

Enhancing Multi-Step Mathematical Reasoning in Large Language Models with Step-by-Step Similarity Prompts and Answer Voting

Qi Ye^{1[0000-0003-2512-1266]}, Xiang Ji², RuiHui Hou³, JingPing Liu⁴ and Tong Ruan⁵,

¹East China University of Science and Technology, 130 Meilong Road, Shanghai, China,
yeh_qi1125@ecust.edu.cn

² East China University of Science and Technology, 130 Meilong Road, Shanghai, China,
jx3315722867@163.com

Abstract. Complex reasoning problems, especially multi-step mathematical reasoning problems, are a difficult class of NLP tasks to solve. Existing methods such as Manual-CoT improve the accuracy of reasoning tasks by manually designing prompts to allow large models to output reasoning paths. However, the quality of the inference steps generated by this method is not high, resulting in many calculation and planning errors. To address the problems, we propose a method that combines similar enhanced step-by-step prompts with an answer voting mechanism. Specifically, we first design a comprehensive prompt template that integrates task prompts, CoT prompts, and format prompts, and then use two similar templates to guide the Large Language Model in generating better inference paths. Furthermore, we use ChatGLM for efficient information retrieval and determine the most accurate answer through a majority voting system. We evaluated our method in five mathematical datasets and one symbolic dataset. The experimental results over GPT-3 show that our proposed method outperforms Zero-shot-CoT and Zero-shot-Program-of-Thought Prompting across all datasets by a large margin of 7.3% and 4.4% respectively, and exceeds Plan-and-Solve in five of six datasets. In particular, on symbolic data sets, our method completely outperforms all comparable methods by a large margin of an average of 13%. Our code and data are publicly available at <https://anonymous.4open.science/r/ESPDE-2740>.

Keywords: Mathematical reasoning · CoT · Similar prompt

1 Introduction

With the advancements in generative large-scale language models, their performance has witnessed remarkable progress in numerous natural language processing (NLP) tasks. Among these tasks, mathematical reasoning holds particular significance, as it serves as a benchmark for evaluating the reasoning abilities of language models.

There are two types of existing methods for solving reasoning problems. One

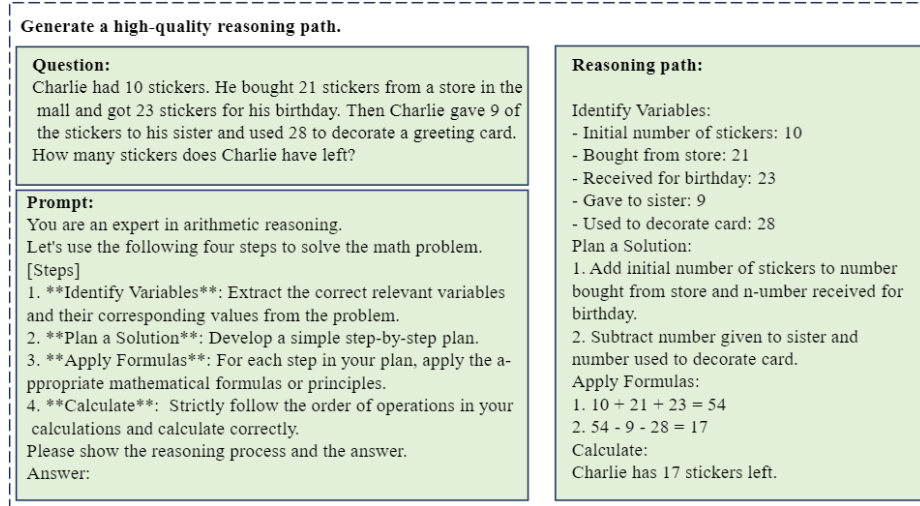


Fig. 1. Example of using our prompt template to generate a high-quality reasoning path. In the figure, the left part is input that is constructed by a question and prompt template, and the right part is the reasoning path generated by LLM.

