The results are illustrated in Figure 3, where the SC strategy brings remarkable performance improvements. Notably, DUP with SC (88.6% and 88.8%) consistently outperforms Zero-shot CoT with SC (86.4% and 86.3%), continuing to prove its superiority.

More Accurate Core Questions and Problem-solving Information Lead to Better Performance

As stated in Section 1, the core of our DUP is to guide LLMs to deeply understand the problems, *i.e.*, extracting the core question and key problem-solving information. To verify it, we conduct contrastive experiments on AQuA, GSM8K, and SVAMP datasets. Specifically, using the GPT-3.5-Turbo as the final responder, we leverage different LLMs (*i.e.*, LLaMA2-Chat-70B, GPT-3.5, GPT-4) to extract the core question in Stage 1 and the key problem-solving information in Stage 2, re-

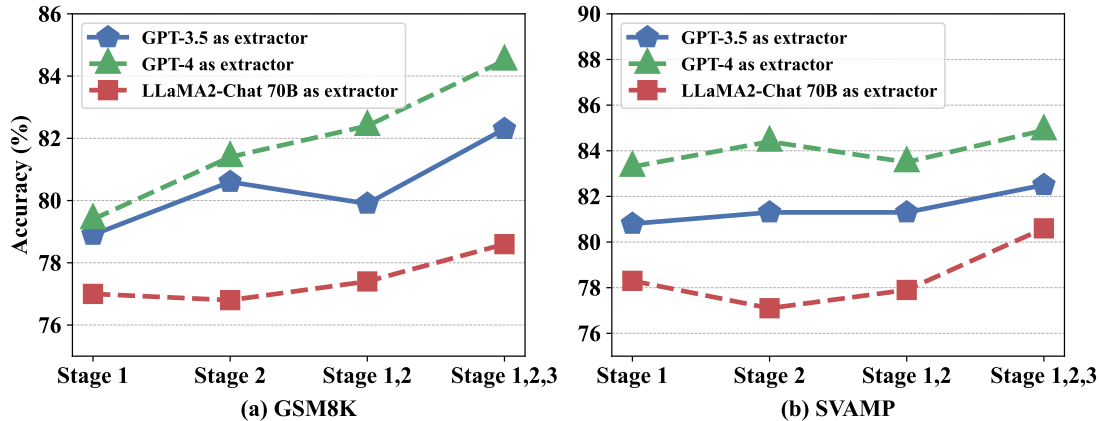


Figure 4: **Analysis of different information extractors used in our DUP.** We use the GPT-4, GPT-3.5-turbo, and Llama-2-Chat 70b to extract core question (Stage1) and problem-solving information (Stage2) extractor, and leverage the extracted contents to guide the responses of GPT-3.5-turbo (Stage3). We see that more accurate core questions and problem-solving information lead to better performance.

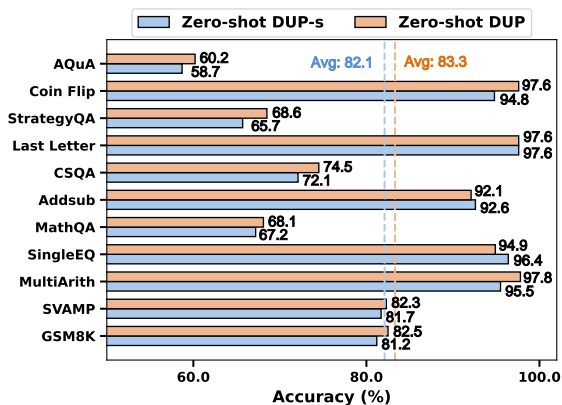


Figure 5: **Performance of DUP and DUP-s across various reasoning tasks on GPT-3.5-Turbo**, where DUP-s merges the three-stage prompts into one prompt. Orange and Blue dashlines represent the average accuracy of DUP and DUP-s, respectively. We see that our simplified DUP-s also achieves remarkable performance with less inference budget.

spectively. The contrastive results are illustrated in Figure 4. As seen, when using the GPT-4 as the extractor, GPT-3.5 responder can achieve better performance than that using GPT-3.5 as the extractor. Conversely, using the LLaMA2-Chat-70B as the extractor leads to worse results. These results demonstrate that better core questions and key problem-solving information can result in better reasoning performance, confirming our statement.

