Achieving >97% on GSM8K: Deeply Understanding the Problems Makes LLMs Better Solvers for Math Word Problems

Anonymous ACL submission

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 LLMs' performance. 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 10 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 an accuracy of 97.1% under zero-shot setting.

1 Introduction

Despite the impressive performance of Large Language Models (LLMs) in diverse NLP tasks (Brown et al., 2020; Touvron et al., 2023; OpenAI, 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. (2022) 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.

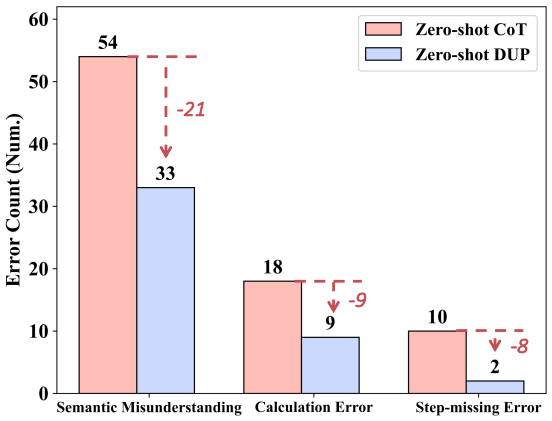

Figure 1: **Error analysis of GSM8K problems with incorrect answers returned by zero-shot CoT and our DUP using GPT-3.5 LLM.** We randomly sample 300 GSM8K problems, and follow (Wei et al., 2022) and (Wang et al., 2023a) to assign the "Semantic Misunderstanding", "Calculation Error" and "Step-missing Error" to each incorrect answer. *We see that our DUP method effectively reduces the errors among all types.*

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), Plan-and-Solve (PS) prompting (Wang et al., 2023a), and Complex CoT (Fu et al., 2023). Although achieving remarkable progress, they still fall short in dealing with complex reasoning tasks, *e.g.*, math word problems. As stated by Wei et al. (2022), there are three main error types in the context of CoT-based reasoning: ***semantic misunderstanding errors***, ***calculation errors***, and ***step-missing errors***. In our preliminary experiments (as shown in Figure 1), we found that CoT has major errors in semantic understanding, 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 LLMs' reasoning abilities 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 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 10 reasoning datasets across math, commonsense, and symbolic reasoning benchmarks. 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%**).

Contributions. 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.

2 Related Works

2.1 Reasoning with Large Language Models

In recent years, we have witnessed numerous large language models (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 various natural language understanding and generation tasks. However, LLMs usually struggle to provide stable and accurate answers when dealing with 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., 2022). Recent works (Yuan et al., 2023; Luo et al., 2023; Yu 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., 2022; Kojima et al., 2022) attempt to use cheaper prompting methods to strengthen the LLMs' reasoning abilities without additional training. Wei et al. (2022) 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) attempt to carefully design more effective prompting strategies to

