SMART: Evaluating LLMs' Mathematical Reasoning via a Human Cognitive Process-Inspired Benchmark

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have achieved remarkable performance across a wide range of mathematical benchmarks. However, concerns remain as to whether these successes reflect genuine reasoning or superficial pattern recognition. Existing evaluation methods, which typically focus either on the final answer or on the intermediate reasoning steps, reduce mathematical reasoning to a shallow input–output mapping, overlooking its inherently multi-stage and multi-dimensional cognitive nature. Inspired by Pólya's problem-solving theory, we propose SMART, a benchmark that decomposes mathematical problem-solving into four cognitive dimensions: **S**emantic Understanding, **M**athematical Reasoning, **A**rithmetic Computation, and **R**eflection & Refinemen**t**, and introduces dimension-specific tasks to measure the corresponding cognitive processes of LLMs. We apply SMART to 22 state-of-the-art open- and closed-source LLMs and uncover substantial discrepancies in their capabilities across dimensions. Our findings reveal genuine weaknesses in current models and motivate a new metric, the All-Pass Score, designed to better capture true problem-solving capability.

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; Wei et al., 2022) have demonstrated impressive performance and are being increasingly integrated into real-world applications *e.g.*, education (Wang et al., 2024), scientific computing (Ma et al., 2025), and decision support (OpenAI, 2024; Guo et al., 2025). With this widespread adoption, assessing their capability boundaries has become essential. The mathematical reasoning, a key indicator of higher-order cognition, serves as a critical benchmark to evaluate the logical thinking and systematic problem-solving of models.

However, existing LLM mathematical benchmarks are misaligned with the human multi-dimensional cognitive process of mathematical problem-solving. Pólya's problem-solving theory (Polya, 2014) formalizes this cognitive process into four progressive dimensions: understanding the problem, devising a plan, executing the plan, and looking back on the solution. Unfortunately, mainstream evaluation approaches, such as GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), reduce this process to simple end-to-end matching, assessing LLMs solely based on final answer correctness (Fig.1a). While recent benchmarks, MR-

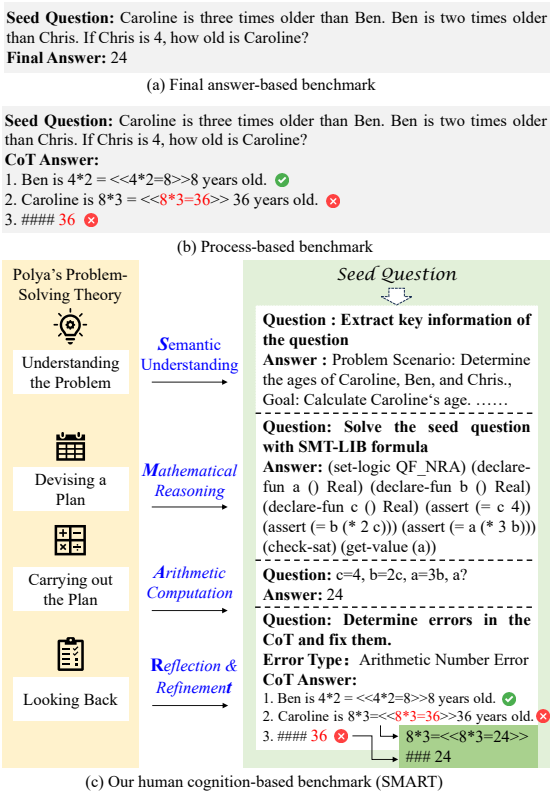

Figure 1: Comparison of evaluation paradigms for LLM mathematical reasoning. Final-answer-based benchmarks evaluate only the final outcome, process-based benchmarks detect errors in reasoning steps, while SMART builds on Pólya's problem-solving theory to evaluate four cognitive dimensions.

GSM8K (Zeng et al., 2025) and ProcessBench (Zheng et al., 2025), have begun incorporating step-by-step solution verification, they still fall short of comprehensively evaluating the distinct cognitive stages that underlie mathematical reasoning (Fig.1b). These two approaches fail to capture the subtle cognitive processes at each problem-solving phase, making it impossible to pinpoint where models struggle in the reasoning processes and, therefore limiting guidance for targeted improvements.

To address these limitations, we propose the first benchmark, called SMART, to systematically evaluate the complete cognitive process of LLMs in mathematical reasoning (Fig.1c). Guided by Pólya's problem-solving theory, SMART systematically decomposes each mathematical problem along the reasoning pipeline into four cognitive dimensions, corresponding to **S**emantic Understanding (Understanding), **M**athematical Reasoning (Reasoning), **A**rithmetic Computation (Arithmetic), and **R**eflection & Refinemen**t** (R&R). This decomposition enables an independent assessment of LLM capabilities in each cognitive dimension, allowing a fine-grained diagnosis of model performance in distinct problem-solving stages. Moreover, we introduce a new metric, the All-Pass Score, which measures model accuracy only when all four dimension-specific tasks are correctly solved.

Creating a comprehensive, multi-task benchmark at scale presents a fundamental challenge: each problem requires carefully designed sub-questions that target the specific cognitive process, demanding extensive human annotation. To make this approach both scalable and cost-effective, we further introduce an automated generation pipeline that transforms seed problems into four-dimensional assessment tasks, incorporating neuro-symbolic (Barrett et al., 2010; De Moura and Bjørner, 2008) and human verification to enable iterative quality validation. Furthermore, these dimension-specific tasks are novel for LLMs and thus contribute to mitigating data contamination.

We evaluate 22 recently released open- and closed-source LLMs on SMART. Experimental results demonstrate that even the most advanced models perform poorly under the All-Pass Score metric, underscoring the challenging nature of our benchmark. In addition, SMART serves as a diagnostic tool, identifying which cognitive dimensions emerge as the primary bottlenecks in mathematical problem-solving. Furthermore, we find that targeted improvements in specific weak dimensions can lead to substantial overall gains in mathematical capability—for example, increasing the final answer accuracy of Qwen2.5-72B by 11.77%. Our main contributions are as follows:

- To evaluate the true mathematical reasoning capability of LLMs, we propose the SMART benchmark that consists of 10,000 questions across distinct cognitive dimensions, and the new All-Pass Score, enabling comprehensive evaluation of the problem-solving process.
- Of equal importance to the SMART benchmark, we introduce a novel data curation and quality control framework that automates the construction of dimension-specific sub-tasks from seed questions and validates the benchmark via rigorous correctness assessment.
- Based on SMART, we reveal substantial disparities in LLMs' mathematical capabilities and offer dimension-specific, interpretable diagnostics that pinpoint weaknesses. Targeting the weakest dimension with a reflection-and-refinement prompt boosts Qwen2.5-72B's final-answer accuracy by 11.77%.

2 Related Work

Mathematical Benchmark. Numerous mathematical benchmarks with varying levels of difficulty have been developed to explore the upper bound of LLMs' mathematical capabilities. These benchmarks range from grade-school-level datasets (Cobbe et al., 2021), to high-school-level datasets (Hendrycks et al., 2021), and extend to expert-level datasets (Glazer et al., 2024). Their scope covers a broad range of mathematical domains, including geometry, number theory, and real analysis. However, despite their increasing difficulty, these benchmarks primarily adopt a final answer-based evaluation approach, making it unclear whether LLMs genuinely understand mathematical concepts or simply rely on pattern-matching to produce correct answers (Mirzadeh et al., 2025). To address this, ProcessBench (Zheng et al., 2025) and PRMBench (Song et al., 2025) have been innovatively proposed to enable process-based evaluation by identifying erroneous steps in the model's mathematical reasoning. Nevertheless, these process-based benchmarks still fall short of capturing human thinking, since they do not evaluate the fine-grained cognitive processes across the stages of problem-solving.