Reduce inference cost without much performance degradation Some readers may be concerned that the three-stage processes in DUP will cause too much inference cost. Hence, we further propose the simplified DUP method, namely

Template
Question: <i>[Input Question]</i> .
Wrong Response: <i>[Wrong Answer]</i> .
Correct Response: <i>[Correct Answer]</i> .
Please judge which type of error it belongs to based on the above information:
1. Semantic Misunderstanding: semantic misunderstanding or lack of commonsense concepts.
2. Calculation error: errors occurred while performing a basic operation.
3. Step-missing errors: missing step and hallucination.
Finally, please explain why this error falls into the category.

Table 9: **Prompts for error analysis.** The slot *[Input Question]* denotes the original problem, where *[Wrong Question]* and *[Correct Question]* denote the incorrect text generated by the LLMs and the original label.

DUP-s, which merges the three-stage prompts into one prompt. The details of our DUP-s method can be found in Table 8. We conduct contrastive experiments on all 11 reasoning benchmarks, and illustrate the results in Figure 5. As seen, DUP-s achieves comparable performance to DUP, and even achieves better performance on two tasks of Addsub and SingleEQ. Therefore, in the case of a limited inference budget, using our simplified DUP-s method is also a good choice.

4.5 Error Analysis

Here, to verify whether DUP indeed reduces the semantic misunderstanding, we randomly select 300 samples for each reasoning dataset, and perform error analysis for the questions with incorrect answers by prompting the GPT-3.5 LLM. The detailed prompt used to categorize the failure examples is shown in Table 9 and the quantitative results are illustrated in Figure 6. As seen, compared with

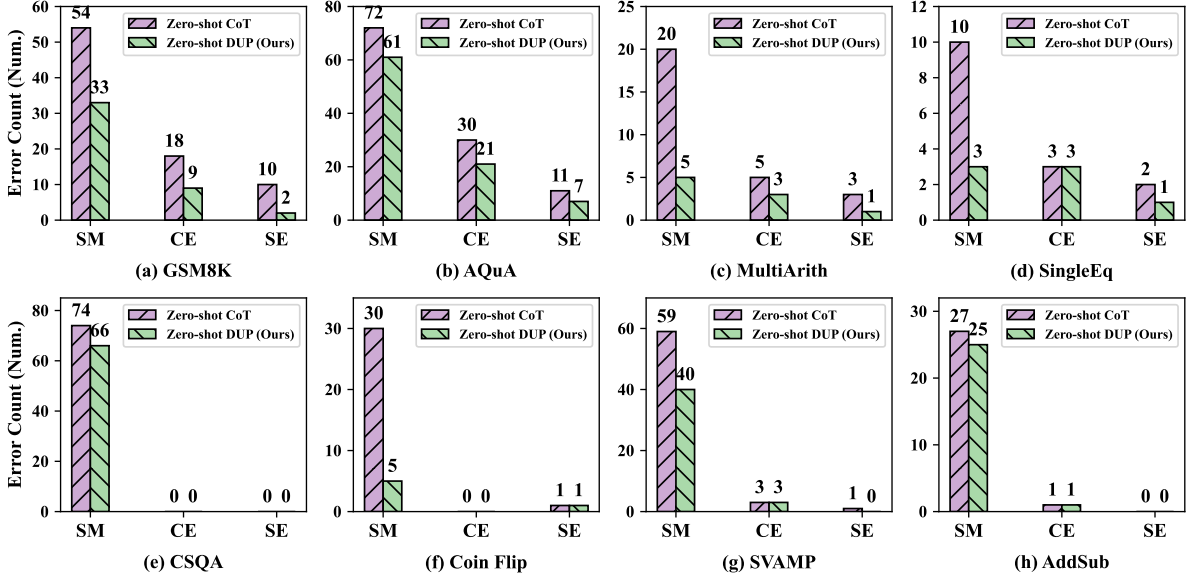


Figure 6: **Quantitative error analyses of different prompting methods.** Notably, “SM”, “CE” and “SE” denote the “Semantic Misunderstanding”, “Calculation Error” and “Step-missing Error”. We randomly select 300 examples for each reasoning dataset (except AQuA which only contains 254 examples), and use GPT-3.5-Turbo LLM to generate responses and count failed answers. We can see that our method reduces the frequency of various error types compared with Zero-shot CoT.