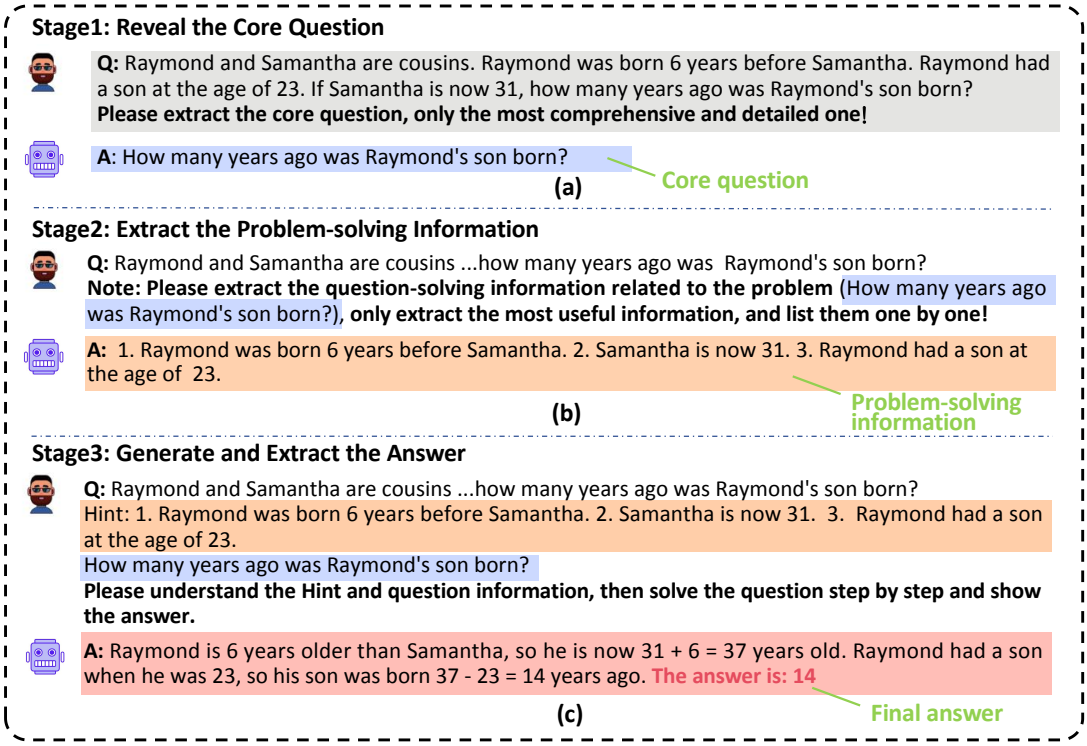

Figure 2: **Illustration of our DUP prompting strategy**, which contains three-stage processes: ❶ revealing the core question from the original input; ❷ extracting the problem-solving information based on the core question; ❸ generating and extracting the final answer via understanding the core question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

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., 2022), 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. In this paper, 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 such a simple prompting method might not introduce too many new technologies, we are one of the rare works to reveal the underlying causes of semantic misunderstanding errors and provide a new view for addressing these errors, which can promote more related research in this field.

3 DUP Prompting

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 question 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.

3.1 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.

3.2 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.

3.3 Stage 3: Generate and Extract the Answer

Given the 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. More details of extracting answer can be found in Appendix A.1.

Dataset	Domain	# Samples	Answer Format
GSM8K	Math	1319	Number
MultiArith	Math	600	Number
AddSub	Math	395	Number
SVAMP	Math	1000	Number
SingleEq	Math	508	Number
AQuA	Math	254	Option
Last Letters	Symbolic	500	String
Coin Flip	Symbolic	500	Yes / No
StrategyQA	Commonsense	2290	Yes / No
CSQA	Commonsense	1221	Option

Table 1: **Details of all evaluated datasets.** "Math", "Symbolic" and "Commonsense" denote the arithmetic, symbolic and commonsense reasoning, respectively. CSQA refers to the CommonensenseQA benchmark.

4 Experiments

4.1 Setup

Tasks and Datasets. We conduct extensive experiments on 6 **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) and SingleEq (Koncel-Kedziorski et al., 2015). Moreover, to investigate the universality of our DUP, we also evaluate it on several reasoning tasks in the other domains, *i.e.*, 2 **Commonsense Reasoning** benchmarks (CommonsenseQA (Talmor et al., 2019), StrategyQA (Geva et al., 2021)) and 2 **Symbolic Reasoning** benchmarks (Last Letter (Wei et al., 2022), Coin Flip (Wei et al., 2022)). The details of all evaluated datasets are shown in Table 1.

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)[1] devises

[1]We adopt the more sophisticated Plan-and-Solve (PS+) prompting with more detailed instructions in this work.

Model	Method	Arithmetic Reasoning						Score	
		SVAMP	GSM8K	AddSub	MultiArith	AQuA	SingleEq	*Avg.*	Δ
Performance of Zero-shot Methods									
GPT-3.5-Turbo	Zero-shot CoT	79.3	78.9	85.8	95.3	53.0	93.5	80.9	-
	Least-to-Most	80.9	77.5	91.3	95.5	57.4	93.5	82.6	+1.7
	Zero-shot PS+	80.7	79.3	86.5	92.0	55.9	93.0	81.2	+0.3
	DUP (Ours)	**82.5**	**82.3**	**92.1**	**97.8**	**60.2**	**94.9**	**84.9**	**+4.0**
GPT-4	Zero-shot CoT	90.4	94.6	92.4	97.8	72.8	95.0	90.6	-
	Least-to-Most	90.3	92.1	92.1	97.1	71.6	95.0	89.7	-0.9
	Zero-shot PS+	92.6	94.3	93.1	**98.1**	75.5	95.3	91.4	+0.8
	DUP (Ours)	**94.2**	**97.1**	**95.1**	**98.1**	**77.1**	**96.0**	**92.9**	**+2.3**
Performance of Few-shot Methods									
GPT-3.5-Turbo	Manual-CoT	78.5	81.6	90.6	95.6	55.9	94.2	82.6	+1.7
	Auto-CoT	82.9	80.2	89.9	99.0	54.3	94.6	83.4	+2.5

Table 2: **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	69.2	-
Least-to-Most	71.9	61.5	66.7	-2.5
Zero-shot PS+	68.8	62.8	65.8	-3.4
DUP (Ours)	**74.5**	**68.5**	**71.5**	**+2.3**
Few-shot Manual-CoT	76.5	64.8	70.8	+1.6
Few-shot Auto-CoT	74.2	62.5	68.3	-0.9

Table 3: **Results of Commonsense Reasoning benchmarks**. Here, GPT-3.5-turbo is used as the reasoner.

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

Table 4: **Results of Symbolic Reasoning benchmarks**. We also use the GPT-3.5-turbo as the reasoner.

a plan to divide the entire task into smaller sub-tasks, and then carries out the sub-tasks according to the plan.

