Teaching-Inspired Integrated Prompting Framework: A Novel Approach for Enhancing Reasoning in Large Language Models

Wenting Tan[1,2,3]*, Dongxiao Chen[2], Jieting Xue[2], Zihao Wang[2], Taijie Chen[3]
[1]Key Laboratory of AI Safety, Institute of Computing Technology, CAS,
[2]NetEase Youdao,
[3]The University of Hong Kong
tanwenting2023@ict.ac.cn
{chendx, xuejt, wangzh}@rd.netease.com
ctj21@connect.hku.hk

Abstract

Large Language Models (LLMs) exhibit impressive performance across various domains but still struggle with arithmetic reasoning tasks. Recent work shows the effectiveness of prompt design methods in enhancing reasoning capabilities. However, these approaches overlook crucial requirements for prior knowledge of specific concepts, theorems, and tricks to tackle most arithmetic reasoning problems successfully. To address this issue, we propose a novel and effective Teaching-Inspired Integrated Prompting Framework, which emulates the instructional process of a teacher guiding students. This method equips LLMs with essential concepts, relevant theorems, and similar problems with analogous solution approaches, facilitating the enhancement of reasoning abilities. Additionally, we introduce two new Chinese datasets, MathMC and MathToF, both with detailed explanations and answers. Experiments are conducted on nine benchmarks which demonstrates that our approach improves the reasoning accuracy of LLMs. With GPT-4 and our framework, we achieve new state-of-the-art performance on four math benchmarks (AddSub, SVAMP, Math23K and AQuA) with accuracies of 98.2% (+3.3%), 93.9% (+0.2%), 94.3% (+7.2%) and 81.1% (+1.2%).

1 Introduction

Large Language Models (LLMs) have made significant strides in the field of Natural Language Processing (NLP), demonstrating outstanding performance across various tasks (Devlin et al., 2018; Brown et al., 2020; Chowdhery et al., 2022). Nonetheless, handling reasoning tasks effectively remains a challenge for LLMs. Evidence suggests that simply scaling up the model size does not provide an adequate solution to this issue (Rae et al., 2021; Srivastava et al., 2022).

To address this issue, a series of new prompting methods are proposed to enhance reasoning in LLMs. Chain-of-Thought (CoT) prompting (Wei et al., 2022), which mimics the human approach to solving multi-step problems by providing LLMs with few-shot exemplars including intermediate reasoning steps. Based on CoT, subsequent studies have further refined this method and improved the performance, such as Zero-shot-CoT (Kojima et al., 2022), Complexity-based CoT (Fu et al., 2022) and Least-to-Most Prompting (Zhou et al., 2022). Self-consistency (SC) is also a breakthrough method that replaces the greedy decoding strategy used in CoT but samples various reasoning paths and selects the answer with the highest consistency (Wang et al., 2022). From another perspective, MathPrompter (Imani et al., 2023) and Program of Thoughts (PoT) prompting (Chen et al., 2022) empower LLMs to generate programming language statements, enabling them to provide more accurate solutions for complex mathematical calculations.

While the prompting methods mentioned above greatly improve the reasoning performance of LLMs, they miss the crucial need for a strong grasp of concepts, theorems, and strategies. Firstly, the knowledge repository of LLMs may be incomplete, lacking enough conceptual and theoretical foundation to tackle certain arithmetic reasoning problems. Secondly, unfamiliarity with specific problem-solving strategies may cause LLMs to assume incorrect preconditions, leading to inaccurate final answers even if the intermediate calculations are correct. These challenges also arise in the process of human practice and problem-solving. Like teachers who provide foundational concepts and examples for students before practice, LLMs require educational-sourced information to ensure accurate reasoning and solutions. Therefore, drawing inspiration from traditional teaching methods, we propose a Teaching-Inspired Integrated Prompting Framework. This framework imitates the guid-

*Work done during internship at NetEase Youdao.
[1]Our code and data are available at https://github.com/SallyTan13/Teaching-Inspired-Prompting.

Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pages 827–839
January 19–24, 2025.

ance provided by teachers to students by delivering concepts or theorems from curated educational databases as background knowledge and presenting reference problems with similar and easy-to-learn solution approaches. Additionally, it incorporates double-check verification and English-Chinese ensemble mechanisms to enhance the overall reasoning ability of LLMs.

Existing arithmetic benchmarks contain a limited number of Multiple-Choice and True-or-False questions. Hence, we create two Chinese mathematical datasets called MathMC and MathToF comprising 1,000 Multiple-Choice and 1,000 True-or-False math problems respectively with answers and detailed rationales.

Our approach is evaluated on nine benchmarks, including six English datasets, one Chinese dataset, and two datasets we created. These experiments are conducted on GPT-3.5-Turbo (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023), respectively. Experimental results demonstrate that the reasoning performances of both language models on nine benchmarks are improved.

Our main contributions are as follows:

- A novel teaching-inspired integrated prompting framework is proposed to improve the reasoning capabilities of LLMs.
- Two Chinese arithmetic datasets (MathMC and MathToF) with answers and detailed rationales are constructed for further facilitating the study of arithmetic reasoning tasks.
- Comprehensive experiments show the effectiveness of our proposed integrated framework, and it achieves new state-of-the-art performance on four benchmarks.