the baseline zero-shot CoT, our DUP reduces semantic misunderstanding effectively. Additionally, we can also find that DUP reduces the calculation and step-missing error as well. One possible reason is that learning more problem-solving information can lead to more accurate reasoning steps. To have a close look, we present a case study on SVAMP, as shown in Figure 7. It can be seen that the zero-shot CoT fails to generate the correct answer, but with the help of DUP, LLMs can better understand the problems and generate an accurate answer.

Moreover, some readers may concern whether the above LLM-based error analyses are reliable and trustworthy. Regarding this concern, we would like to state that, although LLMs may not have the ability to directly solve the difficult reasoning problems, LLMs can be instructed to reflect on their own CoT, which allows them to identify errors and explain the cause of these errors (Madaan et al., 2024). Such a self-reflect ability of LLMs has been explored by many prior studies (Yao et al., 2023; Pan et al., 2024). Specifically, we show some cases of incorrect model predictions on GSM8k and the corresponding error analysis results in Table 11. It can be found that LLM-based analysis method can indeed identify the cause of errors and make the correct classification of errors.

Furthermore, to verify the credibility of this LLM-based analysis method, we manually classify

the error types for several benchmarks. The contrastive results are listed in Table 10. As seen, there is a significant correlation between the manual analysis results and the LLM-based results, validating the reliability of our analyses.

5 Conclusion

In this work, we reveal that deeply understanding the whole problem is crucial for tackling complex reasoning tasks. Consequently, we introduce the DUP method to improve the LLMs’ reasoning abilities by encouraging them to deeply understand the problem. A series of experiments on arithmetic, commonsense, and symbolic reasoning tasks prove that DUP brings consistent and significant performance gains across all benchmarks and LLMs. Additionally, DUP outperforms the other zero-shot counterparts by a large margin, and achieves new SOTA results in two popular benchmarks, *i.e.*, GSM8K and SVAMP. More in-depth discussions and systematic analyses further reveal when and where our DUP works well. Moreover, considering that fully understanding the whole problem may also be beneficial to non-reasoning tasks, we will attempt to expand our method to more fields in future work.

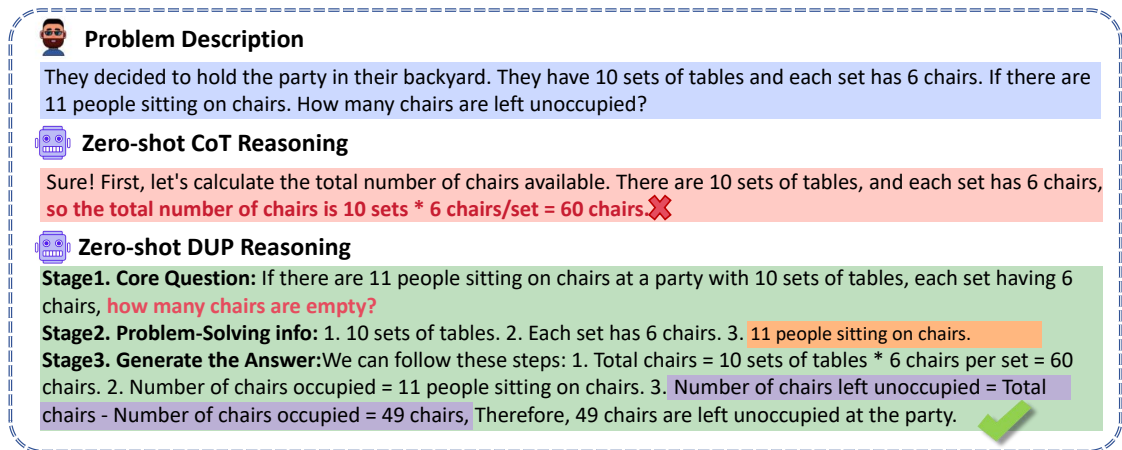


Figure 7: Case study on SVAMP. Zero-shot CoT fails to generate the correct answer, but our DUP can extract more related problem-solving information and make the correct prediction via deeply understanding the problems.