- Manual-CoT (Wei et al., 2022) 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. We use the public GPT-3.5-Turbo (0613) (Ouyang et al., 2022) and GPT-4 (0613) (OpenAI, 2023) as the test LLMs. In this work, all 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 2 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%)**.

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

Commonsense and Symbolic Reasoning. Table 3 shows the performance on Commonsense Reasoning datasets. Considering the experimental cost, we only used GPT-3.5-turbo as the backbone LLM. Compared to zero-shot methods, our DUP

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

Table 5: **Ablation study for different variations of DUP prompting** using GPT-3.5-turbo LLMs on GSM8K and AQuA datasets. 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.

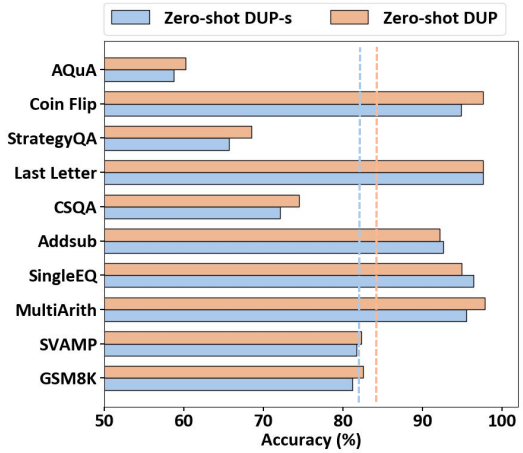

Figure 3: **Performance of DUP and DUP-s across various reasoning tasks on GPT-3.5-Turbo**, where DUP-s merges the three-stage prompts into one prompt. Orange and Blue dashlines represent the average accuracy of DUP and DUP-s, respectively. We see that our simplified DUP-s method also achieves remarkable performance with less inference budget.

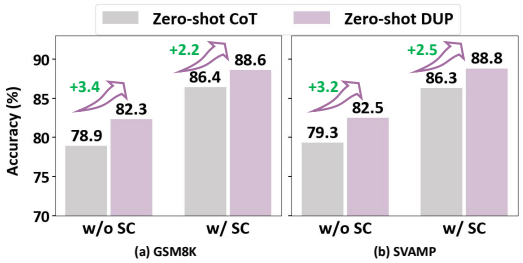

Figure 4: **Results of DUP Prompting with and without self-consistency(SC)** using GPT-3.5-turbo LLM on GSM8K and SVAMP.

method consistently outperforms all other counterparts. In comparison with few-shot methods, our DUP also achieves comparable or even better performance.

Table 4 lists the results on Symbolic Reasoning datasets. On Last Letters, 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 approaches 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 our DUP outperforms other zero-shot counterparts, and has great potential to beat the few-shot methods.

4.3 Ablation Study

In this part, we conduct a series of ablation experiments to investigate 1) the impact of each stage in our DUP, and 2) how to reduce the inference costs and maintain the performance.

Impact of different stages in our DUP. In Table 5, we report the results of various combinations of the three stages in our DUP. As seen, removing each stage results in performance degradation, and the combination of all stages achieves the best performance on GSM8K and AQuA benchmarks. These results demonstrate the importance of each stage in our DUP.

Reduce inference cost without much performance degradation. Some readers may be concerned that the three-stage processes in our DUP will cause too much inference cost. Hence, we further propose the simplified DUP method, namely DUP-s, which merges the three-stage prompts into one prompt. We conduct contrastive experiments on all 10 reasoning benchmarks, and illustrate the results in Figure 3. It can be found that on most tasks, 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.

4.4 Discussion and Analysis

Compatibility with Self-consistency. We employ an innovative decoding strategy with self-consistency (SC) (Wang et al., 2023b) as a sub-