2 Related Work

2.1 In-context Learning

In-context learning (ICL) has emerged as a successful and widely adopted approach to NLP tasks. It enables language models to learn and make predictions based on a few examples (Dong et al., 2022). Unlike supervised learning, ICL does not rely on vast amounts of data and resources for training and fine-tuning language models which makes LLMs easier to apply to various tasks as a service (Sun et al., 2022). By designing the template or format of demonstration and selecting more relevant exemplars (Wei et al., 2022; Fu et al., 2022; Chen et al., 2022; Xiong et al., 2023), the effectiveness of utilizing LLMs to address complex reasoning tasks, such as arithmetic reasoning and commonsense reasoning, has significantly improved.

2.2 Reasoning with Prompting

Chain-of-Thought Based Prompting. As for CoT prompting (Wei et al., 2022), the language model is given a few exemplars with intermediate reasoning steps so that it can offer intermediate steps when solving multi-step problems. Building upon this, Kojima et al. (2022) introduced Zero-shot-CoT, a method that simplifies the human annotation process by replacing few-shot examples with the prompt "Let's think step by step". Subsequently, Complexity-based CoT was introduced (Fu et al., 2022), targeting example selection for multi-step reasoning and demonstrating that inputs with more reasoning chains yield superior performance. This concept is expanded to output selection, favoring results with more reasoning steps. Active Prompting (Diao et al., 2023) leverages uncertainty metrics, aiding the selection of the most informative and important questions for annotation. Furthermore, Self-Consistency (Wang et al., 2022) focuses on refining the original greedy decoding strategy in CoT by sampling different reasoning paths and selecting the most frequently occurring answer.

Program Based Prompting. Unlike CoT prompting, Chen et al. (2022) proposed Program-of-Chain. This approach generates Python programs using LLMs and employs a Python interpreter to compute results, addressing LLMs' limitations in complex calculations and error tendencies. Building on this, Imani et al. (2023) introduce MathPrompter, which also leverages LLMs to generate Python programs and algebraic expressions.

External Knowledge Enhanced Reasoning. Despite the impressive knowledge base and generative capabilities of Large Language Models (LLMs), they still often generate hallucinations or erroneous information. Recent studies demonstrate that augmenting prompts with external knowledge for in-context learning, can enhance their reasoning abilities (Rubin et al., 2022). Lu et al. (2022) proposed a dynamic prompt learning method by policy gradients learning to select in-context examples from training data. Similarly, a post-processing method, Rethinking with Retrieval (He et al., 2022), retrieves external knowledge corresponding to a set of reasoning steps of CoT, giving more faithful explanations. To enhance the retrieval of relevant ex-

Problem: In a set of numbers, the largest number is 15, and the average of the set cannot be (). **Options:** A. 7 B. 13 C. 27
Answer: C
Analyses: According to the method of calculating the average, we calculate the sum of this group of numbers and then divide it by the count of numbers in this group. If the numbers in this group are different, the average will be smaller than the largest number and larger than the smallest number. If the numbers in this group are the same, the largest number, the smallest number, and the average will be equal. Since option C is greater than the maximum number 15, it is impossible. Therefore, option C is not possible, so you should choose option C.

Problem: A number divided by 15, the quotient is 30, the remainder is 8, and this number is 150.
Answer: False
Analyses: According to: dividend = quotient × divisor + remainder. In this case, the divisor is 15, the quotient is 30, and the remainder is 8, so the dividend is equal to 30×15+8=458.

Table 1: Sample Questions from the MathMC (top) and MathToF (bottom) datasets.

ternal information, planning and self-enhancement techniques are employed (Lee et al., 2024; Asai et al., 2023).

3 MathMC and MathToF

We create two datasets, MathMC and MathToF, featuring 1,000 Chinese mathematical Multiple-Choice and 1,000 Chinese True-or-False questions, accompanied by detailed explanations addressing the lack of diverse question types in existing Chinese arithmetic datasets. Sample questions and answers translated to English are shown in Table 1.

In constructing these datasets, we began by collecting a set of 4,000 elementary school-level seed Multiple-Choice questions and another set of 4,000 seed True-or-False questions, spanning grades 1 to 6 in China, with a focus on math problems from grades 4 to 6. These questions are then carefully filtered and proofread to ensure a broad coverage of knowledge points in each question. Through this rigorous filtering and selection process, we create a final dataset of 1,000 Multiple-Choice questions and 1,000 True-or-False questions, each meticulously labeled with answers and explanations. Two datasets feature a wide range of question types, including arithmetic, algebra, geometry, statistics, reasoning, and others. Specific question type statistics are shown in Table 2.

	MathMC	MathToF
Arithmetic	619	675
Algebra	113	61
Geometry	227	197
Statistics	27	37
Reasoning	7	13
Others	7	17
Total	1,000	1,000

Table 2: Question type statistics for the two datasets.

4 Teaching-Inspired Integrated Prompting Framework

We construct a three-stage integrated prompting framework as shown in Figure 1.

4.1 Stage 1: Teaching-Inspired Prompts Generation

Prompts are generated by drawing inspiration from traditional pedagogical methods, emphasizing the use of educational sources. Students begin with foundational theories and concepts from textbooks and workbooks to deeply understand problem principles, then apply these through extensive exercises and examples. To improve the capability of LLMs to solve mathematical reasoning problems, we adapt the aforementioned teaching strategy to reasoning tasks, feeding the LLMs with similar problems and the essential background knowledge (e.g. theorems, concepts, and term definitions) required to solve the specific problem.