Error type	GSM8K		MultiArith		Coin Filp	
	LLM-based	Manual	LLM-based	Manual	LLM-based	Manual
Semantic Misunderstanding	54 / 33	64 / 39	20 / 5	24 / 7	30 / 5	31 / 5
Calculation Error	18 / 9	6 / 2	5 / 3	0 / 0	0 / 0	0 / 0
Step-missing Error	10 / 2	12 / 3	3 / 1	4 / 2	1 / 0	0 / 0

Table 10: Comparison between LLM-based and manual error analysis methods. Notably, we report the analysis results on GPT-3.5’s responses (as mentioned in Figure 6). For each result, we list the number of errors prompting with zero-shot CoT and our DUP, respectively.

References

- 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 *NAACL*.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *NeurIPS*.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2023a. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023b. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *TMLR*.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2022. PaLM: Scaling language modeling with pathways. *arXiv preprint*.
- 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. *arXiv preprint*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. *arXiv preprint*.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *TACL*.

- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*.
- Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *ACL*.
- Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*.
- William J Hoyer, George W Rebok, and Susan Marx Sved. 1979. Effects of varying irrelevant information on adult age differences in problem solving. *Journal of gerontology*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint*.
- Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.
- Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *ACL*.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL*.
- Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. 2023. Tinygsm: achieving > 80% on gsm8k with small language models. *arXiv preprint*.
- Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhunoye, Yiming Yang, and 1 others. 2024. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*.
- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, and 1 others. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop*.
- OpenAI. 2023. Gpt-4 technical report.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.
- Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics*.
- Maria Chiara Pasolunghi, Cesare Cornoldi, and Stephanie De Liberto. 1999. Working memory and intrusions of irrelevant information in a group of specific poor problem solvers. *Memory & Cognition*.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *NAACL*.
- Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, and 1 others. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint*.
- Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *EMNLP*.
- Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint*.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, and 1 others. Multitask prompted training enables zero-shot task generalization. In *ICLR*.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adria Garriga-Alonso, and 1 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL*.

- Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.
- Qingyuan Tian, Hanlun Zhu, Lei Wang, Yang Li, and Yunshi Lan. 2023. R3 prompting: Review, rephrase and resolve for chain-of-thought reasoning in large language models under noisy context. In *Findings of EMNLP*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint*.
- Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *ACL*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *ICLR*.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022a. Emergent abilities of large language models. *TMLR*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *NeurIPS 2022*.
- Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, and Jian-guang Lou. 2023. Re-reading improves reasoning in language models. *arXiv preprint*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Meta-math: Bootstrap your own mathematical questions for large language models. *arXiv preprint*.
- Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint*.
- Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023a. Cumulative reasoning with large language models. *arXiv preprint*.
- Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023b. Automatic chain of thought prompting in large language models. In *ICLR*.
- Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.
- Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2023. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert. *arXiv preprint*.
- Qihuang Zhong, Liang Ding, Yibing Zhan, Yu Qiao, Yonggang Wen, Li Shen, Juhua Liu, Baosheng Yu, Bo Du, Yixin Chen, and 1 others. 2022. Toward efficient language model pretraining and downstream adaptation via self-evolution: A case study on superglue. *arXiv preprint*.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *ICLR*.

❶ *Semantic misunderstanding error*

Question: Last night Rick killed ten wolves and 15 cougars while hunting. Today Rick killed three times as many wolves as cougars and three fewer cougars than the previous night. How many animals did Rick kill?