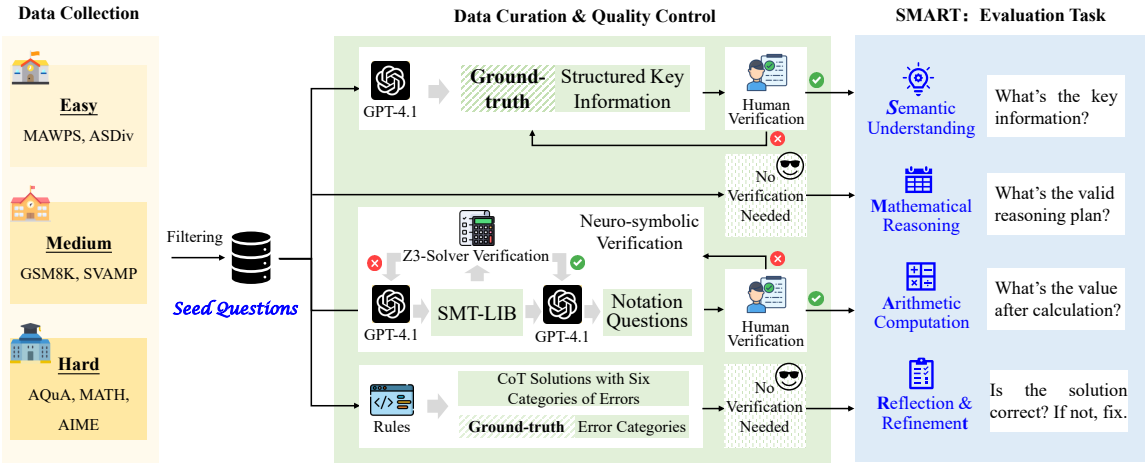

Figure 2: Overview of SMART benchmark construction. First, we collect seed questions from datasets of varying difficulty and filter out those that do not meet our requirements. Second, the seed questions are used to generate dimension-specific tasks along with their corresponding ground-truths. Finally, the generated data are validated through a neuro-symbolic and human verification when needed to ensure data quality.

Dynamic evaluation. The widespread use of benchmarks increases the risk of data contamination, potentially inflating performance evaluations (Li et al., 2024a). Recent studies address these concerns with dynamic evaluation (Zhu et al., 2023, 2024) that generate adaptive test data via predefined rules. GSM-Plus (Li et al., 2024b) and GSM-Symbolic (Mirzadeh et al., 2025) similarly generate variants from seed questions. These approaches have shown encouraging progress in mitigating data leakage and improving robustness in evaluations. However, manually annotating newly generated questions is labor-intensive and costly, motivating the need for automated, scalable data generation and verification.

Despite these recent advances in mathematical benchmarks and dynamic evaluation, persistent limitations underscore the need for a benchmark that comprehensively assesses the entire problem-solving process, provides fine-grained and interpretable analyses, and reduces the cost of constructing benchmarks. To address this gap, SMART is designed to systematically evaluate the mathematical reasoning capabilities of LLMs.

3 The SMART Benchmark

3.1 Overview

SMART is a fine-grained multi-task benchmark for evaluating the problem-solving capabilities of LLMs. Its four sub-tasks are derived from Pólya's problem-solving theory. In *How to Solve It* (Pólya and Conway, 1957), Pólya conceptualized mathematical problem-solving as a four-step cognitive process: (1) Understanding the problem, (2) Devising a plan, (3) Carrying out the plan, and (4) Looking back. Adopting Pólya's problem-solving theory can clarify where LLMs succeed or fail by separating cognitive dimensions. Therefore, building on this cognitive framework, SMART evaluates LLMs along four corresponding cognitive dimensions: Semantic Understanding (Understanding), Mathematical Reasoning (Reasoning), Arithmetic Computation (Arithmetic), and Reflection & Refinement (R&R). The evaluation settings for the four dimension tasks are summarized in Tab. 1.

3.2 Evaluation Sub-Tasks

Understanding. The Understanding task evaluates a model's semantic understanding capability by extracting and organizing key information from the question. In this task, the input question is a seed question, and the model identifies and categorizes essential components into a predefined template. The template comprises five categories: problem scenario, goal, known and unknown quantities, relationships and constraints, and irrelevant information.

We adopt this design to evaluate not only the model's capacity to summarize and highlight salient information but also its depth of comprehension. By requiring the model to distinguish the roles of different elements and their interconnections, the task provides a nuanced measure of problem understanding.

Reasoning. The Reasoning task evaluates the mathematical reasoning capability by requiring LLMs to produce a symbolic formalization of the solution. Given a seed question, the model is prompted to

Task	Question (Verification)	Answer	Evaluator	Ground-truth (Verification)
Understanding	SQ (✕)	SKI	LLM-as-a-Judge	SKI (GPT + Human)
Reasoning	SQ (✕)	SMT-LIB	Z3 Solver + Rule-match	Final answer of SQ (✕)
Arithmetic	NQ (GPT + Human)	Answer of NQ	Rule-match	Final answer of SQ (✕)
Reflection	CoT with Errors (✕)	Error Categories	Rule-match	Error Categories (✕)
Refinement	CoT with Errors (✕)	Refined CoT	Rule-match	Final answer of SQ (✕)

Table 1: Overview of question, answer, evaluator, and ground-truth of each dimension task in SMART. SQ means seed question. NQ means notation-based arithmetic question. SKI means structured key information.

solve it using symbolic formalization in the SMT-LIB format (Barrett et al., 2010).

This task compels the model to capture the underlying logical structure of the problem and the intricate relationships among its components. With few-shot prompting, LLMs easily learn to produce SMT-LIB–formatted answers.

Arithmetic. The Arithmetic task evaluates an LLM's capability to perform arithmetic computation by requiring it to solve notation-based questions containing only numerical values and variables. These notation-based questions are simplified from the seed questions, expressed purely in terms of numbers and variables, and require the execution of basic arithmetic operations.

We design this task to isolate arithmetic skills from other cognitive demands—such as language comprehension or complex reasoning—thereby providing a focused and precise assessment of a model's arithmetic capabilities.

R & R. The Reflection & Refinement task evaluates the LLM's capacity for self-critique. The model is presented with a question and its chain-of-thought (CoT) solution, and is tasked with identifying potential errors in CoT (Reflection). It then revises the errors and produces a refined CoT (Refinement). Importantly, if the model fails to detect all errors during the Reflection stage, it is not allowed to proceed to Refinement.

3.3 Benchmark Construction

As shown in Fig. 2, SMART is constructed in three stages: data collection, data curation, and quality control. Through this deliberately designed pipeline, we can automatically generate four dimension-specific tasks with corresponding ground truths, while requiring significantly less human verification compared to traditional benchmarks that rely heavily on manual annotation.

3.3.1 Data Collection

We begin by collecting a diverse set of seed questions from seven widely used mathematical problem datasets spanning three difficulty levels. Easy questions are drawn from MAWPS (Koncel-Kedziorski et al., 2016) and ASDiv (Miao et al., 2020); medium questions from GSM8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021); and hard questions from AQuA (Ling et al., 2017), MATH (Hendrycks et al., 2021), and AIME 2024 (Huggingface, 2024).

To ensure verifiability and sufficient reasoning complexity, we filter questions that can be formalized in SMT-LIB format (so their solutions can be validated using the Z3 solver) and require at least two reasoning steps, preventing the SMART benchmark from being overly trivial. After filtering, we obtain 2,000 seed questions, which serve as the foundation for constructing the dimension-specific evaluation tasks.

3.3.2 Data Curation

As shown in Tab.1, the questions used in the Understanding and Reasoning tasks, as well as the ground truths for the Reasoning, Arithmetic, and Refinement tasks, are directly derived from the original seed questions and therefore do not require additional verification. This design significantly reduces the cost of human annotation while maintaining high data quality. Below, we describe the curation process for the remaining dimensions.

Structured Key Information. The ground-truth for the Understanding task is the structured key information, which is generated by GPT-4.1.