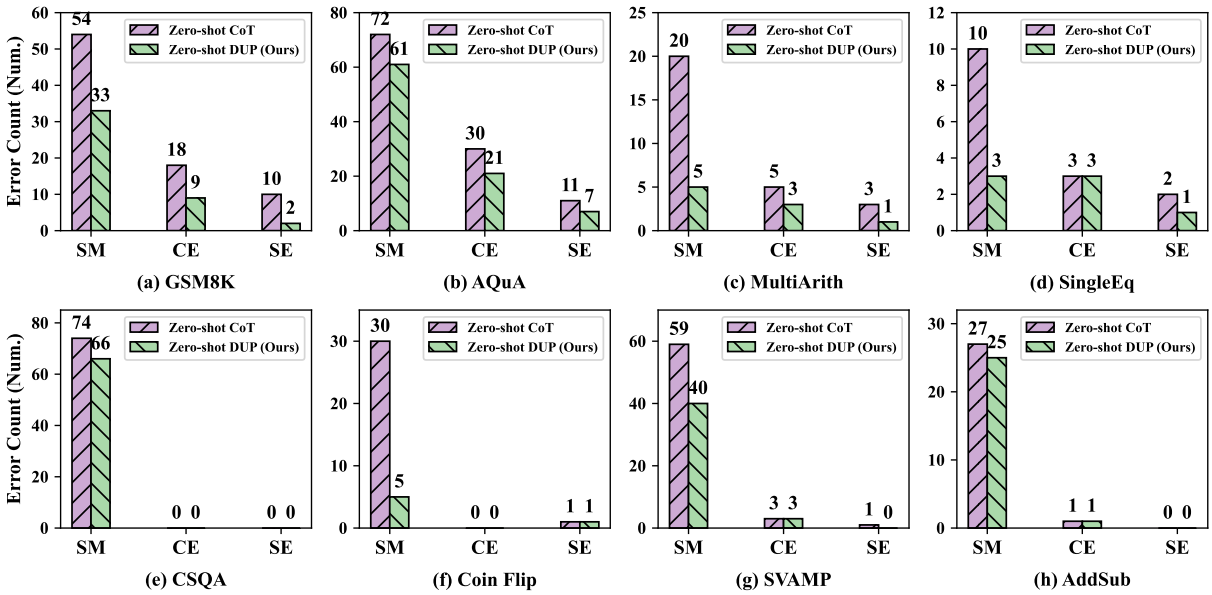

Figure 5: **Quantitative error analyses of different prompting methods.** Notably, "SM", "CE" and "SE" denote the "Semantic Misunderstanding", "Calculation Error" and "Step-missing Error". We randomly select 300 examples for each reasoning dataset (except AQuA which only contains 254 examples), and use 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.

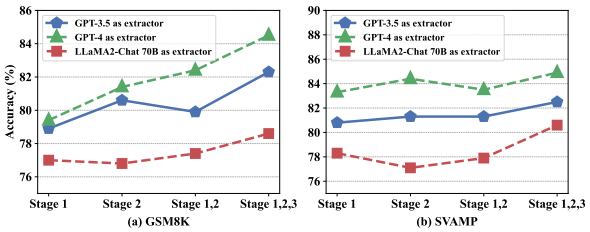

Figure 6: **Analysis of different information extractors used in our DUP**. We use the GPT-4, GPT-3.5-turbo, and Llama-2-Chat 70b to extract core question (Stage1) and problem-solving information (Stage2) extractor, and leverage the extracted contents to guide the responses of GPT-3.5-turbo (Stage3). We see that more accurate core questions and problem-solving information lead to better performance.

Method	GSM8K	AddSub	Avg.	Δ
LLaMA-2-Chat-13b				
Zero-shot CoT	35.1	70.6	52.8	-
DUP (Ours)	**35.9**	**79.7**	**57.8**	+5.0
LLaMA-2-Chat-70b				
Zero-shot CoT	53.9	75.6	64.7	-
DUP (Ours)	**56.4**	**87.8**	**72.1**	+7.4
CodeLLaMA-Instruct-13b				
Zero-shot CoT	24.2	73.1	48.6	-
DUP (Ours)	**28.1**	**74.6**	**51.3**	+2.7
CodeLLaMA-Instruct-34b				
Zero-shot CoT	39.1	81.2	60.1	-
DUP (Ours)	**43.5**	**86.0**	**64.7**	+4.1

Table 6: **Results of various Open-source LLMs on GSM8K and Addsub.** We see that our DUP method still achieves much better performance than the baseline Zero-shot CoT among all open-source LLMs.

stitute 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 4, 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 the superiority of our DUP method.

Whether DUP also works well on Open-source LLMs. In the above experiments, we mainly evaluate our DUP in the close-source GPT LLMs. To verify whether our DUP also works well on other open-source LLMs, we evaluate our method on 4 widely-used LLMs, *i.e.*, LLaMA-2-Chat 13b and

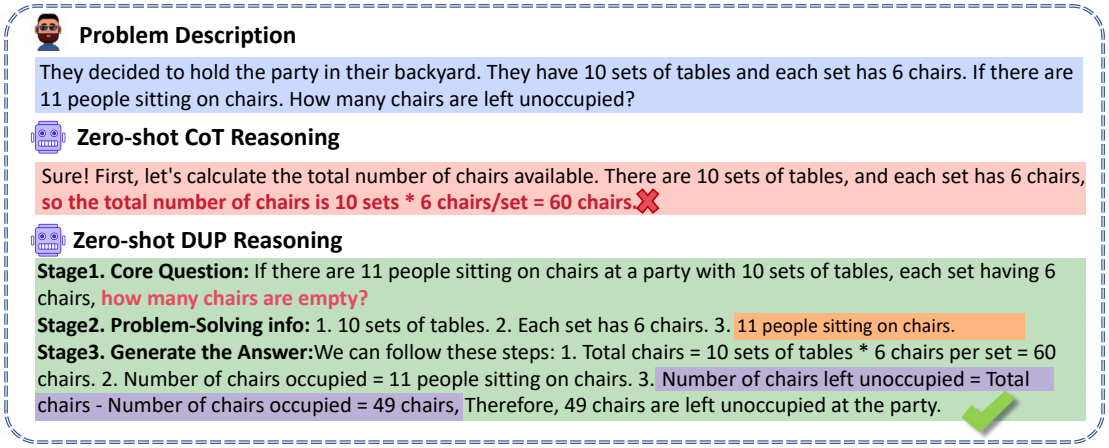

Figure 7: **Case study on SVAMP.** Zero-shot CoT fails to generate the correct answer, but our DUP method can make the correct prediction via deeply understanding the problems.