Figure 2(a) illustrates the process of obtaining similar problems. We tokenize the test problem, preserving special math expressions, and retrieve a set of candidate problems, $\mathbb{P}$ by using BM25 (Robertson et al., 2009). The same problems and those differing only in numerical values are excluded. Candidates are ranked by their Longest Common Subsequence (LCS) length with the test problem.

Figure 2(b) describes background knowledge acquisition. We tokenize the test problem and

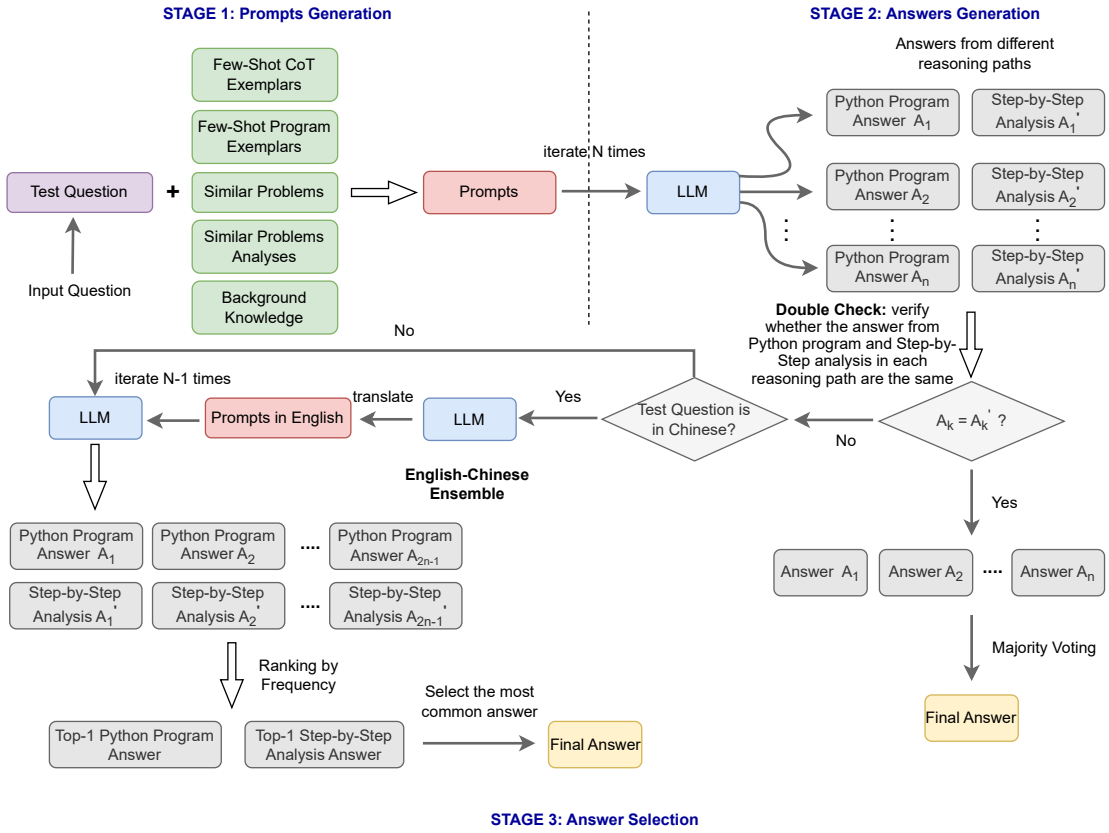

Figure 1: Architecture of our Teaching-Inspired Integrated Prompting Framework.

analyses, constructing a token set $\mathbb{T}$ by removing stopwords and operands. An LLM aids in extracting key knowledge points and uncertain theorems. These tokens serve as queries to retrieve relevant theorems and conceptual knowledge from a knowledge database, yielding background knowledge candidates, $\mathbb{K}$. Candidates are ranked by LCS length, with the top three selected. Similar to obtaining similar problems, the LCS length between each candidate k_i and the combined text is computed. Top-3 candidates are selected based on LCS length.

Therefore, prompts are mainly composed of three elements: few-shot CoT + PoT exemplars (2 cases) (one case = question + CoT exemplars + Python program exemplars + answer), similar questions and their analyses, and background knowledge. Sample teaching-inspired prompts are shown in the Appendix C.2. These prompts help LLMs generate intermediate steps for the final answer and craft the Python program required to solve the problem.

4.2 Stage 2: Answers Generation

We utilize the self-consistency method, allowing LLMs to iterate N times and generate N different paths (problem-solving strategies) for the answers.

4.3 Stage 3: Answers Selection

Double-Check Verification. We initially compare the results generated by each pathway in the N possible solution paths, i.e., verifying if the outputs of the Python programs align with the corresponding step-by-step answers. This comparison process double-checks the computation results, thereby enhancing the trustworthiness of the final answer. If all paths yield consistent answers, the most frequent answer from the consistent answers is chosen as the output via majority voting. Otherwise, the LLM is tasked to provide N-1 additional answers to the problem. After that, the process transitions to the Further Selection stage. The inclusion of Python programs and the implementation of a double-check verification strategy reduce the probability of simple calculation errors and enhance the reliability of the language model.