utilizes the reasoning path output by large models to construct a fine-tuning instruction set, which is implemented by fine-tuning the model. However, this type of method requires the additional construction of a fine-tuning instruction set and is limited by the number of parameters in the model. The effect of fine-tuning may not be as good as directly using prompts. The other is to improve the quality of the reasoning path output of large models by optimizing the prompts, which is more effective and convenient. Therefore, many works are performed using this type of method. For example, considering enhancing the inferential capabilities of a model cannot be solely achieved by increasing its parameter count [12, 2], researcher [21] introduced a novel approach Manual-CoT, which analyzes thought chains to uncover answers, offering a promising avenue to bolster the reasoning capabilities of large language models. CoT-based approaches have been instrumental in tackling complex and multi-step mathematical reasoning tasks. However, Manual-CoT relies on human-designed prompts, and the quality of these prompts directly impacts the inference performance. To alleviate the reliance on manual effort, [8] discovered that appending a prompt such as “Let’s think step by step” after the problem, without utilizing any examples, yielded impressive results. Furthermore, [22] proposed the Auto-CoT approach, which also eliminates the need for manual prompt construction. Although previous methods have improved the model’s ability to solve multi-step reasoning problems to some extent, computational errors, missing steps, and misinterpretation of problem statements still persist. To address this, [19] introduced the Plan-and-Solve method [19], they use specific phrases such as “pay attention to calculations” in the prompts to help the model reduce specific types of errors. While the method has enhanced certain aspects of performance, we observed that it

does not completely address calculation inaccuracies and misunderstandings of questions in our experiments. These calculation errors arise primarily from issues in variable extraction, formula inaccuracies or omissions, and basic arithmetic errors.

Inspired by the decomposition task thought of Plan-and-Solve, we refined the prompt template by segmenting it into multiple distinct points, thereby enhancing the clarity of its structure. To further reduce computational errors, we introduced formula application steps and incorporated additional information for each subsection within the template. Moreover, research [1] indicates that minor variations in prompts significantly affect the performance of large language models (LLMs). Leveraging this knowledge, we developed two semantically similar yet content-diverse prompt templates to induce a broader range of inference pathways in LLMs. Based on these, we introduce an innovative methodology termed enhanced stepwise prompt and diverse path exploration (ESPDE). This approach begins with the reconfiguration of the stepwise prompt template, which is segmented into three distinct components: task prompt, CoT prompt, and format prompt. As shown in figure 1, we can obtain a standardized high-quality path that contain no missing steps and are generated in strict accordance with the steps outlined in our defined cot prompt by concatenating the problem and prompt together. Subsequently, we feed two similar prompt templates, along with the question, into Large Language Models (LLMs) to generate a reasoning pathway. Utilizing both LLMs and the Python re-module, we meticulously extract potential answers from these paths. The process culminates in the identification of the most accurate response through a majority voting mechanism, thereby enhancing the precision and reliability of the outcomes. Our methodology was rigorously assessed across five mathematical datasets and a symbolic dataset. Additionally, to test our method’s adaptability and efficiency in varied contexts, we also conducted experiments on another instruct LLM model, gpt-3.5-turo-instruct¹. The extensive experiments prove the efficiency of our method.

Our key contributions are the following:

1. We propose a novel framework for solving mathematical reasoning problems based on a similar enhanced prompt template, an accurate answer extraction method, and a voting mechanism.
2. Experimental results on public datasets prove the effectiveness of our solution. Furthermore, the result demonstrates that our method achieves state-of-the-art (SOTA) performance on five of six datasets. On average, it exceeds the previous method by 1.2%.
3. To verify the robustness of the method, we also experiment on a symbolic dataset and the result shows that our method completely outperforms all comparable methods.

¹ <https://platform.openai.com/docs/models/gpt-3-5-turbo>

2 Related Work

Related work in this paper can be divided into two parts, the CoT prompting method and the mathematical reasoning task. The former is a typical method to solve multi-step reasoning problems, while the latter is the description of a specific multi-step task which is mathematical reasoning

2.1 Mathematical Reasoning Task

Mathematical reasoning tasks have been a focal point in the advancement of artificial intelligence and natural language processing (NLP), challenging models to understand and solve complex mathematical problems using natural language. This area intersects with both cognitive science, in understanding how humans solve mathematical problems, and computer science, in developing algorithms that can replicate or assist in these processes. There have been studies in understanding NLP models' capabilities [7, 9, 13, 10, 14] to solve arithmetic/algebraic questions. A notable approach in this field has been the use of large language models (LLMs) like GPT-3, developed by OpenAI², which have shown promising results in solving arithmetic and algebraic problems through natural language understanding and generation. These models leverage vast amounts of textual data to learn patterns and problem-solving strategies that can be applied to mathematical reasoning. Our work aims to use a similar enhanced prompt template to make LLM generate high-quality reasoning paths, enhance the inference ability of LLM, and improve accuracy on mathematical inference datasets