Notation-based Questions. For the Arithmetic task, the input questions are notation-based questions. Directly converting a seed question into a notation problem is non-trivial, as it requires simplifying natural language into structured mathematical operations while preserving logical relationships among variables. To address this challenge, we adopt a two-stage process: seed questions are first formalized into SMT-LIB representations using GPT-4.1 to capture their underlying logic, and these formal expressions are subsequently translated into notation-based arithmetic questions also

Model	Understanding LLM@Un	Reasoning ACC@Re	Arithmetic ACC@Ar	R&R ACC@RR	Reflection ACC@R-t	Refinement ACC@R-m	Final Answer ACC@Fi	All-Pass Score ACC@All
Open-source models								
Phi4-14B	93.41	64.12	96.45	22.06	26.75	82.34	89.55	20.77
Gemma3-27B	94.31	40.81	39.94	24.23	32.55	74.35	54.62	11.96
GLM4-32B	94.11	55.92	36.62	22.85	28.50	88.57	54.13	11.56
Qwen2.5-72B	**95.86**	62.53	35.55	33.85	35.75	94.69	41.05	13.95
Qwen3-32B	93.32	**86.14**	**98.50**	**54.75**	56.70	96.52	**93.75**	**50.72**
Qwen3-30B-A3B	93.14	85.20	98.15	43.89	**58.25**	75.36	92.85	44.11
Llama3.1-8B	88.58	9.05	62.15	3.30	8.80	37.46	45.15	2.74
Llama3.3-70B	94.74	49.26	94.72	40.30	41.75	**96.53**	88.95	37.25
Llama4-Scout-17B-16E	94.85	62.65	97.11	29.04	49.5	58.67	91.24	29.54
Mistral-Small-3.1-24B	94.65	28.96	49.55	24.55	36.65	66.98	77.55	16.87
Mistral-Large-123B	94.85	48.95	43.74	43.12	50.61	81.62	62.25	21.85
Closed-source models								
GPT-4o	95.22	64.45	45.64	42.60	45.30	94.04	57.65	24.87
GPT-4.1	**95.30**	73.29	45.71	59.90	63.80	93.89	59.31	31.74
o4-mini	95.25	90.72	98.25	55.05	64.61	85.22	93.75	56.23
o3	95.02	89.12	98.45	**71.51**	**78.62**	90.97	91.44	**64.87**
GPT-5	95.18	93.40	**98.52**	66.85	71.65	93.30	94.45	61.24
DeepSeek-V3	95.12	65.25	98.35	68.65	71.10	**96.55**	93.43	45.74
DeepSeek-R1	94.92	**93.61**	98.15	62.50	69.80	89.54	**97.92**	57.14
Claude3.5-Sonnet	94.75	80.85	97.61	33.05	65.85	50.19	92.61	33.04
Claude3.7-Sonnet	94.85	74.18	98.11	57.43	66.35	86.51	93.53	57.91
Gemini2.5-Flash-Preview	94.69	89.72	98.12	35.85	50.35	71.20	93.33	38.69
Gemini2.5-Pro-Preview	94.86	91.45	97.95	54.80	76.90	71.26	94.15	56.25

Table 2: The performance of open and closed-source models on the SMART benchmark.

with GPT-4.1.

R&R. For the Reflection & Refinement (R&R) task, the input question consists of the seed question paired with a chain-of-thought (CoT) solution containing deliberately injected errors. The outputs are the error categories and the corrected CoT. We define six error categories: arithmetic inaccuracies, omitted steps, hallucinated content, logical disorder, redundancy, and operator misuse. Error CoTs and their error types are generated according to predefined rules, so no additional verification is required. Detailed descriptions of these error types are provided in the Appendix Fig.10.

3.3.3 Quality Control

In Fig. 2, only the ground-truth for Understanding task and the questions for Arithmetic task are generated by LLMs, and thus require additional verification. To ensure the quality of the SMART benchmark, we implement a combined neuro-symbolic method and human verification procedure. This mechanism identifies and filters out low-quality samples and iteratively regenerates new data until the required quality standards are met.

Neuro-symbolic Verification. We employ neuro-symbolic verification to ensure the correctness of the SMT-LIB expressions used in the Arithmetic dimension. Directly comparing the generated SMT-LIB with ground-truth expressions is challenging. Instead, we leverage the Z3 Solver (De Moura and Bjørner, 2008) to automatically compute the result of a symbolic formula and compare it against the ground-truth answer from the original seed question, as a correct SMT-LIB expression should yield the same answer as its seed question. This generation–validation process is repeated until the SMT-LIB expression produces the correct answer.

Human Verification. The notation-base questions as question for the Arithmetic task and the structured key information as ground-truth for the Understanding task are both performed using GPT-4.1 and cannot be validated by the neuro-symbolic method. Thus, we follow the human verification protocol proposed by (Chen et al., 2024) to ensure the reliability of these generated data. Specifically, a randomly selected 10% subset is manually reviewed by human annotators. If the sampled data fail to meet quality standards, the low-quality portions are regenerated, then re-sampled and re-verified until they pass inspection.

As a result, SMART benchmark comprises 10,000 test instances, including 2,000 original seed questions and 8,000 carefully curated, task-specific variants. Through a combination of neuro-symbolic methods and human verification, we ensure that each instance meets quality standards.

4 Experiments

4.1 Models

We evaluate 22 recent open- and closed-source LLMs using our SMART framework, covering both general-purpose and reasoning-specialized models. To ensure deterministic outputs, the temperature is set to 0. For each dimension-specific task, we employ a three-shot prompting strategy.

4.2 Evaluation Metrics

Understanding. We adopt the LLM-as-a-Judge evaluation approach (Zheng et al., 2023), introducing the metric LLM@Un to evaluate the quality of generated structured information. To mitigate potential preference bias (Li et al.) inherent in LLM-based judging, we independently employ GPT-4.1 and DeepSeek-V3 as two judging models. Each judge evaluates the semantic similarity between the model-generated context and the reference ground-truth, assigning a similarity score ranging from 1 to 100. The final LLM@Un score is the average of the scores from GPT-4.1 and DeepSeek-V3.

Reasoning. We do not evaluate the correctness of the generated SMT-LIB expressions, since multiple logically equivalent solutions can exist for a problem. Instead, we validate these symbolic expressions by executing them with the Z3 Solver and evaluating the solver's output with the ground-truth answer of the seed question via rule-matching. The metric for the Arithmetic task is accuracy and is computed as $ACC@Re = \frac{N_{correct}}{N_{total}}$, which means the percentage of accurately answered questions.

Arithmetic. We evaluate answers of notation-based questions by comparing them with the ground-truth of the seed questions using rule matching, and use accuracy-based metrics ACC@Ar.

R&R. For the Reflection task, models are required to identify error categories within a given CoT. The outputs are compared with the ground-truth categories, and the accuracy metric is reported as ACC@R-t. For the Refinement task, the final answer is extracted from the refined CoT using rules. The refined solution is considered correct if the extracted answer matches the ground-truth of the seed question, and the accuracy metric is reported as ACC@R-m.

All-Pass Score. The All-Pass Score is an integrated metric (ACC@All) that combines performance across all evaluation dimensions. Specifically, a model achieves an All-Pass success if it simultaneously meets the following criteria: (1) obtaining a score of at least 90% on the Understanding task; (2) correctly solving the Reasoning task; (3) correctly solving the Arithmetic task; and (4) successfully completing the entire R&R task. We require at least 90% on Understanding to demand near-exact semantic extraction while allowing minimal lexical variation, which stabilizes LLM-as-a-Judge scoring and keeps All-Pass difficulty comparable across dimensions.

4.3 Performance on the SMART benchmark

The performance of the 22 evaluated LLMs on the SMART benchmark is detailed in Tab. 2, which reports the scores for all evaluation dimensions, All-Pass Score, and final answer accuracy.

4.3.1 Dimension specific-task results

Our results indicate that the LLMs generally demonstrate a strong capacity for problem understanding, with most LLM@Un scores exceeding 90%. This suggests a general proficiency in grasping relevant information and interpreting problem statements. However, significant performance disparities emerge in the Reasoning dimension, where ACC@Re scores range widely from 9.05% to 93.61%. A similar divergence is observed in the Reflection task, with the highest score (78.62%) being nearly nine times greater than the lowest (8.80%). These findings suggest that symbolic reasoning and error reflection capabilities represent critical bottlenecks, particularly for smaller models. In contrast, the Arithmetic and Refinement tasks appear relatively less challenging, with leading LLMs achieving near-perfect performance. For example, o3 attains an ACC@Ar of 98.45%, while DeepSeek-V3 reaches an ACC@R-m of 96.55%, demonstrating their strength in computational and corrective capabilities.