Answer: 73

Model Output: Let's break down the information provided in the hint and the question to solve for the total number of animals Rick killed. 1. Last night, Rick killed 10 wolves and 15 cougars. 2. Today, Rick killed three times as many wolves as cougars. 3. Today, Rick killed three fewer cougars than the previous night. First, let's calculate the number of animals Rick killed today: - Since Rick killed three times as many wolves as cougars today, the number of wolves he killed is 3 times the number of cougars he killed last night, which is $15 * 3 = 45$ wolves. - Today, Rick killed three fewer cougars than the previous night, so the number of cougars he killed today is $15 - 3 = 12$ cougars. Now, we can calculate the total number of animals Rick killed: - Last night, Rick killed 10 wolves and 15 cougars, which is a total of $10 + 15 = 25$ animals. - Today, Rick killed 45 wolves and 12 cougars, which is a total of $45 + 12 = 57$ animals. Therefore, the total number of animals Rick killed in both nights is $25 + 57 = 82$ animals.

Error Analysis: The main problem in incorrect response lies in the misunderstanding of the problem conditions. The incorrect response is that Rick killed three times the number of cougars killed yesterday, which is $15 * 3 = 45$ wolves. This is incorrect. According to the problem, Rick killed three times as many wolves today as he killed cougars today, not yesterday's. Due to Rick killing 3 fewer cougars today than yesterday, which is $12 (15 - 3 = 12)$, he killed 36 wolves today. Therefore, the total number of animals Rick killed in both nights should be $10 + 15 + 12 + 36 = 73$.

❷ *Calculation error*

Question: Mary has 6 jars of sprinkles in her pantry. Each jar of sprinkles can decorate 8 cupcakes. Mary wants to bake enough cupcakes to use up all of her sprinkles. If each pan holds 12 cupcakes, how many pans worth of cupcakes should she bake?

Answer: 4

Model Output: To find out how many pans worth of cupcakes Mary should bake in order to use up all 6 jars of sprinkles, we can use the following steps: 1. Calculate the total number of cupcakes that can be decorated with 6 jars of sprinkles: **Total cupcakes = 6 jars * 8 cupcakes per jar = 288 cupcakes.** 2. Determine the number of pans needed to bake 288 cupcakes: Number of pans = Total cupcakes / cupcakes per pan = $288 \text{ cupcakes} / 12 \text{ cupcakes per pan} = 24$ pans. Therefore, Mary should bake 24 pans.

Error Analysis: This calculation error mistakenly calculated " $6 * 8$ " as 288 when calculating the total number of cupcakes, which is a clear multiplication error. The correct calculation should be $6 * 8 = 48$ cupcakes. Then, to determine how many pans are needed, the total number of pans should be divided by the number of cupcakes each pan can hold, which is $48 / 12 = 4$ pans. Therefore, the correct answer is that Mary should bake 4 pans.

❸ *Step-missing error*

Question: A team of 4 painters worked on a mansion for 3/8ths of a day every day for 3 weeks. How many hours of work did each painter put in?

Answer: 189

Model Output: To calculate the total hours of work each painter put in over the course of 3 weeks, we can follow these steps: 1. Calculate the total number of days worked: **3 weeks = 3 * 7 days/week = 21 days.** 2. Calculate the total hours worked by the team each day: $3/8$ ths of a day = $3/8 * 24 \text{ hours} = 9$ hours. 3. Calculate the total hours worked by the team over 3 weeks: **Total hours = 9 * 3 = 27 hours.** So, each painter put in approximately 27 hours of work over the course of 3 weeks.

Error Analysis: The error is due to the missing of critical calculation steps during the calculation process. During the problem-solving process, the total number of working days was calculated as " $3 * 7 = 21$ ", but this step was ignored when calculating the final hours. The correct calculation process should be $9 * 21 = 189$ hours. Therefore, the correct answer should be 189 hours.

Table 11: **Cases of error analysis.** Notably, we present some incorrect predictions of GPT-3.5-Turbo equipped with our DUP on the GSM8K. The key information is highlighted in orange and red.