2.2 CoT Prompting

With the emergence of large language models, more and more researchers are exploring the use of CoT [21] prompting to stimulate the logical reasoning ability of large models and help solve complex reasoning problems. CoT prompting is a technique that allows large language models (LLMs) to solve a problem as a series of intermediate steps before giving a final answer. Standard CoT requires input examples to execute, which is in few-shot setting. Many works [15–18] try to improve CoT prompting to make LLM generate high-quality reasoning paths. [8] proposed a method that does not require input examples, using a "universal" trigger sentence "Let's think step by step" to make LLM generate reasoning chains. Additionally, there are some works that attempt to automatically generate reasoning steps to eliminate human efforts in a prompt design, such as Auto-CoT [23]. It first automatically obtains k examples by clustering the given dataset and then generates rationales for the selected examples. Finally, demonstration examples are constructed by adding the generated rationales to selected examples as CoT prompts. PoT [4] is a unique LLM reasoning method. It is not just about generating natural language answers, but also requires the cre-

² <https://platform.openai.com>

ation of an executable program that can be run on program interpreters such as Python to produce actual results. Plan-and-Solve [19] is a method by focuses on

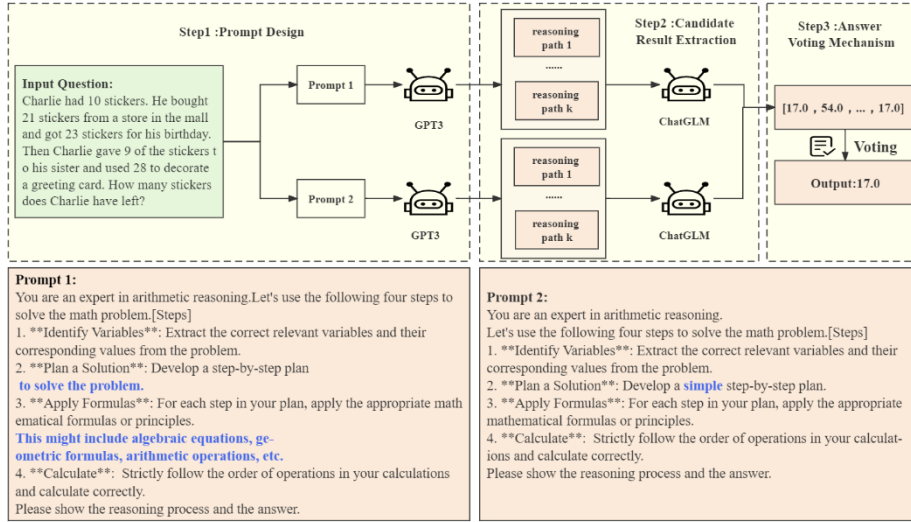


Fig. 2. The main framework and prompt template of our method. The prompt 1 and 2 are a pair of similar prompts and their difference have been labeled in blue color. The GPT3 represents text-davinci-003 or gpt-3.5-turbo-instruct.

eliciting multi-step reasoning by LLMs in a zero-shot setting. They ask LLMs to write a plan to decompose a complex reasoning task into multiple reasoning steps. Our method draws on their ideas of decomposing problems and expands on this basis. Specifically, we first design a comprehensive prompt template that integrates task prompts, CoT prompts, and format prompts, and then use two similar templates to guide the Large Language Model(LLM) in generating better inference paths. Additionally, we use ChatGLM for efficient information retrieval and determine the most accurate answer through a majority voting system. We refer readers to the survey [5]for more related works.

3 Methodology

Overview In this study, we introduce ESPDE, a method that significantly improves LLMs’ capacity for generating coherent reasoning paths and accurately predicting answers. The framework is shown in figure 2. For a given problem, we initially craft two semantically similar prompts by hand. Using these, the LLM is instructed to generate k inference paths for each prompt. Using ChatGLM, we extract specific answers from both sets of paths. Ultimately, we employ a voting mechanism to determine the optimal answer from the aggregated responses.