4.3.2 Granular insights

The SMART benchmark framework uncovers nuanced performance differences among LLMs that are obscured by final answer metrics alone. For example, while o4-mini and Claude3.7-Sonnet exhibit similar final answer accuracies (93.75% and 93.53%, respectively), o4-mini demonstrates markedly higher proficiency in the Reasoning dimension (90.72%) compared to Claude3.7-Sonnet (74.18%). A similar trend is observed when comparing o4-mini and DeepSeek-V3, further illustrating SMART benchmark's capability to reveal fine-grained gaps that traditional outcome-based metrics

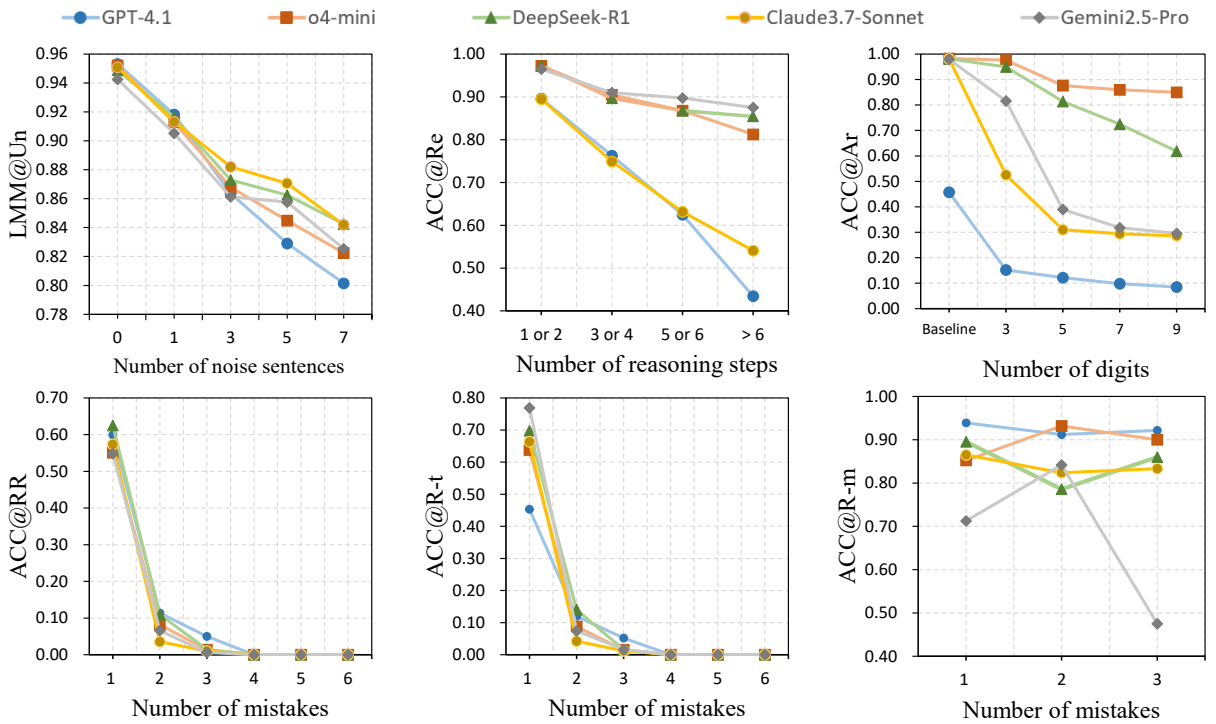

Figure 3: The performance across the varying difficulty settings for each SMART dimension.

Model	Perturbation	Final Answer		Understanding		Reasoning		Arithmetic		R&R	
		ACC@An	PD	LLM@Un	PD	ACC@Reason	PD	ACC@Ar	PD	ACC@RR	PD
GPT-4.1	Seed question	59.31	/	95.41	/	73.29	/	45.71	/	59.90	/
	+ Noise insertion	41.23	18.08↓	88.52	6.89↓	46.54	**26.75**↓	39.40	6.31↓	45.67	14.23↓
	+ Adding operation	28.45	30.86↓	91.32	4.09↓	57.53	15.76↓	20.64	**25.07**↓	48.12	11.78↓
	+ Numerical variation	24.81	34.50↓	92.83	2.58↓	52.81	20.48↓	18.34	**27.37**↓	51.86	8.04↓
Claude-3.7	Seed question	93.53	/	95.09	/	74.18	/	98.11	/	57.43	/
	+ Noise insertion	87.60	5.93↓	89.87	5.22↓	65.81	**8.37**↓	91.82	6.29↓	47.85	9.58↓
	+ Adding operation	76.21	17.32↓	92.34	2.75↓	54.38	19.80↓	67.43	**24.39**↓	50.78	6.65↓
	+ Numerical variation	62.74	30.79↓	93.17	1.92↓	58.92	15.26↓	58.29	**39.82**↓	52.85	4.58↓

Table 3: The performance degradation of evaluation dimensions when three types of perturbations are added to the seed questions. PD refers to the performance drop. The most affected dimension in each case is highlighted in bold.

Method	Refined Dimension	ACC@An	ACC@An with self-refine
Llama3.1-8B	Reasoning	45.15	43.85 (↓ 1.30)
Mistral-Small-24B	Reasoning	77.55	78.15 (↑ 0.61)
GLM-24B	Arithmetic	54.13	56.28 (↑ 2.15)
Qwen2.5-72B	Arithmetic	41.05	52.82 (↑ 11.77)

Table 4: Self-Refine prompting on a specific dimension.

miss. Additionally, although o4-mini and GPT-4.1 perform similarly on the Understanding and Reflection & Refinement dimensions, o4-mini's final answer accuracy (93.75%) is markedly higher than GPT-4.1's (59.31%). Our framework attributes this disparity primarily to GPT-4.1's lower capability in two key dimensions—Reasoning, where it scored 73.29% compared to o4-mini's 90.72%, and Arithmetic, where it achieved only 45.71% in contrast to o4-mini's 98.25%. Thus, the SMART framework facilitates a deeper analysis and interpretation of the underlying causes for performance differences.

4.3.3 All-Pass Score remains a challenge

The All-Pass Score effectively distinguishes model performance. The highest All-Pass Score achieved 64.87% for o3, which is substantially lower than its final answer accuracy of 91.44%. This metric offers a much clearer assessment of a model's mathematical capability and highlights considerable room for further improvement.

4.4 How Does Task Difficulty Impact Different Dimensions of SMART?

To investigate how task difficulty affects model performance across different SMART dimensions, we construct new sets of dimension-specific questions with varied difficulty levels and evaluate five

leading closed-source LLMs on this dynamic test set. Task difficulty is manipulated in the following ways: for the Understanding dimension, by varying the number of added irrelevant sentences; for the Reasoning dimension, by grouping questions according to the number of required reasoning steps; for the Arithmetic dimension, by changing the number of digits (referring to digit length in scientific notation, not the number of operands); and for the R&R dimension, by altering the mistakes introduced into the CoT.

It is important to note that modifying the number of digits in arithmetic questions changes the ground-truth answer. To ensure correctness, we simultaneously update both the numerical values in the arithmetic questions and their corresponding SMT-LIB representations, subsequently employing the Z3 Solver to generate new ground-truth answers. For the other dimensions, the ground-truth answers remain unchanged.

As shown in Fig. 3, increasing task complexity generally leads to notable performance degradation across all dimensions. Notably, GPT-4.1 and Claude3.7-Sonnet show pronounced sensitivity in the Reasoning dimension, with ACC@Re scores dropping sharply from approximately 90% to below 60% as the number of reasoning steps increases. In contrast, the remaining models maintain ACC@Re scores above 80% even with more than six reasoning steps. In the Arithmetic dimension, o4-mini demonstrates robust performance (ACC@Ar $> 80\%$) even with nine-digit numbers, whereas GPT-4.1's accuracy falls below 10%. For Reflection tasks, introducing just two error types into the CoT results in a steep decline in detection accuracy for all models, with none able to reliably detect all errors when four or more distinct mistake types are present.

4.5 How Do Fine-grained Dimensions Influence the Performance of Final Answer Accuracy?

Prior work has shown that LLMs experience significant drops in final answer accuracy when evaluated on perturbed versions of questions (Li et al., 2024b; Zhu et al., 2023; Li et al., 2024a). However, the underlying reasons for this degradation remain insufficiently explored. To address this gap, we adapt three perturbation strategies from (Li et al., 2024b) and apply them to both the seed questions and their corresponding dimension-specific variants, aiming to identify which dimensions are most susceptible to performance loss under these perturbations.

As shown in Tab. 3, all evaluated dimensions exhibit substantial performance drops (PD) under perturbations. When noise is introduced, the reasoning dimension is most affected for both GPT-4.1 (26.75%) and Claude 3.7-Sonnet (8.37%). Conversely, additional operations or numerical modifications lead to the greatest drops in the Arithmetic dimension. These results suggest that irrelevant information primarily undermines reasoning capabilities, while changes to operations or numeric values predominantly impact arithmetic proficiency. Ultimately, vulnerabilities across all dimensions collectively reduce final answer accuracy.