English-Chinese Ensemble. In the additional N-1 solution requests, if the problem is in Chinese, we instruct the language model to translate the test problem, the background knowledge, and similar problems into English before generating solutions. This approach is adopted since LLM might not fully understand certain Chinese expressions, and translation can aid in generating accurate results.

Further Selection. We evaluate the frequency of the most common answer in both the Python pro-

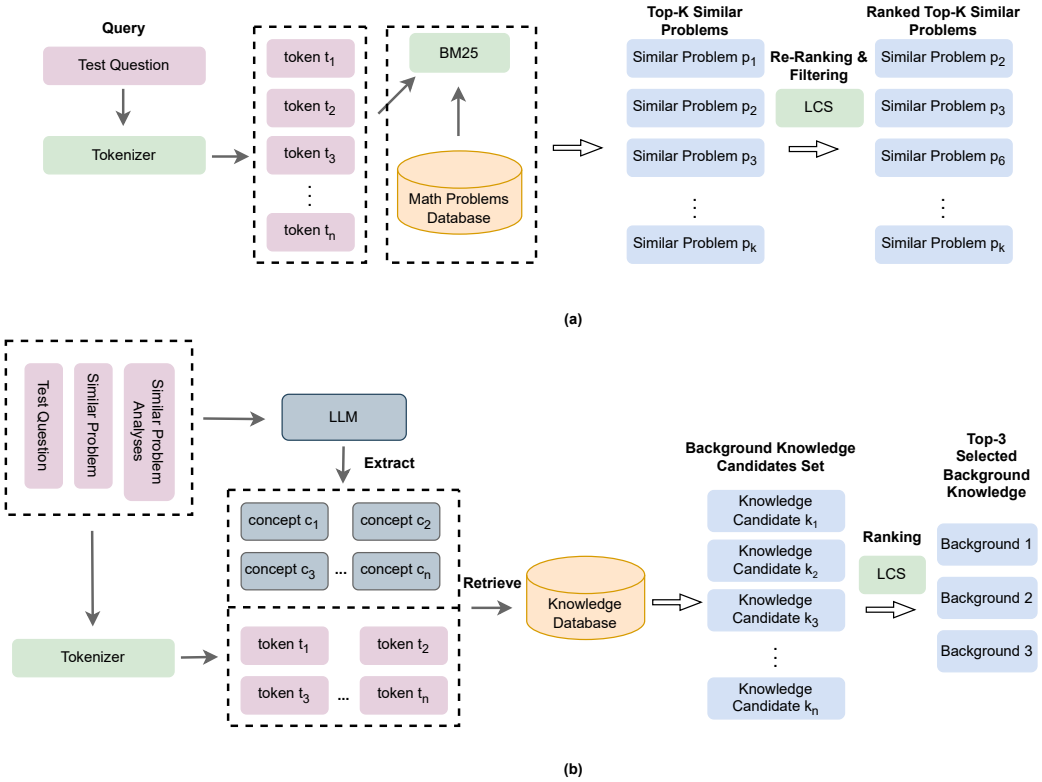

Figure 2: The procedure of similar problems retrieval (a) and the background knowledge generation (b).

gram outputs (code-ans) and the results derived from step-by-step solutions (step-by-step-ans). If the most frequent answer from the Python program (top-code-ans) has a frequency equal to or higher than that of the most frequent answer from the step-by-step solution (top-step-by-step-ans), then the top-code-ans is selected as the output. Conversely, the top-step-by-step-ans is chosen as the final output.

5 Experiments

5.1 Experimental Settings

Datasets. Our method is evaluated on six English mathematical reasoning benchmarks: AddSub (Hosseini et al., 2014), SingleEQ (Koncel-Kedziorski et al., 2015), SVAMP (Patel et al., 2021), MultiArith (Roy and Roth, 2015), GSM8K (Cobbe et al., 2021), AQuA (Ling et al., 2017), one Chinese math reasoning benchmark, Math23K (Wang et al., 2017), and two datasets (MathMC and MathToF) we construct.

Models. For our experiments, we use two LLMs from the GPT series: GPT-3.5-Turbo (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023). We perform all experiments using OpenAI's API, ensuring that our methodology aligns with standard practices and is easy to replicate.

Prompts and Hyperparameters. Specific prompts are detailed in Appendix C. Only the most relevant similar problem is included in the prompts for the experiments. For the greedy decoding strategy, the temperature is set to 0.0, while for the Self-Consistency strategy, it is adjusted to 0.5.

5.2 Main Results

We compare the evaluation results of our integrated prompting framework with the Chain-of-Thought method on GPT-3.5-Turbo and GPT-4 models. As shown in Table 3, our framework improves the mathematical reasoning performance significantly over two models on seven math benchmarks, especially improving 8.8% on GSM8K, 24.8% on Math23K, 8.0% on SingleEQ and 10.2% on AQuA when used on GPT-3.5-Turbo. Surprisingly, with GPT-4 and our integrated prompting framework, we achieve the new state-of-the-art performance on four math benchmarks (AddSub, SVAMP, Math23K and AQuA) with accuracies of 98.2% (+3.3%), 93.9% (+0.2%), 94.3% (+7.2%) and 81.1% (+1.2%).