3.1 Prompt Design

This section explains the process of prompt template construction. Given a question input, we design two similar semantic prompts for the LLM to output k candidate paths based on them, which will contain the final answers. Studies have shown the significant impact of slight variations in prompts on the performance of large language models (LLMs) [1]. These models exhibit distinct representations in response to minor differences in input, thereby influencing output results. Building on this insight, we design two prompt templates with similar semantics but distinct content. Specifically, as depicted in figure 2, prompt 1 comprises three integral components: task prompt, CoT prompt, and format prompt. The task prompt, delineated by a task description and the trigger sentence “Let’s use the following four steps to solve the problem. ’, directs the attention of the large language model (LLM) toward specific domain knowledge, facilitating adherence to the sequence outlined in the CoT prompt. Central to this template is the CoT prompt, which serves as a guide for the LLM in problem-solving endeavors. For mathematical reasoning tasks, the CoT prompt encompasses four key steps: “Identify Variables”, “Plan a Solution”, “Apply Formulas”, and “Calculate”. Figure 2 illustrates the inclusion of explanatory details within each step, helping the LLM to focus on crucial elements and mitigating the impact of calculation errors from multiple angles. This approach fosters a deeper comprehension of the problem context, thereby enhancing the LLM’s reasoning capabilities. Meanwhile, the format prompt “Please show the reasoning process and the answer.” ensures that the LLM generates clear, step-by-step reasoning paths in our experiments. In addition, as shown in Figure 2, Prompt 2 has reduced some content compared to Prompt 1, but the two are still semantically similar, and their differences are highlighted in blue.

In the case of symbolic reasoning tasks, we adopt the Plan-and-Solve methodology [19]. The CoT prompt involves steps such as “Plan a Solution” and “Solve the problem”, complemented by explanatory content. The task prompt and format prompt are almost identical. In a word, by appending our prompt template with specific questions and inputting them into the LLM, we aim to produce two pairs of high-quality reasoning paths.

3.2 Candidate Result Extraction

During the answer extraction phase, given 2k reasoning paths, we leverage ChatGLM (6B) and GPT-3 (175B) to extract responses from mathematical and symbolic reasoning paths, respectively. Specifically, we designed different prompts for different answer types. For multiple choice questions, our method employs the prompt “Please extract the answer option from the text. The response format should be “ The answer is xxx.”. Just reply option. Example: The answer is (A) xx.” for extracting responses from reasoning paths. For the answer to the number and the symbolic answer, we use the prompt “Please extract the answer from the text. The answer is xxx.”. Submitting these prompts alongside

the relative reasoning path to ChatGLM yields the desired outcomes. Following this, we utilize Python’s regular expression module to further refine extracted values. The rationale behind employing the distinct model lies in the ease and cost-effectiveness of using ChatGLM. In contrast, accurately extracting string-based answers for symbolic reasoning poses a substantial challenge. Based on empirical findings, we opt for the text-davinci-003³ model to enhance the effectiveness of extracting such answers, showcasing commendable performance. The pertinent experimental outcomes are presented in section 5.2. In a word, after the candidate result extraction stage, we will obtain 2k answers for one question.

3.3 Answer Voting Mechanism

In this section, we will get the final answer from all 2k answers. Given 2k answers, we use a voting strategy based on the self-consistency [20] characteristics of the answer to get the final answer. Specifically, we employ the majority vote rule, a prevalent voting system wherein the candidate answer must garner the most votes to be deemed the correct one. This method is employed in the election and decision-making processes of numerous countries. Let v_i represent the number of votes for the i -th candidate answer, out of a total of 2k candidate answers. The condition for candidate answer a to win can be expressed as:

$$\left\{ \begin{array}{l} v_a > v_i \\ \text{for all } i \neq a, \\ \text{where } i, a \in \{1, 2, \dots, 2k\} \end{array} \right. \quad (1)$$

This formula states that candidate a is declared the winner if their vote count v_a is higher than that of any other candidate across the entire set of candidates participating in the election. And v_a is the definitive answer.

4 Experimental Setup

In this section, we detail our experimentation process, which covers dataset selection, comparison with baseline methods, and our customized implementation approach.

4.1 Datasets

To make a comprehensive evaluation of the proposed method, in this paper, we use multiple datasets on both mathematical reasoning and symbolic reasoning tasks. The metric we used for mathematical and symbolic reasoning tasks is accuracy(%).

For **arithmetical task**, we conduct a series of experiments on five different datasets, including (1) AddSub [7] dataset of addition and subtraction arith-

³ <https://platform.openai.com/docs/deprecations>

metic word problems, (2) GSM8K [6] dataset of high quality linguistically diverse grade school math word problems created by human problem writers, (3) MultiArith [13] dataset consists high-quality question description and formulas, (4) SVAMP [11] a challenge set for elementary-level Math Word Problems and (5) AQuA-RAT [10] dataset that contains algebraic word problems with rationales.

For **symbolic reasoning task**, we use the Last Letter Concatenation [21] dataset while the question is to splice the last letter of each word in the sentence. These datasets are classical reasoning datasets that many other methods also use. We did not consider another CoinFlip [21] dataset, as its problem is relatively simple and many methods have achieved performance exceeding 99%. Therefore, we chose the more difficult Last Letter dataset.