4.6 Improving LLMs via Self-Refine Prompting on Weak Dimensions

To enhance the mathematical capabilities of LLMs, we apply self-refinement prompting specifically to the weakest step identified in SMART. The specific prompts are provided in the Appendix. As shown in Tab. 4, targeting the reasoning or arithmetic dimension leads to notable performance gains for Gemma3-27B, Mistral-Small, and Qwen2.5-72B. In contrast, Llama3.1-8B experiences a slight performance drop, likely due to its limited capacity for self-reflection. These results demonstrate that the SMART framework is an effective diagnostic tool for pinpointing a model's weakest dimension and that targeted intervention on this dimension can improve mathematical performance.

5 Conclusion

We present SMART, a benchmark designed to evaluate the mathematical problem-solving capabilities of LLMs. Inspired by Pólya's theory of problem solving, SMART decomposes the reasoning process into four cognitive dimensions—Understanding, Reasoning, Arithmetic, and Reflection & Refinement—and introduces a novel All-Pass Score metric for comprehensive evaluation. We also propose a data curation and quality control framework that iteratively verifies generated test data to ensure reliability. Experiments on 22 open- and closed-source LLMs reveal that Reasoning and Reflection remain key bottlenecks, while targeted improvements on weak dimensions can enhance overall mathematical capability. We hope SMART provides a foundation for more systematic and interpretable evaluation of LLMs' reasoning processes in future research.

6 Limitation

While our proposed framework provides a comprehensive evaluation platform, it is important to acknowledge its inherent limitations. In particular, although Z3 and SMT-LIB effectively handle linear, integer, and some nonlinear constraints, their problem-solving capabilities are restricted. They are unsuitable for highly complex nonlinear problems and certain NP-complete combinatorial tasks. To overcome these limitations, future work will investigate using Lean (Moura and Ullrich, 2021) to formalize and prove complex mathematical theorems involving higher-order logic and intricate proof structures beyond SMT solvers' scope.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 model card.

anthropic. 2024. Claude 3.5 sonnet.

anthropic. 2025. Claude 3.7 sonnet and claude code.

Clark Barrett, Aaron Stump, Cesare Tinelli, and 1 others. 2010. The smt-lib standard: Version 2.0. In *Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, UK)*, volume 13, page 14.

Yuyan Chen, Chenwei Wu, Songzhou Yan, Panjun Liu, and Yanghua Xiao. 2024. Dr. academy: A benchmark for evaluating questioning capability in education for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3138–3167.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, and 1 others. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, and 1 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Google. 2024. Gemini 2.5: Our most intelligent ai model.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Huggingface. 2024. aime2024.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 1152–1157.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. Preference leakage: A contamination problem in llm-as-a-judge, 2025. *URL https://arxiv.org/abs/2502.01534*.

Jiatong Li, Renjun Hu, Kunzhe Huang, Yan Zhuang, Qi Liu, Mengxiao Zhu, Xing Shi, and Wei Lin. 2024a. Perteval: Unveiling real knowledge capacity of llms with knowledge-invariant perturbations. *Advances in Neural Information Processing Systems*, 37:10679–10706.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024b. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, Ming Zhang, and Qun Liu. 2023. Fimo: A challenge formal dataset for automated theorem proving. *Preprint*, arXiv:2309.04295.

Xuetao Ma, Wenbin Jiang, and Hua Huang. 2025. Problem-solving logic guided curriculum in-context learning for LLMs complex reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8394–8412, Vienna, Austria. Association for Computational Linguistics.

Meta. 2025. llama4-model-card.md. https://github.com/meta-llama/llama-models/blob/main/models/llama4/MODEL_CARD.md.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*.

MistralAITeam. 2024. Mistral-small-instruct-2409. https://huggingface.co/mistralai/Mistral-Small-Instruct-2409.

Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pages 625–635. Springer.

OpenAI. 2024. Learning to reason with llms.

OpenAI. 2025a. Introducing gpt-4.1 in the api.

OpenAI. 2025b. Introducing openai o3 and o4-mini.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

George Polya. 2014. *How to solve it: A new aspect of mathematical method*, volume 34. Princeton university press.

George Pólya and John Horton Conway. 1957. *How to solve it: A new aspect of mathematical method*. Princeton University Press Princeton.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25299–25346, Vienna, Austria. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S Yu, and Qingsong Wen. 2024. Large language models for education: A survey and outlook. *arXiv preprint arXiv:2403.18105*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhongshen Zeng, Pengguang Chen, Shu Liu, Haiyun Jiang, and Jiaya Jia. 2025. Mr-gsm8k: A meta-reasoning benchmark for large language model evaluation. In *The Thirteenth International Conference on Learning Representations*.

Zhongshen Zeng, Yinhong Liu, Yingjia Wan, Jingyao Li, Pengguang Chen, Jianbo Dai, Yuxuan Yao, Rongwu Xu, Zehan Qi, Wanru Zhao, and 1 others. 2024. Mr-ben: A meta-reasoning benchmark for evaluating system-2 thinking in llms. *Advances in Neural Information Processing Systems*, 37:119466–119546.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, William Song, Tiffany Zhao, Pranav Raja, Charlotte Zhuang, Dylan Slack, and 1 others. 2024. A careful examination of large language model performance on grade school arithmetic. *Advances in Neural Information Processing Systems*, 37:46819–46836.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024, Vienna, Austria. Association for Computational Linguistics.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. 2023. Dyval: Dynamic evaluation of large language models for reasoning tasks. In *The Twelfth International Conference on Learning Representations*.

Kaijie Zhu, Jindong Wang, Qinlin Zhao, Ruochen Xu, and Xing Xie. 2024. Dynamic evaluation of large language models by meta probing agents. In *International Conference on Machine Learning*, pages 62599–62617. PMLR.

A Appendix

A.1 Seed Question Collection and Filtering

The foundation of the SMART benchmark is a seed dataset comprising 2,000 problem instances. These were curated from seven widely-used mathematical problem datasets: GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), ASDiv (Miao et al., 2020), AQuA (Ling et al., 2017), MAWPS (Koncel-Kedziorski et al., 2016), MATH (Hendrycks et al., 2021), and problems from the AIME 2024 competition (Huggingface, 2024). These initial 2,000 seed questions, along with their subsequently generated dimension-specific variations (four per seed question), form the complete SMART benchmark, totaling 10,000 test instances.

Several criteria were applied during the selection and processing of these seed questions. To ensure consistency in question format, problems from the AQuA dataset, originally multiple-choice, were converted into an open-ended format; the textual content of the correct option was adopted as the ground-truth for these transformed questions. Furthermore, we excluded problems whose solutions fundamentally rely on operations not readily expressible or automatically verifiable using SMT-LIB, such as calculations involving the greatest common divisor (GCD), the least common multiple (LCM), or the determination of maximum/minimum values within a set. Questions requiring multiple distinct numerical values in their answers were also omitted. To maintain a baseline level of difficulty and focus on multi-step problem-solving, we filtered out mathematical problems that could be solved in a single reasoning step. The frequency distribution of reasoning steps for the selected 2,000 seed questions is depicted in Fig. 4, which illustrates that the majority of problems in the SMART benchmark involve multiple reasoning steps, with a notable concentration in the 2 to 7 step range.

A.2 Data Examples of SMART

Fig. 5 presents a data sample of SMART, which contains the seed question, the extracted context, the SMT-LIB expression, the arithmetic notation question, the CoT, and the final answer.

Fig.6 illustrates a seed question and its four-dimensional tasks. In the Understanding task, the model extracts key information from the seed question. In the Reasoning task, it solves the problem by producing an SMT-LIB formulation. In the Arithmetic task, it answers the corresponding notation-based arithmetic question. In the Reflection & Refinement task, it first identifies error categories in the provided CoT (Reflection) and then generates a corrected CoT (Refinement).

A.3 Prompts and Rules for SMART Data Curation

For each seed question, SMART generates distinct variants to evaluate the four targeted problem-solving dimensions. The generation and ground-truth creation for each dimensional task are described below.