Additionally, we present the results of GPT-3.5-Turbo and GPT-4 on two datasets we created, MathMC and MathToF. As seen in Table 3, leveraging our prompt method on GPT-3.5-Turbo yields a significant enhancement in performance, boosting reasoning accuracy on MathMC by 18.8% and MathToF by 10.5%, respectively. On deploying the same prompting framework on GPT-4, we observe a marked improvement as well, with increases of

Method	Dataset								
	AddSub	SingleEQ	MultiArith	SVAMP	GSM8K	AQuA	Math23K	MathMC	MathToF
Previous SoTA	95.7	98.8	100.0	93.7	97.0	79.9	87.1	-	-
CoT (GPT-3.5-Turbo)	89.6	90.2	95.0	82.2	75.5	60.6	63.5	60.0	68.3
Ours (GPT-3.5-Turbo)	**92.7**	**98.2**	**98.0**	**86.0**	**84.3**	**70.8**	**88.3**	**78.8**	**78.8**
	(+3.1)	(+8.0)	(+3.0)	(+3.8)	(+8.8)	(+10.2)	(+24.8)	(+18.8)	(+10.5)
CoT (GPT-4)	95.7	94.5	98.6	92.6	91.2	76.4	83.2	88.1	82.5
Ours (GPT-4)	**98.2**	**98.6**	**99.0**	**93.9**	**94.8**	**81.1**	**94.3**	**92.2**	**89.2**
	(+2.5)	(+4.1)	(+0.4)	(+1.3)	(+3.6)	(+4.7)	(+11.1)	(+4.1)	(+6.7)

Table 3: Evaluation results of teaching-inspired integrated prompting framework applied on different models, GPT-3.5-Turbo and GPT-4. Our approach improves performance on different models and benchmarks. Our approach achieves state-of-the-art performance on AddSub, SVAMP, Math23K and AQuA benchmarks on GPT-4. Previous state-of-the-art performance are from (Gao et al., 2023) for SingleEQ, (Wang et al., 2022) for MultiArith, (Zhao et al., 2023) for AddSub and SVAMP, (Zhou et al., 2023) for GSM8K, (Zheng et al., 2023) for AQuA, (Zhang et al., 2022) for Math23K dataset.

Method	Dataset								
	AddSub	SingleEQ	MultiArith	SVAMP	GSM8K	AQuA	Math23K	MathMC	MathToF
Ours	**92.7**	**98.2**	**98.0**	**86.0**	**84.3**	**70.8**	**88.3**	**78.8**	**78.8**
w/o BG + Sim_Prob	90.3	95.4	97.2	84.7	83.4	68.5	79.6	64.4	73.0
	(-2.4)	(-2.8)	(-0.8)	(-1.3)	(-0.9)	(-2.3)	(-8.7)	(-14.4)	(-5.8)
w/o Python Program	93.4	98.2	97.8	85.7	75.2	-	84.2	-	-
	(+0.7)	(+0.0)	(-0.1)	(-0.3)	(-9.1)	-	(-4.1)	-	-
w/o Ans_Selection	91.4	91.3	95.0	83.6	75.4	60.6	76.5	70.6	74.3
	(-1.3)	(-6.9)	(-3.0)	(-2.4)	(-8.9)	(-10.2)	(-11.8)	(-8.2)	(-4.5)
w/o Ch_En_Ens	-	-	-	-	-	-	88.5	75.8	77.8
	-	-	-	-	-	-	(+0.2)	(-3.0)	(-1.0)

Table 4: Ablation study on different components of our proposed integrated prompting framework, conducted using seven public datasets and two newly created datasets, with all experiments performed on the GPT-3.5-Turbo model.

4.1% and 6.7% in respective metrics. These results demonstrate the efficacy of our approach in facilitating reasoning tasks.

5.3 Ablation Study

In this section, we conduct ablation studies to investigate the impact of various components in our proposed integrated prompting framework and the influence of different numbers of similar problems.

5.3.1 Similar Problems and Background Knowledge.

Removing similar problems and background knowledge from the prompts leads to a general decline in accuracy across nine datasets shown in Table 4. This indicates that similar problems and backgrounds play a guiding role in enhancing LLM reasoning.

5.3.2 Python Program Generation and Double-Check Verification Strategy.

The results in Table 4 demonstrate that removing the Python program generation and double-check strategies has minimal impact or even a slight improvement in accuracy for simpler math problem sets (AddSub, SingleEQ and MultiArith). However, for more complex problem sets (GSM8K, Math23K), there is a noticeable decrease in accuracy by 9.1% and 4.1% respectively. This indicates that incorporating this kind of strategy helps compensate for LLMs susceptibility to computational errors in complex calculations.

5.3.3 Answer Selection Strategy.

Analyzing the experimental results, it can be found that the accuracy decreases across nine datasets, especially on more complex datasets including GSM8K, AQuA, and Math23K, where the accuracy drops by 8.9%, 10.2%, and 11.8% respectively.

When combined with self-consistency and double-check-verification methods, simple calculation errors or occasional faulty reasoning can be avoided. Different problem-solving paths and calculation methods (Python programs or natural language) produce the same results for a given problem.

5.3.4 English-Chinese Ensemble Strategy.

We evaluate the impact of the English-Chinese Ensemble strategy on three Chinese datasets. When this component is removed, the accuracy on MathMC and MathToF drops by 3.0% and 1.0%. This finding suggests that translating Chinese problems into English can make it easier for the language model to understand, thereby generating more accurate solutions.