4.2 Baselines

We compare ESPDE with the following five baselines: Zero-shot-CoT [8], Zero-shot-PoT [4], Manual-CoT [21], Auto-CoT [22] and Plan-and-Solve [19]. Zero-shot-CoT simply adds "Let's think step by step" after each question as a prompt. Zero-shot-PoT uses Codex to express the reasoning process as a program and execute it by the computer to obtain the answer. Manual-CoT uses eight artificially constructed prompts to elicit reasoning in LLM. Auto-CoT samples questions with diversity and generates reasoning chains to construct demonstrations. Plan-and-Solve divides the prompt into two parts: first, devise a plan to break the task into subtasks and then carry out the subtasks according to the plan.

4.3 Implementation

We divide the implementation plan into two steps: the first step is to generate the reasoning paths, and the second step is to extract answers from the reasoning paths. In the first step, we use the text-davinci-003 [3] model as the backbone, which is the version of the public ChatGPT model from the OpenAI API with 175 billion parameters. We select this model because it is more capable than the text-davinci-002 [3] version of GPT-3 and is designed specifically for instruction-following tasks. Additionally, it is widely used by most other reasoning methods including all the baselines compared in this paper. For parameter settings, we set the temperature to 0.25, top p to 1 to reduce the randomness of generated reasoning paths, and set the n to 5 to get 5 paths from the decoder of the model for each sample. In the second step, we use ChatGLM with 6 billion parameters and follow default settings.

However, according to the latest announcement from OpenAI, the text-davinci-003 will be abandoned on January 4, 2024. To establish the generality and applicability of our method, we performed experiments on mathematical datasets using gpt-3.5-turbo-instruct, which is the latest recommended alternative to the text-davinci-003 model, as suggested by OpenAI.

Table 1. Accuracy (%) comparison of ESPDE with five baselines on five mathematical reasoning datasets. The five baselines belong to zero-shot or few-shot settings. The ESPDE* means that the result comes from gpt-3.5-turo-instruct. The best result is bold in the zero-shot setting. The last column is average scores. Notably, each result is the average of multiple experiments. The '-' means that the experiment result cannot be reproduced anymore due to the extraction model having been deprecated.

Methods	Arithmatical					Symbolic	
	AddSu b	AQu A	GSM8K	Multi- Arith	SVAMP	AVG	LastLetter
*few-shot							
Manual-CoT [21]	91.6	48.4	58.4	93.6	80.3	74.5	70.6
Auto-CoT [22]	90.8	41.7	57.1	95.5	78.1	72.6	-
*zero-shot							
CoT [8]	85.3	38.9	56.4	83.8	69.9	66.9	64.8
PoT [4]	85.1	43.9	57.0	92.2	70.8	69.8	-
Plan-and-Solve [19]	92.2	46.0	59.3	91.8	75.7	73.0	75.2
ESPDE (ours)	92.9	44.5	60.7	94.1	78.5	74.2	83.5
ESPDE* (ours)	87.4	53.0	72.8	96.2	82.5	78.4	-

5 Experimental Results

In this section, we showcase our experimental findings for two tasks. Furthermore, we compare prompts across various methods, assess the efficacy of various extraction techniques, and perform an error analysis on 100 samples from the GSM8K dataset.

5.1 Main Results

In the experiments, we evaluated ESPDE on six datasets of two categories of reasoning tasks. Table 1 and 3 show the experimental results, which are overall better than the existing sota method and far exceed other baseline methods.

Arithmetic Reasoning. Table 1 summarizes comparisons between our method and five baselines on arithmetic reasoning datasets. The result shows that in zero-shot setting, our approach completely surpasses previous methods and reaches state-of-the-art on four datasets except AQuA-RAT. Specifically, our ESPDE achieves score gains of 0.7%, 1.4%, 2.3%, and 2.8% over the previous state-of-art method on AddSub, GSM8K, MultiArith, and SVAMP, respectively. Additionally, our method completely outperforms CoT and PoT on all five datasets, with an average improvement of 7.3% and 4.4% respectively, indicating its superior performance. Compared with another competitive method, Plan-and-Solve, the performance of our approach is also competitive. The results show that our method outperforms Plan-and-Solve on four of the five math datasets, with an

average improvement of 1.2%. The exception of the result on AQuA-RAT could be due to the dataset itself which we find there are mistakes in the dataset and also the dataset is more challenging.