A.3.1 Understanding

To generate ground-truth for the understanding dimension, we utilize GPT-4.1 to perform context extraction. The extracted context is structured into the following components: Problem Scenario (describing the overall context of the problem), Goal (specifying what needs to be solved), Known Quantities (listing explicitly provided numerical values or facts), Unknown Quantities (identifying variables or values to be determined), Relationships and Constraints (detailing connections and limitations between pieces of information), and Irrelevant Information (pinpointing details not pertinent to the solution). The prompt employed to guide GPT-4.1 in extracting this contextual information for ground-truth generation is presented in Fig. 7.

Dataset	Final Answer	Cognitive Stages				Source	Size
		Understanding	Reasoning	Arithmetic	R&R		
GSM8k (Cobbe et al., 2021)	✓					Human	8,792
GSM1k (Zhang et al., 2024)	✓					GSM8K	1,205
MATH (Hendrycks et al., 2021)	✓					2 datasets	12,500
MINIF2F (Zheng et al., 2022)	✓		✓			4 datasets	488
FIMO (Liu et al., 2023)	✓		✓			IMO	149
MR-GSM8k (Zeng et al., 2025)					✓	GSM8k	1,428
MR-Ben (Zeng et al., 2024)					✓	3 datasets	5,975
ProcessBench (Zheng et al., 2025)					✓	4 datasets	3,400
SMART (ours)	✓	✓	✓	✓	✓	7 datasets	10,000

Table 5: Compression between our SMART and other benchmarks.

A.3.2 Arithmetic

The arithmetic capability of LLMs is measured by their performance on notation-based arithmetic questions that share the same reasoning logic and final answer as the seed question. Directly converting a seed question into an arithmetic notation problem is challenging for LLMs. This difficulty arises because it requires simplifying complex natural language into structured mathematical operations while preserving the logical relationships between variables. Such a transformation is non-trivial, as the model must accurately interpret the problem's intent, handle ambiguous phrasing, and correctly map linguistic constructs to arithmetic operations. Therefore, to address this, we first generate an SMT-LIB representation of the seed question, which simplifies the reasoning logic among variables. Subsequently, we utilize GPT-4.1 to convert this SMT-LIB representation into the arithmetic notation problem, and then manually checked by human annotators. The prompt for that process is shown in Fig. 8 and Fig. 9.

A.3.3 Reflection & Refinement

For the Reflection & Refinement dimension, we first generate CoT solutions containing deliberate errors. To create these erroneous CoTs, one of the six error types defined in Fig. 10 (*e.g.*, arithmetic number error and skipped step) is uniformly sampled and injected into a correct CoT.

For the Refinement task, direct verification of the LLM-corrected CoT is complex. Instead, we evaluate the refined CoT by extracting the final answer of the refined CoT. The LLM is considered to have successfully passed the Refinement task if the extracted final answer matches the ground-truth of the original seed question.

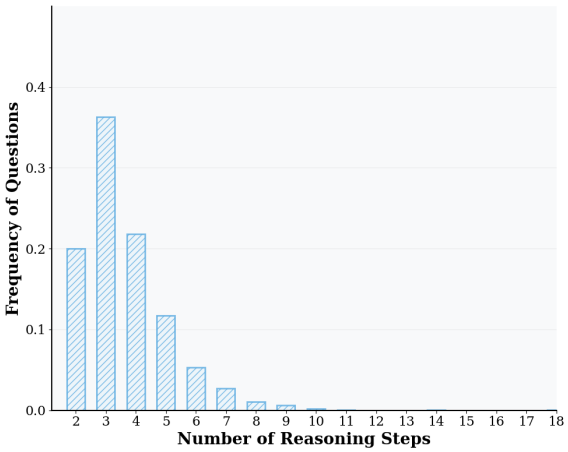

Figure 4: The reasoning step statistics of the seed question dataset.

A.4 Experiment Setting

We evaluate 22 recent open-source and closed-source LLMs using our SMART evaluation framework. The open-source models include Phi4 (Abdin et al., 2024), Gemma3 (Team et al., 2025), GLM4 (GLM et al., 2024), Qwen2.5 (Team, 2024), Qwen3 (Yang et al., 2025), Llama3 (AI@Meta, 2024), Llama4 (Meta, 2025), and Mistral (MistralAITeam, 2024). The closed-source models assessed are GPT-4o (Achiam et al., 2023), GPT-4.1 (OpenAI, 2025a), o4-mini, o3, GPT-5 (OpenAI, 2025b), DeepSeek-V3 (Liu et al., 2024), DeepSeek-R1 (Guo et al., 2025), Claude3.5 (anthropic, 2024), Claude3.7 (anthropic, 2025), and Gemini2.5 (Google, 2024).

All experiments were conducted on a Linux server equipped with two NVIDIA H800 GPUs (80GB). The GPUs were used for deploying and performing inference on open-source models. The Python version used was 3.9.20, and the version of the Transformers package was 4.46.0.

Model	Final Answer		Understanding		Reasoning		Arithmetic		R&R	
	Zero-shot	Three-shot	Zero-shot	Three-shot	Zero-shot	Three-shot	Zero-shot	Three-shot	Zero-shot	Three-shot
GPT-4.1	43.25	59.34	90.25	95.41	59.52	73.29	38.65	45.71	37.05	59.90
o4-mini	92.25	93.75	93.58	95.22	85.63	90.72	97.12	98.25	51.20	55.05
Gemini2.5 -Flash-Preview	90.65	93.33	91.12	94.25	81.47	89.72	95.41	98.18	30.74	35.85

Table 6: Zero-shot and three-shot prompt engineer strategy comparison in SMART.

Method	CoT to SMT-LIB	Question to SMT-LIB
GPT-4.1	92.58	73.29
O4-mini	95.68	90.72
DeepSeek-R1	97.54	93.61
Claude3.7-Sonnet	96.25	74.18
Gemini2.5-Pro	97.24	91.45

Table 7: Accuracy for transferring CoT and Seed question to SMT-LIB formula

A.5 Prompts for Evaluation in SMART

A.5.1 Understanding

We evaluate the generated structured key information by comparing it with the ground-truth structured key information and then use LLMs as the judge model to give the similarity score. The evaluation prompt is presented in the Fig. 11.

A.5.2 Reasoning

For the reasoning dimension, we introduce a symbolic formalization task to evaluate the symbolic reasoning capability of LLMs. The question for this task is the seed question, and LLMs are asked to generate the SMT-LIB expression of the question, without solving the problem. The prompt for this task is shown in Fig. 8. Subsequently, the Z3 Solver is used to compute the results of the generated SMT-LIB expression. Finally, we compare the results of SMT-LIM expression to the ground-truth of seed questions.

A.5.3 Arithmetic

For the arithmetic dimension, we introduce a numeric calculation task to evaluate the arithmetic capability of LLMs. The question for this task is a notation-based arithmetic problem, and LLMs are asked to solve it using the prompt shown in Fig. 12. Then, we compare the results of the notation-based questions to the ground-truth of seed questions.

A.5.4 Reflection & Refinement

For the reflection & refinement dimension, we propose an error correction task that requires LLMs to detect mistakes in the chain-of-thought (CoT) of seed questions, correct these mistakes, and generate a new answer for the seed question. The first step involves detecting errors in the CoT, given the question and CoT, with the answer being the specific name of the introduced error type. The evaluation prompt is shown in Fig. 13 and prompt for more errors in Fig. 14. If LLMs fail to detect all mistakes, they do not need to attend the following refinement task. The second step is to fix errors in CoT and generate a refined CoT with the prompt shown in Fig.15. The final step is to extract the new final answer of the refined CoT with rule-matching. If LLMs successfully detect all mistakes and generate the correct final answer based on the corrected CoT, we consider the model to have passed the error correction task.

A.6 The Self-Refine Prompting

We apply self-refinement prompting specifically to the weakest step identified in the problem-solving process to improve the mathematical capability of LLMs. The prompts used for self-refinement on the reasoning and arithmetic dimensions are shown in Fig.16 and Fig.17.

A.7 Difficulty setting for Dimension-specific Task

To generate dimension-specific questions with varying difficulty, we employ the following strategies:

For the understanding dimension, difficulty is controlled by progressively introducing irrelevant sentences, sourced from other problems, as 'noise' within the seed question's text. The number of such noise sentences dictates the complexity of the context extraction task. The ground-truth for these modified questions is updated by incorporating these noise sentences into the 'Irrelevant Information' category of the context extraction template.