70b models (Touvron et al., 2023), CodeLLaMA-Instruct 13b and 34b models (Roziere et al., 2023). As seen in Table 6, in the cases of open-source LLMs, our DUP can still outperform the baseline zero-shot CoT by a large margin on GSM8K and AddSub benchmarks. This also proves the universality of our DUP method.

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, respectively. The contrastive results are illustrated in Figure 6. 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.

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. The detailed quantitative results are illustrated in Figure 5. As seen, compared with the baseline zero-shot CoT, our DUP reduces semantic misunderstanding effectively, indicating its effectiveness. 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 our DUP, the LLMs can better understand the problems and generate an accurate answer. More case studies on different benchmarks can be found in Appendix A.2.

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 prompting 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 prompting 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.

Limitations

DUP prompting generally requires three visits to LLMs, which indeed increases the inference costs. Although we attempt to merge the three stages of DUP as a single one, this approach would slightly lead to worse performance. We will further explore how to reduce the inference costs without losing any performance in future work.

Ethics and Reproducibility Statements

Ethics. We take ethical considerations very seriously and strictly adhere to the ACL Ethics Policy. This paper aims to improve the LLMs' reasoning abilities via a novel prompting strategy. All used models (or APIs) and datasets in this paper are publicly available and have been widely adopted by researchers. All experimental results upon these open models and datasets are reported accurately and objectively. Thus, we believe that this research will not pose any ethical issues.

Reproducibility. In this paper, we discuss the detailed experimental setup and provide enough information to re-product our results, such as all used prompts and inference settings. More importantly, ***we have provided our code in the supplementary materials*** to help reproduce the experimental results of this paper.

References

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, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. 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, et al. 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*.

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

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*.

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*.

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*.

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*.

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*.

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, et al. 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, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 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, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. 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. Metamath: 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*.

Qihuang Zhong, Liang Ding, Yibing Zhan, Yu Qiao, Yonggang Wen, Li Shen, Juhua Liu, Baosheng Yu, Bo Du, Yixin Chen, et al. 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*.

A Appendix

A.1 Prompt details.

We show the detailed prompts used in this work, covering the prompts for inference, extracting answers and error analysis. Specifically, Table 8 shows the inference template for all reasoning tasks. Tables 9, 10 and 11 list the prompts for extracting answers for Arithmetic Reasoning, Commonsense Reasoning and Symbolic Reasoning benchmarks, respectively. Moreover, the prompt used to categorize the failure examples is shown in Table 7.

A.2 More Case Studies

To have a close look, we provide more case studies for each dataset in this part, *i.e.*, AQuA (Table 12), GSM8K (Table 13), MultiArith (Table 14), SVAMP (Table 15), AddSub (Table 16), SingleEq (Table 17), CommonsenseQA (Table 18), StrategyQA (Table 19), and Coin Flip (Table 20). Specifically, taking the question of Table 12 as an example,

Template
Question: [*Input Question*]. **Wrong Response**: [*Wrong Answer*]. **Correct Response**: [*Correct Answer*]. 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 you select.

Table 7: **Prompts for error analysis.** The slot [*Input Question*] denotes the original input problem. The slots [*Wrong Question*] and [*Correct Question*] denote the incorrect text generated by the LLMs and the original label.

No.	Template	Reasoning tasks
1	**Extract core question**: Please extract core question, only the most comprehensive and detailed's one! **Extract problem-solving information** : Please extract the most useful information related to the core question([*Core Question*]), Only extract the most useful information, and list them one by one! **Generate the answer**: Hint: [*Problem-solving Info*], \n[*Core Question*]. \n Please understand core question and problem-solving information, then solve thequestion step by step and show the answer.	GSM8K, AddSub, SVAMP, MultiArith, SingleEq, AQuA, CSQA, StrategyQA, Coin Flip
2	**Prompt**: Please accurately understand the question useful information and solve the question step by step.	Last Letter

Table 8: **Reasoning prompt templates for all reasoning tasks**. Notably, [*Core Question*] indicates the extracted core question, and [*Problem-solving Info*] indicates the extracted problem-solving information to the problem.

we present the outputs of the three-stage processes of our DUP method, respectively. The extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in red. Please refer to the tables for more details.