5.3.5 The Number of Similar Problems.

We explore the effect of the number of similar problems by adding different numbers of varying or the same similar problems[2] to prompts. Figure 3(a) shows that the effectiveness of adding similar questions is not solely determined by quantity. When the added questions differ significantly from the target question, they can negatively impact accuracy. However, within a certain similarity threshold, increasing the number of similar questions improves LLMs' reasoning accuracy. Figure 3(b) demonstrates that including multiple identical top-similar questions in prompts leads to a notable improvement. This approach indirectly addresses the challenge of acquiring external information, helping LLMs capture and utilize relevant external knowledge more effectively.

6 Conclusion

This paper presents an innovative teaching-inspired integrated prompting framework, to conquer the limitations of LLMs in arithmetic reasoning tasks. The framework emulates the teaching process to introduce essential concepts, theorems, and similar problems to LLMs. It also incorporates double-check and answer selection mechanisms, which significantly enhance their ability to perform arithmetic reasoning tasks. Empirical results reveal that employing our framework leads to substantial improvements in arithmetic reasoning accuracy. Our study also underscores the need for more diverse and comprehensive benchmarks for evaluating the performance of arithmetic reasoning, which we address by introducing the MathMC and MathToF datasets. In future work, researchers can further refine and explore its applicability to other domains.

[2]Adding K similar problems means repeating the exact same similar problem K times within the prompt.

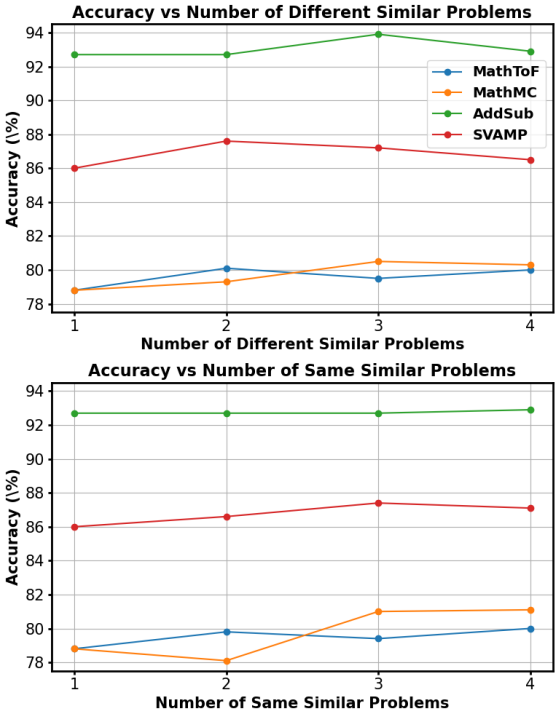

Figure 3: Results of adding different numbers of varying or the same similar problems into prompts.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

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 arXiv:2204.02311*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias

Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Hangfeng He, Hongming Zhang, and Dan Roth. 2022. Rethinking with retrieval: Faithful large language model inference. *arXiv preprint arXiv:2301.00303*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. 2024. Planrag: A plan-then-retrieval augmented generation for generative large language models as decision makers. *arXiv preprint arXiv:2406.12430*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *arXiv preprint arXiv:2209.14610*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774, 2023.*

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

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 arXiv:2112.11446*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 845–854.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Jing Xiong, Zixuan Li, Chuanyang Zheng, Zhijiang Guo, Yichun Yin, Enze Xie, Zhicheng Yang, Qingxing Cao, Haiming Wang, Xiongwei Han, et al. 2023. Dq-lore: Dual queries with low rank approximation re-ranking for in-context learning. *arXiv preprint arXiv:2310.02954*.

Wenqi Zhang, Yongliang Shen, Yanna Ma, Xiaoxia Cheng, Zeqi Tan, Qingpeng Nong, and Weiming Lu. 2022. Multi-view reasoning: Consistent contrastive learning for math word problem. *arXiv preprint arXiv:2210.11694*.

Xu Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Qizhe Xie. 2023. Automatic model selection with large language models for reasoning. *arXiv preprint arXiv:2305.14333*.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, et al. 2023. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *arXiv preprint arXiv:2308.07921*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Similar Problems and Background Knowledge Database

Figure 2 illustrates the process of similar problems and background knowledge retrieval.

For the background knowledge database, we curated a rich repository from mathematical textbooks, exercise workbooks, and web sources. From these resources, we extract a wealth of knowledge points and background information, encompassing theories, theorems, and problem-solving methodologies. Subsequently, we compile and store these findings to establish a comprehensive knowledge base.

In parallel, the similar problems database comprises problems sourced from mathematical textbooks and workbooks, each accompanied by detailed analyses derived from these materials.

B Ablation Study of Double-Check Verification

To explore the effectiveness of our double-check verification, we conduct experiments on three datasets (AddSub, SVAMP and GSM8K) with different levels of difficulty. The experiments compare results with and without the double-check verification strategy. With the double-check verification strategy, the framework generates N solution paths (including step-by-step answers and Python program outputs). If the two types of results are consistent, the framework directly outputs the final answer. If not, it generates N-1 additional answers and applies the majority voting method to determine the final output. without the double-check verification strategy, the framework generates a fixed 2N-1 candidate answers (step-by-step answers and program outputs) and directly uses majority voting to produce the final answer.