For the reasoning dimension, question complexity is defined by the number of distinct mathematical operations ($e.g., +, -, \times, \div, mod$) required to formulate the solution. Problems are then categorized into multiple difficulty levels based on this

operational count.

In the arithmetic dimension, complexity is varied by altering the number of digits in the numerical values involved (*e.g.*, changing '12' to a five-digit number like '34.823'), rather than solely their magnitude, as precision with more digits presents a distinct challenge. The ground-truth for these modified arithmetic problems is obtained through our quality control method.

In the reflection & refinement dimension, difficulty is modulated by the number and types of mistakes deliberately injected into the Chain-of-Thought solutions. We randomly introduce a varying number of distinct error types (from the categories listed in Fig. 10) into the CoT to create different levels of challenge for the error detection and correction tasks.

Each seed question undergoes a two-stage transformation process. In the first stage, it is decomposed into four distinct, dimension-specific tasks that enable fine-grained evaluation of individual capabilities. In the second stage, these tasks are further rewritten using validated augmentation strategies to generate diverse variants that test the robustness and adaptability of LLMs. This hierarchical, two-phase generation framework enhances reliability and scalability by enabling comprehensive and granular evaluation while mitigating risks of overfitting and data contamination.

A.8 Examples for questions with different difficulty settings

Fig.18 shows examples of different difficulty settings for the understanding dimension evaluation. The sentences with a red background in the image represent irrelevant noise sentences, and the more noise sentences there are, the harder the task of extracting the effective context becomes.

Fig.19 shows examples of questions with different reasoning steps, which indicate different reasoning difficulties.

Fig.20 presents arithmetic questions with numbers in different digits. Numbers with more digits are more difficult for the arithmetic evaluation task.

Fig.10 presents CoT with different types of errors. CoT with more mistakes is more difficult for the reflection and refinement evaluation task.

A.9 Zero-shot vs. three-shot

We conducted additional comparative experiments evaluating model performance with and without three-shot examples across all stages. The results are presented in the Tab.6. As shown in the table, removing the three-shot examples leads to a significant performance drop across all models for dimensions. This confirms the substantial positive impact of including a few-shot example for these more complex reasoning stages. Our comparative results show that adding a few-shot example provides an improvement in all dimension scores across.

A.10 Ablation Study for Reasoning Task

LLMs demonstrate strong capabilities in translating natural language into formal language, and the formal translation process itself is not the primary cause of poor performance on the Reasoning task. To verify this, we conducted an ablation study ('CoT-to-SMT-LIB') where models converted correct natural language CoT into SMT-LIB formulas in Tab.7. Accuracy was verified via Z3 execution against ground-truth answers. All evaluated models achieved over 90% accuracy, indicating that the main challenge lies in generating correct reasoning paths—not in the formal translation. This supports our claim that the Reasoning dimension effectively captures a model's core mathematical Reasoning capability.

A.11 Is the Final Answer Accuracy Reliable for Measuring Mathematical Capability?

Given the impressive performance of LLMs and the potential for data contamination, there is a concern that models might solve problems correctly without possessing genuine underlying mathematical capability. To investigate this, we conduct experiments to compute confusion matrices comparing final answer correctness with performance on our SMART dimensions, as illustrated in Fig. 21. We posit that true mathematical capability is more accurately reflected by instances where LLMs correctly solve not only the original seed question but also simultaneously succeed in the corresponding reasoning and arithmetic dimension tasks.

Across all confusion matrices, the False Negative (FN) values are consistently non-zero. This indicates that LLMs can sometimes arrive at correct final answers through heuristic shortcuts or other opaque mechanisms, even when their intermediate reasoning or calculation processes are flawed. For example, GPT-4.1 exhibits a notable FN rate of 11.90% in the final answer and arithmetic confusion matrix. Similarly, Claude3.7-Sonnet shows an FN rate of 22.95% in the final answer and reason-

ing confusion matrix. These FN cases represent instances where final answer accuracy overestimates the model's grasp of the intermediate steps.

Conversely, False Positive (FP) scores denote cases where a model successfully completes an intermediate task but ultimately yields an incorrect final answer. Except for GPT-4.1, most evaluated LLMs exhibit relatively low FP rates across various confusion matrices, indicating that accurate intermediate reasoning generally correlates with correct final outputs.

True Positive (TP) scores capture instances where a model not only produces the correct final answer but also performs intermediate reasoning and arithmetic correctly. We regard this TP metric as a more reliable indicator of genuine mathematical problem-solving capability. For high-performing models such as o4-mini, DeepSeek-R1, and Gemini2.5-Pro-Preview, TP scores in the reasoning & arithmetic confusion matrix closely match their ACC@Fi values. In contrast, GPT-4.1 and Claude3.7-Sonnet exhibit significantly lower TP scores relative to their ACC@Fi, suggesting that their final answer accuracy may overestimate their true reasoning capabilities.

The Seed Question:
A container ship left from Dania Pier and traveled north. An aircraft carrier left 4 hours later traveling at 30 miles per hour in an effort to catch up to the container ship. After traveling for 8 hours the aircraft carrier finally caught up . What was the container ship 's average speed in miles per hour ?

The Extracted Context:
{problem scenario: A container ship and an aircraft carrier are traveling with the latter attempting to catch up
to the former.
goal: Determine the average speed of the container ship in miles per hour.
quantities: known: [Aircraft carrier speed: 30 miles per hour,
Aircraft carrier travel time: 8 hours,
Time after which aircraft carrier left: 4 hours],
unknown: [Container ship's average speed (in miles per hour)] ,
relationships and constraints: [Aircraft carrier and container ship meet after aircraft carrier travels for 8 hours,
Container ship travels for 4 hours longer than the aircraft carrier.],
Irrelevant information: [The ships left from Dania Pier, Direction of travel is north]}

The SMT-LIB Expression:

```
(set-logic QF_NRA)        (assert (= a 30))
(declare-fun a () Real)   (assert (= b 8))
(declare-fun b () Real)   (assert (= c 4))
(declare-fun c () Real)   (assert (= d (+ b c)))
(declare-fun d () Real)   (assert (= e (/ (* a b) d)))
(declare-fun e () Real)   (check-sat) (get-value (e))
```

The Arithmetic Notation Question:
a=30, b=8, c=4, d=b+c, e=(ab)/d, e?

The Chian-of-Thought:
To find the average speed of the container ship, we need to analyze the information given about both the container ship and the aircraft carrier. First, we note that the aircraft carrier left 4 hours after the container ship. This means that by the time the aircraft carrier started its journey, the container ship had already been traveling for 4 hours. Next, we know that the aircraft carrier traveled for 8 hours before catching up to the container ship. Therefore, the total time the container ship was traveling when it was caught up is: 4 hours (before the aircraft carrier left)+8 hours (while the aircraft carrier was traveling)=12 hours. Now, we need to calculate the distance the aircraft carrier traveled during its 8 hours of travel. The speed of the aircraft carrier is given as 30 miles per hour. Thus, the distance traveled by the aircraft carrier is: Distance=Speed×Time=30 miles/hour×8 hours=240 miles. Since the aircraft carrier caught up to the container ship, this means that the container ship also traveled the same distance of 240 miles in the 12 hours it was traveling. Now, we can find the average speed of the container ship using the formula: Average Speed=Total Distance/Total Time. Substituting the values we have: Average Speed=240 miles/12 hours=20 miles per hour. Thus, the average speed of the container ship is 20.0 miles per hour. #### 20.0

The Final Answer:
20.0

Figure 5: A data sample in the SMART benchmark.