No.	Template	Arithmetic Reasoning
1	Here is a math question and a model's answer about this question. Please extract the EXACT number from the answer text as the final answer for question. QUESTION: {}. \nANSWER: {} Final format should be a legal 'number' without any suffix such as '$'. The final answer is:	GSM8K, AddSub, SVAMP, MultiArith, SingleEq
2	Here is a math question and a model's answer about this question. Please extract the EXACT choice from the answer text as the final answer for question. QUESTION: {}. \nANSWER: {} Final format should be a legal 'options',If you can't find the right choice , just answer Z. The final answer is:	AQUA

Table 9: Prompts for extracting answers with `GPT-3.5-turbo` on **Arithmetic Reasoning**.

No.	Template	Commonsense Reasoning
1	Here is a Commonsense question and a model's answer about this question. Please extract the EXACT one choice from the answer text as the final answer for question. QUESTION: {}. \nANSWER: {} Final format should be a legal 'choice'(eg. (A) or (b)),If you can't find the correct choice, just answer the one that is closest to the answer. The final answer is:	CommonsenseQA
2	Here is a Commonsense question and a model's answer about this question. Please extract theEXACT one choice from the answer text as the final answer for question. QUESTION: {}. \nANSWER: {} Final format should be a legal 'string'(Yes or No), If you Uncertain or unknow, Please understand that the question and answer information outputs the closest answer,you can only output Yes or No. The final answer is:	StrategyQA

Table 10: Prompts for extracting answers with `GPT-3.5-turbo` on **Commonsense Reasoning**.

No.	Template	Symbolic Reasoning
1	Here is a question related to concatenating last letters, along with a model's answer to this question. Please extract the EXACT string from the answer text to determine the final answer to the question. QUESTION: {}. \nANSWER: {} Note: Please extract the string after (the answer is:), or the string that appears at the end of the sentence. The final answer is:	Last letters
2	Here is a Coin flip related question and a model's answer about this question. Please extract the EXACT string from the answer text as the final answer for question. QUESTION: {}. \nANSWER: {} Note: If you find that the coin is still heads up, the answer is "yes". If you find that the coin is not still heads up or tails up. the answer is "no". The final answer is:	Coin Flip

Table 11: Prompts for extracting answers with `GPT-3.5-turbo` on **Symbolic Reasoning**.

Q: A number is as much greater than 36 as is less than 86. Find the Number. Answer Choices: A)60 B)56 C)51 D)61 E)41

Stage1: Reveal the core question.
A: Please extract core question, only the most comprehensive and detailed one!
Output: A number is as much greater than 36 as is less than 86. Find the number.

Stage2: Extract the problem-solving information.
A: Note: Please extract the most useful information related to the core question (**A number is as much greater than 36 as is less than 86. Find the number**), only extract the most useful information, and list them one by one!
Output: 1. The number is as much greater than 36 as it is less than 86.

Stage3: Generate and extract the answers.
A: Hint: 1. The number is as much greater than 36 as it is less than 86.
A number is as much greater than 36 as is less than 86. Find the number.
Please understand the Hint and question information,then solve the question step by step and show the answer.
Output: The number that is as much greater than 36 as it is less than 86 can be found by calculating the average of 36 and 86. Average = (36 + 86) / 2 Average = 61. So, the number is 61, which is option D.

Table 12: **Example outputs by Zero-shot DUP for AQuA**. Notably, the extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in **red**.

Q: Morisette and Kael were asked to bring fruits. Morisette brought 5 apples and 8 oranges, while Kael brought twice the amount of apples and half the number of oranges than Morisette. How many fruits do they have in total?

Stage1: Extract the core question.
A: Please extract core question, only the most comprehensive and detailed one!
Output: How many fruits do Morisette and Kael have in total?

Stage2: Extract the problem-solving information.
A: Please extract the most useful information related to the core question (How many fruits do Morisette and Kael have in total?), only extract the most useful information, and list them one by one!
Output: -Morisette brought 5 apples and 8 oranges.\n- Kael brought twice the amount of apples and half the number of oranges than Morisette.

Stage3: Generate and extract the answers.
A: -Morisette brought 5 apples and 8 oranges.\n- Kael brought twice the amount of apples and half the number of oranges than Morisette.\nHow many fruits do Morisette and Kael have in total? \nPlease understand the Hint and question information,then solve the question step by step and show the answer.
Output: Morisette brought 5 apples + 8 oranges = 13 fruits.\nKael brought 2 * 5 apples = 10 apples.\nKael also brought 1/2 * 8 oranges = 4 oranges.\n So, Kael brought a total of 10 apples + 4 oranges = 14 fruits.\nTherefore, Morisette and Kael have 13 fruits + 14 fruits = 27 fruits in total.

Table 13: **Example outputs by Zero-shot DUP for GSM8K**. Notably, the extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in **red**.

Q: Each chocolate bar in a box cost $3. If a box had 7 bars total and Olivia sold all but 4 bars, how much money would she have made?