In few-shot setting, our zero-shot ESPDE substantially outperforms the Auto-CoT on four arithmetic reasoning tasks except for MultiArith in table 1 and improves by an average of 1.6%. Compared with Manual-CoT, our method (74.2%) outperforms in three out of five datasets, averaging only 0.3% lower than Manual-CoT (74.5%). While this is an unfair comparison, the result shows that even compared with some few-shot methods our method still performs well. Importantly, we also conducted experiments on another model, referred to as Model gpt-3.5-turbo-instruct. As shown in table 1, experimental results far exceed previous methods on all math datasets except AddSub. On average, it is improved by 4.2% compared to using text-davinci-003 version in our method. The experimental results further confirmed the versatility and adaptability of our method. The positive results obtained with gpt-3.5-turbo-instruct validate the efficacy and robustness of our approach across different models. This reinforces the notion that our method exhibits excellent flexibility and can be applied successfully in various contexts.

Symbolic Reasoning. Table 1 also shows the results of ESPDE against Zero-shot-CoT, Plan-and-Solve and Manual-CoT on symbolic reasoning dataset. It is clear that our method significantly outperforms previous methods over the Last Letter dataset. Our ESPDE achieves respective improvements of 18.7% over Zero-Shot-CoT, 8.3% over Plan-and-Solve, and 12.9% over Manual-CoT

5.2 Detailed Analysis

Comparison results of Different Prompts. To verify the effectiveness of our prompt template, we conduct experiments using the GPT-3.5-turbo-instruct model to compare our prompt with two other methods, Zero-Shot-CoT and Plan-and-Solve, on the MultiArith dataset. The results, shown in table 2, demonstrate that our prompt (96.2%) outperformed the other two methods, achieving scores of 91.8% and 89.8%, respectively. This significant performance difference confirms the effectiveness of our method

For citations of references, we prefer the use of square brackets and consecutive numbers. Citations using labels or the author/year convention are also acceptable. The following bibliography provides a sample reference list with entries for journal articles [1], an LNCS chapter [2], a book [3], proceedings without editors [4], as well as a URL [5].

Furthermore, we sought to explore the impact of prompts by analyzing variations at different stages in our prompt construction process. Initially, we broke down the specific problem-solving steps, providing the model with more granular information. Subsequently, we augmented the prompt with additional specific descriptions. Finally, we incorporated formulas to create the final prompt. As illustrated in the table, each successive modification step led to an improvement in performance. Initially, the model achieved a score of 95.6%, which increased to

96% after incorporating more specific description information into the prompt. Finally, by including relevant formulas, the performance further improved to a score of 96.2%. By systematically refining the prompts, we effectively harnessed the potential of the language model, leveraging its capabilities to deliver superior results. This observation highlights the importance of crafting appropriate prompts to enable large language models to produce higher-quality results and arrive at accurate answers.

Table 2. Comparison of the accuracy (%) of prompts from different methods and the construction of prompt of our method. In the table, No.1 and No.2 mean the prompt of Zero-shot-CoT and Plan-and-Solve respectively. And No.3 to No.5 is the comparison of our trigger sentences. The maximum value is bold.

No.	Prompt	MultiArith
1	Let’s think step by step.	91.8
2	Let’s first understand the problem, extract relevant variables and their corresponding numerals, and make a complete plan. Then, let’s carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and common-sense), solve the problem step by step, and show the answer.	89.8
3	**Identify Variables** : Extract the correct relevant variables and their corresponding values from the problem. **Plan a Solution** : Develop a step-by-step plan to solve the problem. **Calculate** : Calculate correctly.	95.6
4	**Identify Variables** : Extract the correct relevant variables and their corresponding values from the problem. Develop a step-by-step plan to solve the problem. **Calculate** : Strictly follow the order of operations in your calculations and calculate correctly	96
5	**Identify Variables** : Extract the correct relevant variables and their corresponding values from the problem. **Plan a Solution** : Develop a step-by-step plan to solve the problem. **Apply Formulas** : For each step in your plan, apply the appropriate mathematical formulas or principles. This might include algebraic equations, geometric formulas, arithmetic operations, etc. **Calculate** : Strictly follow the order of operations in your calculations and calculate correctly	96.2

Comparison results of Different Extraction Models. For cost and efficiency considerations, we compared the extraction performance of two models, text-davinci-003 (175B) and ChatGLM (6B), on AddSub and Last Letters datasets, respectively. The former model is a paid option with a greater number of parameters, whereas the latter is free but has fewer model parameters, suggesting lower capabilities. Based on Table 3, the performance of ChatGLM (98.7%) and text-davinci-003 (99.5%) is similar to the mathematical reasoning

dataset. However, when it comes to the symbolic reasoning dataset, ChatGLM (45.6%) significantly underperforms compared to text-davinci-003 (99.6%). As a result, we have implemented distinct models for each dataset type, allowing us to reduce costs without compromising the overall outcomes.