We report the average number of generation outputs and average accuracy of each experiment. From the table 5, it can be observed that when generating 3 outputs, our double-check verification method achieves higher accuracy. When generating 5 outputs, the accuracy is comparable or slightly higher, while generating 9 outputs, our strategy is slightly lower than the majority voting approach. However, the double-check verification strategy requires fewer outputs compared to generating 5 or 9 outputs, significantly saving both time and cost.

C Prompts

This section presents the System Prompt, Teaching-Inspired Prompt, and Chain-of-Thought Prompt used in our experiments.

C.1 System Prompt

You are a super smart elementary school math teacher. You need to read the math problem carefully and solve it in a step-by-step way to be sure you have the right answer. You often make mistakes in calculations, so please be careful when calculating.

Please do not be influenced by the typos in the question and reason based on the semantics of the question.

Dataset	Method	# Outputs	Accuracy (%)
AddSub	Double-Check Verification	3.07	92.7
	Fixed Outputs	3.00	92.4
	Fixed Outputs	5.00	92.8
	Fixed Outputs	9.00	93.6
SVAMP	Double-Check Verification	3.27	86.0
	Fixed Outputs	3.00	85.4
	Fixed Outputs	5.00	85.9
	Fixed Outputs	9.00	88.1
GSM8K	Double-Check Verification	3.85	84.3
	Fixed Outputs	3.00	79.4
	Fixed Outputs	5.00	83.2
	Fixed Outputs	9.00	84.2

Table 5: Performance comparison between the double-check verification strategy and the fixed-output majority voting strategy (i.e., without double-check verification), conducted on the GPT-3.5-Turbo model.

Please make sure your replies as simple and easy to understand as possible.

C.2 Teaching-Inspired Prompts

This section shows the Teaching-Inspired Prompts format along with an example of the prompt.

C.2.1 Teaching-Inspired Prompts Format

If there is a reference question and the reference question is very similar to the question you need to answer, you should think based on the analysis process of the reference question, but you cannot be affected by its question stem. You still need to return the complete analysis process of the question you need to answer.

Reference question: `sim_question`
Reference analysis: `sim_analysis`
Reference answer: `sim_answer`

You may use the following background knowledge when analyzing the problem:
Background: `background`
Question: `question to be solved`

C.2.2 Example

If there is a reference question and the reference question is very similar to the question you need to answer, you should think based on the analysis process of the reference question, but you cannot be affected by its question stem. You still need to return the complete analysis process of the question you need to answer.

Reference question: In Class 6, there are a total of 52 students. Among them, 30 students like to eat rice, and 29 students prefer noodles. The number of students who like both rice and noodles is ( ).
Reference analysis: Based on the information "There are a total of 30 students who like to eat rice and 29 students who prefer noodles," we can calculate the total number of students who like either rice or noodles: 30 + 29 = 59. However, this count includes the students who like both rice and noodles twice. Therefore, applying the principle of inclusion-exclusion, we can determine that the number of students who like both rice and noodles is 59 - 52 = 7. Thus, the answer is 7.
Reference answer: 7

You may use the following background knowledge when analyzing the problem:
Background: principle of inclusion-exclusion: $|A \cup B| = |A| + |B| - |A \cap B|$
Question: In order to prepare for the fruit party, Class 3 made statistics on the two kinds of fruits that everyone liked. 38 students like to eat bananas, 32 students like to eat fragrant pears, and 10 students like both. How many students are in Class 3?

C.3 Chain-of-Thought Prompt

C.3.1 Math Word Problems

a) Few-Shot Examples
Examples:
Question: Xiaoming is 5 years old this year, and his father is 25 years old this year. How old will

Xiaoming be when his father is 30 years old?
thought:
When the father is 30 years old, 5 years have passed since he was 25.
At this time, Little Ming should be 10 years old (5 + 5).
steps:
1. We need to figure out how many years it will take for the father to reach 30 years old from now (25 years old). This can be obtained by subtracting 25 from 30, that is, 30-25=5. Therefore, the father still needs 5 years to reach 30 years old.
2. We know that Little Ming is now 5 years old, so his age will increase in the next 5 years. Since his age increases by 1 year every year, in 5 years his age will increase by 5 years.
3. If we add Little Ming's current age of 5 to the increase of 5 years in the next 5 years, we can get Little Ming's age when his father is 30 years old. That is 5+5=10.
answer: 10

Question: Xiaoming read 30 pages on the second day, and read one more page than the second day on the first day. How many pages did he read on the first day?
thought:
Since Xiaoming read 30 pages on the second day and read one more page than the second day on the first day, Xiaoming read 31 pages on the first day.
steps:
1. Xiaoming read one more page on the first day than on the second day.
2. Xiaoming read 30 pages on the second day.
3. Therefore, the number of pages Xiaoming read on the first day is one more than that of the second day.
4. Thus, Xiaoming read 30 pages + 1 page on the first day, which is equal to 31 pages.
answer: 31 pages

b) Reply Format
When you are certain that the answer is correct, you need to return the following content:
thought: [Return your thinking process for solving this problem.]
steps: [Return the detailed solution steps.]
answer: [The answer to the question. If there are multiple questions in the problem, the answer format should be: (1) Answer to the first question. (2) Answer to the second question....]
Important: Your return format must be consistent with the Examples
Important: The content you return must include fore keywords: thought, steps, and answer, and the content of every keyword cannot be empty. Besides, every keyword should be in English.