Stage1: Extract the core question.
A:Please extract core question, only the most comprehensive and detailed one!
Output: How much money would Olivia have made if she sold all but 4 bars from a box of 7 chocolate bars, where each bar costs $3?

Stage2: Extract the problem-solving information.
A: Please extract the most useful information related to the core question (How much money would Olivia have made if she sold all but 4 bars from a box of 7 chocolate bars, where each bar costs $3?), only extract the most useful information, and list them one by one!
Output:1. Each chocolate bar costs $3. 2. There are 7 bars in a box. 3. Olivia sold all but 4 bars.

Stage3: Generate and extract the answers.
A: Hint: 1. Each chocolate bar costs $3. 2. There are 7 bars in a box. 3. Olivia sold all but 4 bars. How much money would Olivia have made if she sold all but 4 bars from a box of 7 chocolate bars, where each bar costs $3?
Please understand the Hint and question information,then solve the question step by step and show the answer.
Output: Olivia sold all but 4 bars from a box of 7 chocolate bars, where each bar costs $3. So, Olivia sold 7 - 4 = 3 bars. The total amount of money Olivia would have made is 3 bars * $3 per bar = $9.

Table 14: **Example outputs by Zero-shot DUP for MultiArith**. Notably, the extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in red.

Q: Baker made 144 cakes. He sold 71 of them. Then he made 111 more cakes.How many more cakes did baker make than those he sold?

Stage1: Extract the core question.
A: Please extract core question, only the most comprehensive and detailed one!
Output: How many more cakes did the baker make than those he sold?

Stage2: Extract the problem-solving information.
A: Please extract the most useful information related to the core question (How many problems did Paige have to do for homework after she finished 44 of the problems at school, if she initially had 43 math problems and 12 science problems?), only extract the most useful information, and list them one by one!
Output: - Baker made 144 cakes.\n- He sold 71 of them.\n- He made 111 more cakes.

Stage3: Generate and extract the answers.
A: Hint: - Baker made 144 cakes.\n- He sold 71 of them.\n- He made 111 more cakes. \nHow many more cakes did the baker make than those he sold? \nPlease understand the Hint and question information,then solve the question step by step and show the answer.
Output: The baker initially made 144 cakes and then made an additional 111 cakes, so in total, he made 144 + 111 = 255 cakes. He sold 71 cakes. Therefore, the baker made 255 - 71 = 184 more cakes than he sold.

Table 15: **Example outputs by Zero-shot DUP for SVAMP**. Notably, the extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in red.

Q: Craig walked 0.2 mile from school to David 's house and 0.7 mile from David 's house to his own house . How many miles did Craig walk in all?

Stage1: Extract the core question.
A: Please extract core question, only the most comprehensive and detailed one!
Output: How many miles did Craig walk in all?

Stage2: Extract the problem-solving information.
A: Please extract the most useful information related to the core question (How many miles did Craig walk in all?), only extract the most useful information, and list them one by one!
Output: - Craig walked 0.2 mile from school to David 's house.\n- Craig walked 0.7 mile from David 's house to his own house.

Stage3: Generate and extract the answers.
A: Hint: - Craig walked 0.2 mile from school to David 's house.\n- Craig walked 0.7 mile from David 's house to his own house. \nHow many miles did Craig walk in all? \nPlease understand the Hint and question information,then solve the question step by step and show the answer.
Output: Craig walked a total of 0.2 mile + 0.7 mile = 0.9 mile.

Table 16: **Example outputs by Zero-shot DUP for AddSub**. Notably, the extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in red.

Q: Dana earns $13 per hour. She worked 9 hours on Friday, 10 hours on Saturday, and 3 hours on Sunday. How much money did Dana earn in all?

Stage1: Extract the core question.
A: Please extract core question, only the most comprehensive and detailed one!
Output: How much money did Dana earn in total if she earns $13 per hour and worked 9 hours on Friday, 10 hours on Saturday, and 3 hours on Sunday?

Stage2: Extract the problem-solving information.
A: Please extract the most useful information related to the core question (How much money did Dana earn in total if she earns $13 per hour and worked 9 hours on Friday, 10 hours on Saturday, and 3 hours on Sunday?), only extract the most useful information, and list them one by one!
Output: 1. Dana earns $13 per hour.\n2. She worked 9 hours on Friday.\n3. She worked 10 hours on Saturday.\n4. She worked 3 hours on Sunday.

Stage3: Generate and extract the answers.
A: Hint: 1. Dana earns $13 per hour.\n2. She worked 9 hours on Friday.\n3. She worked 10 hours on Saturday.\n4. She worked 3 hours on Sunday. \nHow much money did Dana earn in total if she earns $13 per hour and worked 9 hours on Friday, 10 hours on Saturday, and 3 hours on Sunday? \nPlease understand the Hint and question information,then solve the question step by step and show the answer.
Output: Dana worked a total of 9 hours on Friday + 10 hours on Saturday + 3 hours on Sunday = 22 hours. Since Dana earns $13 per hour, for the 22 hours she worked, she earned 22 hours * $13/hour = 286.