Table 3. Extraction accuracy (%) comparison of ChatGLM with text-davinci-003 on Last Letters and AddSub dataset. The best result is bold.

Models	Last Letter	AddSub
ChatGLM(6B)	45.6	98.7
Text-Davinci-003(175B)	99.6	99.5

Table 4. Error distribution rate comparison between Plan-and-Solve and ESPDE on GSM8K’s 100 data points. The PS+ means the Plan-and-Solve method and the lower error rate is bold.

Error Type	PS+	ESPDE
Variable extraction error	11%	8%
Plan error	12%	13%
Calculation error	15%	9%

Error Analysis. We evaluated our method’s effectiveness by comparing its error distributions with Plan-and-Solve on the GSM8K dataset. Specifically, we randomly sampled 100 data points, generated thought chains with both methods’ prompts respectively, and identified error examples. Then we manually analyzed the error samples and classified the errors into three types: variable extraction, plan, and calculation. Table 4 shows the error distributions. As shown in table 4, our method significantly outperforms Plan-and-Solve and reduced variable extraction and calculation errors by 3% and 6%, respectively. Furthermore, it shows a similar performance to Plan-and-Solve in planning errors with only 1% lower. The result may be attributed to the complexity of some questions, which pose challenges for both human and machine comprehension, and the incompleteness of the plan, which reflects the limitations of the model

6 Conclusion

In this paper, we focus on mathematical reasoning tasks and extend to symbolic reasoning tasks. We studied how to make LLM generate high-quality inference paths and extract accurate answers from paths, and propose ESPDE to solve the two reasoning tasks. ESPDE leverages two similar comprehensive prompt templates that integrate task prompts, CoT prompts, and format prompts to guide LLM to generate high-quality reasoning paths. Then use ChatGLM for efficient and economical answer extraction, and determine the most accurate answer through a majority voting system. Evaluation on six datasets across two

types of reasoning problems shows ESPDF outperforms the previous zero-shot baselines and performs comparable with few-shot CoT prompting on multiple arithmetic reasoning datasets.

We compare the prompts of different methods and the result shows the proposed prompt template can generate a high-quality reasoning process. We also make error analysis on 100 random samples of GSM8K and the result shows that our method indeed reduces calculation errors and variable extraction errors. Furthermore, the result of the experiment on expanded model gpt-3.5-turbo-instruct shows the robustness of our method. However, we cannot solve the plan errors efficiently, which may be limited to the performance of the model. We leave the problem as our future work.

References

1. Arora, S., Narayan, A., Chen, M.F., Orr, L., Guha, N., Bhatia, K., Chami, I., Re, C.: Ask me anything: A simple strategy for prompting language models. In: The Eleventh International Conference on Learning Representations (2023), <https://openreview.net/forum?id=bhUPJnS2g0X>
2. bench authors, B.: Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research* (2023), <https://openreview.net/forum?id=uyTL5Bvosj>
3. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) *Advances in Neural Information Processing Systems*. vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020)
4. Chen, W., Ma, X., Wang, X., Cohen, W.W.: Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research* (2023), <https://openreview.net/forum?id=YfZ4ZPt8zd>
5. Chu, Z., Chen, J., Chen, Q., Yu, W., He, T., Wang, H., Peng, W., Liu, M., Qin, B., Liu, T.: A survey of chain of thought reasoning: Advances, frontiers and future (2023)
6. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., Schulman, J.: Training verifiers to solve math word problems. *ArXiv abs/2110.14168* (2021), <https://api.semanticscholar.org/CorpusID:239998651>
7. Hosseini, M.J., Hajishirzi, H., Etzioni, O., Kushman, N.: Learning to solve arithmetic word problems with verb categorization. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp. 523–533. Association for Computational Linguistics, Doha, Qatar (Oct 2014). <https://doi.org/10.3115/v1/D14-1058>, <https://aclanthology.org/D14-1058>

8. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. ArXiv abs/2205.11916 (2022), <https://api.semanticscholar.org/CorpusID:249017743>
9. Koncel-Kedziorski, R., Hajishirzi, H., Sabharwal, A., Etzioni, O., Ang, S.D.: Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics* 3, 585–597 (2015)
10. Ling, W., Yogatama, D., Dyer, C., Blunsom, P.: Program induction by rationale generation: Learning to solve and explain algebraic word problems. In: Barzilay, R., Kan, M.Y. (eds.) *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 158–167. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). <https://doi.org/10.18653/v1/P17-1015>, <https://aclanthology.org/P17-1015>
11. Patel, A., Bhattamishra, S., Goyal, N.: Are NLP models really able to solve simple math word problems? In: Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., Zhou, Y. (eds.) *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pp. 2080–2094. Association for Computational Linguistics, Online (Jun 2021). <https://doi.org/10.18653/v1/2021.naacl-main.168>, <https://aclanthology.org/2021.naacl-main.168>
12. Rae, J.W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Casirer, A., Powell, R., van den Driessche, G., Hendricks, L.A., Rauh, M., Huang, P.S., Glaese, A., Welbl, J., Dhathathri, S., Huang, S., Uesato, J., Mellor, J.F.J., Higgins, I., Creswell, A., McAleese, N., Wu, A., Elsen, E., Jayakumar, S.M., Buchatskaya, E., Budden, D., Sutherland, E., Simonyan, K., Paganini, M., Sifre, L., Martens, L., Li, X.L., Kuncoro, A., Nematzadeh, A., Gribovskaya, E., Donato, D., Lazaridou, A., Mensch, A., Lepiau, J.B., Tsimpoukelli, M., Grigorev, N.K., Fritz, D., Sottiaux, T., Pajarskas, M., Pohlen, T., Gong, Z., Toyama, D., de Masson d’Autume, C., Li, Y., Terzi, T., Mikulik, V., Babuschkin, I., Clark, A., de Las Casas, D., Guy, A., Jones, C., Bradbury, J., Johnson, M.G., Hechtman, B.A., Weidinger, L., Gabriel, I., Isaac, W.S., Lockhart, E., Osindero, S., Rimell, L., Dyer, C., Vinyals, O., Ayoub, K.W., Stanway, J., Bennett, L.L., Hassabis, D., Kavukcuoglu, K., Irving, G.: Scaling language models: Methods, analysis & insights from training gopher. ArXiv abs/2112.11446 (2021), <https://api.semanticscholar.org/CorpusID:245353475>
13. Roy, S., Roth, D.: Solving general arithmetic word problems. In: Marquez, L., Callison-Burch, C., Su, J. (eds.) *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pp. 1743–1752. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015). <https://doi.org/10.18653/v1/D15-1202>, <https://aclanthology.org/D15-1202>
14. Roy, S., Roth, D.: Mapping to Declarative Knowledge for Word Problem Solving. *Transactions of the Association for Computational Linguistics* 6, 159–172 (03 2018)
15. Saparov, A., He, H.: Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In: *The Eleventh International Conference on Learning Representations (2023)*, <https://openreview.net/forum?id=qFVVBzXxR2V>
16. Shaikh, O., Zhang, H., Held, W.B., Bernstein, M., Yang, D.: On second thought, let’s not think step by step! bias and toxicity in zero-shot reasoning. ArXiv abs/2212.08061 (2022), <https://api.semanticscholar.org/CorpusID:254686088>

17. Suzgun, M., Scales, N., Scharli, N., Gehrmann, S., Tay, Y., Chung, H.W., Chowdhery, A., Le, Q.V., hsin Chi, E.H., Zhou, D., Wei, J.: Challenging big-bench tasks and whether chain-of-thought can solve them. In: Annual Meeting of the Association for Computational Linguistics (2022), <https://api.semanticscholar.org/CorpusID:252917648>
18. Wang, B., Min, S., Deng, X., Shen, J., Wu, Y., Zettlemoyer, L., Sun, H.: Towards understanding chain-of-thought prompting: An empirical study of what matters. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 2717–2739. Association for Computational Linguistics
19. Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R.K.W., Lim, E.P.: Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In: Annual Meeting of the Association for Computational Linguistics (2023), <https://api.semanticscholar.org/CorpusID:258558102>
20. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
21. Wei, J., Wang, X., Schuurmans, D., Bosma, M., hsin Chi, E.H., Xia, F., Le, Q., Zhou, D.: Chain of thought prompting elicits reasoning in large language models. ArXiv abs/2201.11903 (2022), <https://api.semanticscholar.org/CorpusID:246411621>
22. Zhang, Z., Zhang, A., Li, M., Smola, A.: Automatic chain of thought prompting in large language models. arXiv preprint arXiv:2210.03493 (2022)
23. Zhang, Z., Zhang, A., Li, M., Smola, A.: Automatic chain of thought prompting in large language models. In: The Eleventh International Conference on Learning Representations (2023), <https://openreview.net/forum?id=5NTt8GFjUHkr>