C.3.2 Multiple-Choice Problems

a) Few-Shot Examples
Examples:
stem: The approximate distance from Xiao Ning's home to school, given that he walks an average step length of 58 centimeters and has taken 135 steps, is about ()
options: A.8000m B.80m C.70m
thought: Based on the formula distance = number of steps × length per step, write the equation 58 × 135, calculate it using the integer multiplication method, and get the result of 7830. Then, according to the rounding rule, the answer can be solved.
steps:
1. Using the formula distance = number of steps × length per step, derive the equation 58 × 135.
2. According to the equation 58 × 135 = 7830 cm, determine the distance from Xiao Ning's house to the school as 7830 cm.
3. Since the options are in meters and the result we calculated earlier is in centimeters, we should convert centimeters to meters. 7830 cm = 78.3 m
4. Applying rounding rules, 78.3 m is approximately equal to 80 m, so option B should be selected.
answer: B

stem: Which of the following statements is correct?
options: A. A ray is 50 meters long B. There are 6 big months (31 days) and 6 small months (30 days) in a year C. 1/3:1/4 and 4:3 can form a proportion D. The whole year in 2020 has 365 days.
thought: Determine whether four choices in the question are correct or not
steps:
1. Option A, since a ray has only one endpoint and extends infinitely in one direction, it cannot be measured in terms of length. Therefore, Option A is incorrect.
2. Option B, there are 7 big months and 5 small months in a year, so the statement in Option B is incorrect.
3. Option C, to form a proportion, the ratios on

both sides should be equal. 1/3:1/4 = 4:3 = 4/3, and 4:3 is equal to 4/3. Therefore, it can form a proportion with 4:3. The statement in Option C is correct.
4. Option D, 2020 is a leap year because it is divisible by 4, so the whole year has 366 days. The statement in Option D is incorrect.
5. Therefore, the correct answer is Option C.
answer: C

stem: Which of the following expressions has a value greater than 100?
options: A.50+45 B.90+20 C.90-80
thought: Compare the result of adding each equation to 100.
steps:
1. The result of option A is 50 + 45 = 95, which is less than 100, so Option A is incorrect.
2. The result of option B is 90 + 20 = 110, which is greater than 100, so Option B is correct. The correct answer is B.
3. To prevent calculation errors, let us calculate the answer for Option C again. 90 - 80 = 10, which is less than 100, so Option C is also incorrect.
4. Therefore, the final answer is B.
answer: B

b) Reply Format
When you are certain that the answer is correct, you need to return the following content:
thought: <It's necessary. Return your thinking process for solving this problem.>
steps: <It's necessary. The steps for solving the question, with as much detail as possible.>
answer: <It's necessary. The specific option to the question, such as A/B/C/D.>
Important: Your return format must be consistent with the Examples
Important: The content you return must include the keyword: thought, steps and answer and the content of every keyword cannot be empty. Besides, each keyword should be in English.

C.3.3 True-or-False Problems

a) Few-Shot Examples
Examples:
Question: True or False: The number that is 100 more than the largest three-digit number is 1999.
thought: Firstly, we need to know what the largest three-digit number is, and then calculate the largest three-digit number plus 100 to determine whether the result is equal to 1999. If the result is not equal to 1999, then the statement is false. If it is equal to 1999, then the statement is true.
steps:
1. The largest three-digit number is 999.
2. Adding 100 to 999 results in 1099.
3. The result of the calculation is 1099, which is not equal to 1999. Therefore, the answer to this question is false.
answer: False

Question: True or False: The "9" in 0.019 is in the hundredth place.
thought: The first decimal place to the right of the decimal point is the tenth place, the second decimal place is the hundredth place, and the third decimal place is the thousandth place.
steps:
1. To determine the hundredth place, we need to look at the second decimal place to the right of the decimal point.
2. Looking at the third decimal place to the right of the decimal point in 0.019, we find that it is 9.
3. We can conclude that the "9" in 0.019 is in the thousandth place.
4. Therefore, the statement in the question is false.
answer: False

Question: True or False: The remainder is never greater than the quotient.
thought: This statement can be judged by the relationship between the remainder, divisor, and quotient, or by giving examples to see if the statement is true or false.
steps:
1. Generally, the remainder cannot be greater than the divisor, but there is no absolute relationship between the remainder and the quotient.
2. For example, 104 divided by 33 equals 3 with a remainder of 5, where the remainder of 5 is greater than the quotient 3.
3. Another example is 3 divided by 4, which equals 0 with a remainder of 3, where the remainder 3 is greater than the quotient 0.
4. Therefore, based on the counterexamples and concept relationships, we can conclude that this statement is false.
5. Therefore, the final answer is false.
answer: False

b) Replay Format
When you are certain that the answer is correct,

you need to return the following content:
thought: <It's necessary. Return your thinking process for solving this problem.>
steps: <It's necessary. The steps for solving the question, with as much detail as possible.>
answer: <It's necessary. If you believe that the statement in the question is correct, return True. If you believe that the statement in the question is false, return False.>
Important: Your return format must be consistent with the Examples
Important: The content you return must include the keywords: thought, steps and answer. and the content of every keyword cannot be empty. Besides, each keyword should be in English.