Table 17: **Example outputs by Zero-shot DUP for SingleEq**. Notably, the extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in red.

Q: The artist was sitting quietly pondering, then suddenly he began to paint when what struck him? Answer Choices: (A)sadness (B)anxiety (C)inspiration (D)discomfort (E)insights

Stage1: Extract the core question.
A: Please extract core question, only the most comprehensive and detailed one!
Output: What struck the artist and prompted him to start painting?

Stage2: Extract the problem-solving information.
A: Please extract the most useful information related to the core question (What struck the artist and prompted him to start painting?), only extract the most useful information, and list them one by one!
Output: - The artist was sitting quietly pondering\n- Suddenly, he began to paint\n- Something struck him and prompted him to start painting

Stage2: Generate and extract the answers.
A: Hint: - The artist was sitting quietly pondering\n- Suddenly, he began to paint\n- Something struck him and prompted him to start painting \n3. She worked 10 hours on Saturday.\n4. She worked 3 hours on Sunday. \nWhat struck the artist and prompted him to start painting? \nPlease understand the Hint and question information,then solve the question step by step and show the answer.
Output: Based on the information provided, the answer is (C) inspiration. The artist was sitting quietly pondering, and suddenly something struck him, prompting him to start painting. This suggests that he was inspired by something, perhaps an idea, a vision, or a feeling, which motivated him to express himself through painting.

Table 18: **Example outputs by Zero-shot DUP for CommonsenseQA**. Notably, the extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in red.

Q: Can the original name of the zucchini be typed on the top row of a QWERTY keyboard?

Stage1: Extract the core question.
A: Please extract core question, only the most comprehensive and detailed one!
Output: Can the original name of the zucchini be typed on the top row of a QWERTY keyboard?

Stage2: Extract the problem-solving information.
A: Please extract the most useful information related to the core question (Can the original name of the zucchini be typed on the top row of a QWERTY keyboard?), only extract the most useful information, and list them one by one!
Output: 1. The original name of the zucchini is “Cucurbita pepo.” \n2. The top row of a QWERTY keyboard contains the letters “QWERTYUIOP.” \n3. The original name of the zucchini cannot be typed using only the top row of a QWERTY keyboard.

Stage3: Generate and extract the answers.
A: Hint: 1. The original name of the zucchini is “Cucurbita pepo.” \n2. The top row of a QWERTY keyboard contains the letters “QWERTYUIOP.” \n3. The original name of the zucchini cannot be typed using only the top row of a QWERTY keyboard. \nCan the original name of the zucchini be typed on the top row of a QWERTY keyboard? \nPlease understand the Hint and question information,then solve the question step by step and show the answer.
Output: No, the original name of the zucchini “Cucurbita pepo” cannot be typed on the top row of a QWERTY keyboard. The top row of a QWERTY keyboard contains the letters “QWERTYUIOP” and none of these letters are present in the original name of the zucchini. Therefore, it is not possible to type the original name of the zucchini using only the top row of a QWERTY keyboard.

Table 19: **Example outputs by Zero-shot DUP for StrategyQA**. Notably, the extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in red.

Q: A coin is heads up. Walter does not flip the coin. Mikayla does not flip the coin. Larry does not flip the coin. Ryan does not flip the coin. Is the coin still heads up? Note that "flip" here means "reverse".

Stage1: Extract the core question.
A: Please extract core question, only the most comprehensive and detailed one!
Output: Is the coin still heads up if no one flips it?

Stage2: Extract the problem-solving information.
A: Please extract the most useful information related to the core question (Is the coin still heads up if no one flips it?), only extract the most useful information, and list them one by one!
Output: 1. The initial position of the coin is heads up.\n2. Walter does not flip the coin.\n3. Mikayla does not flip the coin.\n4. Larry does not flip the coin.\n5. Ryan does not flip the coin.

Stage3: Generate and extract the answers.
A: Hint: 1. The initial position of the coin is heads up.\n2. Walter does not flip the coin.\n3. Mikayla does not flip the coin.\n4. Larry does not flip the coin.\n5. Ryan does not flip the coin. \nIs the coin still heads up if no one flips it? \nPlease understand the Hint and question information,then solve the question step by step and show the answer.
Output: Based on the given information, we can conclude the following: \n1. The initial position of the coin is heads up.\n2. Walter does not flip the coin. Therefore, the coin remains heads up.\n3. Mikayla does not flip the coin. Therefore, the coin remains heads up.\n4. Larry does not flip the coin. Therefore, the coin remains heads up.\n5. Ryan does not flip the coin. Therefore, the coin remains heads up.Since no one flips the coin, the coin remains heads up.

Table 20: **Example outputs by Zero-shot DUP for Coin Flip**. Notably, the extracted core question and key problem-solving information are highlighted in blue and orange. The final answer is highlighted in red.