Seed Question: Among the 900 residents of Aimeville, there are 195 who own a diamond ring, 367 who own a set of golf clubs, and 562 who own a garden spade. The town square features a large statue of a rabbit wearing sunglasses. In addition, each of the 900 residents owns a bag of candy hearts. There are 437 residents who own exactly two of these things, and 234 residents who own exactly three of these things. Find the number of residents of Aimeville who own all four of these things.
Answer: 73.0

Dimension 1: Understanding

Question 1: Extract key information of the seed question
Answer 1: {"Problem Scenario": "Residents of Aimeville own various items.",
"Goal": "Find the number of residents who own all four specified items." },
"Quantities": {"Known": ["Total number of residents: 900",
"Residents with diamond ring: 195", "Residents with golf clubs: 367",
"Residents with garden spade: 562", "Residents with candy hearts: 900",
"Residents with exactly two items: 437", "Residents with exactly three items: 234"],
"Unknown": ["Number of residents owning all four items"] },
"Relationships and Constraints": ["Each resident owns a bag of candy hearts.",
"Some residents own more than one specified items."],
"Irrelevant Information": ["The town square features a large statue of a rabbit wearing sunglasses. "]}

Dimension 2: Reasoning

Question 2: Solve the seed question with SMT-LIB formula
Answer 2: (set-logic QF_NRA) (declare-fun $count_r$ () Real) (declare-fun $count_c$ () Real) (declare-fun $count_s$ () Real)
(declare-fun $s1_{sum}$ () Real) (declare-fun $n1_{exact}$ () Real) (declare-fun $n2_{exact}$ () Real) (declare-fun $n3_{exact}$() Real)
(assert (= $count_r$ 195)) (assert (= $count_c$ 367)) (assert (= $count_s$ 562)) (assert (= $n1_{exact}$ 437)) (assert (= $n2_{exact}$ 234))
(assert (= $s1_{sum}$(+ $count_r$ $count_c$ $count_s$)))
(assert (= $s1_{sum}$ (+ $n1_{exact}$ (* 2 $n2_{exact}$) (* 3 $n3_{exact}$))))
(check-sat) (get-value ($n3_{exact}$))

Dimension 3: Arithmetic

Question 3: r=195, c=367, s=562, n0=900, n1=437, n2=234, s1=r+c+s, s1=n1+2n2+3n3, n3? **Solve this expression-based problem.**
Answer 3: 73

Dimension 4: Reflection & Refinement

Question 4: CoT: Let w, x, y, z denote the number of residents who own 1,2,3 and 4 of these items, respectively. We know $w + x + y + z = 900$, since there are 900 residents in total……. Plugging in x and y once more, we get w+4z=548. Solving $w + z = 229$ and $w + 4z = 448$, we get $z = 73$ #### 73. **Please carefully review each step in the CoT and determine error types.**
Answer 4: Arithmetic error [1].

Question 5: Provide corrected CoT solution with error fixed
Answer 5: ……Plugging in x and y once more, we get w+4z=448. Solving $w + z = 229$ and $w + 4z = 448$, we get $z = 73$ #### 73.

Figure 6: An overview of the SMART framework for evaluating the mathematical problem-solving process. The SMART contains four distinct dimensions. Each dimension is evaluated using dimension-specific tasks and metrics, ensuring a comprehensive assessment of the model's problem-solving capabilities. The yellow-highlighted symbolic assertion illustrates an inferred condition that is not explicitly stated in the original problem.

System Prompt:
You are a helpful assistant and good at following instructions.

User Prompt:
You are an expert analyst specializing in breaking down math word problems into their core components to facilitate understanding and solving. Below is a structured key information template. Do not add notes in the response.
Template:
{"problem description": {
"problem scenario": "Provide a brief, one-sentence description of the overall situation or context.",
"goal": "State clearly what needs to be calculated or found."},
"quantities": {
"known": [// List of explicitly stated numerical values or key quantities],
"unknown": [// List of quantities that need to be found (usually corresponds to the goal)},
"relationships and constraints": [// List of statements describing how quantities/entities relate, or rules governing the scenario],
"irrelevant information": [// List of information present in the text but likely not needed for calculation]}

Three-shot Examples
[Examples]

The Given Question:
[Question]

Now, analyze the next math problem. Generate the context of the question. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Answer of Task:
[Answer]

Figure 7: The prompt for LLMs to extract context from a seed question.

System Prompt:
You are a helpful assistant and good at following instructions.

User Prompt:
You are a highly skilled mathematician, NLP expert, and reasoning analyzer. Your task is to convert a Math word problem into an SMT-LIB expression. Follow these instructions:
1. Define Variables: Use abstract variable names (e.g., a, b, c) that do not reflect the actual meaning of the variables in the problem.
2. Formulate Constraints: Use mathematical relationships from the problem to establish constraints for the SMT-LIB formula.
3. SMT-LIB Syntax: Use proper SMT-LIB syntax. The logic should be set to QF_NRA or QF_NIA as appropriate. Include (check-sat) and (get-value ...) commands to verify satisfiability and extract the result.
4. Check: Ensure all the variables in SMT-LIB formula are declared.
5. Do not write comments.

Three-shot Examples
[Examples]

The Given Question:
[Seed question]

Now, analyze the next math problem. Generate the symbolic expression of the math word problem. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Answer of Task:
[SMT-LIB]

Figure 8: The prompt for LLMs to convert the seed question to a symbolic expression.

System Prompt:
You are a helpful assistant and good at following instructions.

User Prompt:
You are a highly skilled mathematician. You will be given a math problem, and your task is to convert the original SMT-LIB formula into a pure arithmetic notation problem, accurately reflecting the relationships and modified values without adding any additional background information. I will also give you the answer of the SMT-LIB formula, and you have to check that the answer of the math problem is same to the SMT-LIB' answer. Do not mention answer in the new generated question.

Three-shot Examples
[Examples]

The Given Symbolic Expression :
[SMT-LIB]

The Given Answer:
[Final answer]

Now, analyze the next SMT-LIB expression. Generate arithmetic notation problem of the SMT-LIB expression. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Answer of Task:
[Answer]

Figure 9: The prompt for LLMs to convert the SMT-LIB expression to an arithmetic notation problem.

1. Arithmetic Number Error

Definition: Randomly selects a number in the CoT and replaces it with a different, randomly generated number.

Example: The number of downloads of the program in the second month increased to 3*60 = <<3*60=180>>180. In the first two months, the total number of downloads of the program was 180+60 = <<180+60=280>>280. In the third month, the number of downloads of the program reduced by 30/100*180 = <<30/100*180=54>>54. There were 104-54 = <<180-54=126>>126 downloads in the third month. In the three months, the total number of downloads of the program was 126+240 = <<126+240=366>>366. #### 366.

2. Skipped Step Error

Definition: Randomly removes one sentence (a segment delimited by a period) from the CoT.

Example: The number of downloads of the program in the second month increased to 3*60 = <<3*60=180>>180. In the first two months, the total number of downloads of the program was 180+60 = <<180+60=240>>240. In the third month, the number of downloads of the program reduced by 30/100*180 = <<30/100*180=54>>54. There were 180-54 = <<180-54=126>>126 downloads in the third month. #### 366.

3. Hallucinatory Insertion Error

Definition: Randomly selects a sentence from another CoT and inserts it into a random position in the primary CoT.

Example: The number of downloads of the program in the second month increased to 3*60 = <<3*60=180>>180. In the first two months, the total number of downloads of the program was 180+60 = <<180+60=240>>240. In the third month, the number of downloads of the program reduced by 30/100*180 = <<30/100*180=54>>54. The total amount of fabric is 2+2=<<2+2=4>>4 bolt of fabric. There were 180-54 = <<180-54=126>>126 downloads in the third month. In the three months, the total number of downloads of the program was 126+240 = <<126+240=366>>366. #### 366.

4. Logical Order Error

Definition: Randomly selects two sentences in the CoT and swaps their positions.

Example: In the first two months, the total number of downloads of the program was 180+60 = <<180+60=240>>240. The number of downloads of the program in the second month increased to 3*60 = <<3*60=180>>180. In the third month, the number of downloads of the program reduced by 30/100*180 = <<30/100*180=54>>54. There were 180-54 = <<180-54=126>>126 downloads in the third month. In the three months, the total number of downloads of the program was 126+240 = <<126+240=366>>366. #### 366.

5. Redundant Output Error

Definition: Randomly selects a sentence in the CoT, duplicates it, and inserts the copy into a random position.

Example: The number of downloads of the program in the second month increased to 3*60 = <<3*60=180>>180. In the first two months, the total number of downloads of the program was 180+60 = <<180+60=240>>240. In the third month, the number of downloads of the program reduced by 30/100*180 = <<30/100*180=54>>54. In the third month, the number of downloads of the program reduced by 30/100*180 = <<30/100*180=54>>54. There were 180-54 = <<180-54=126>>126 downloads in the third month. In the three months, the total number of downloads of the program was 126+240 = <<126+240=366>>366. #### 366.

6. Incorrect Operator Error

Definition: Replaces a mathematical operator (+, -, ×, ÷) in the CoT with a different, randomly selected operator.

Example: The number of downloads of the program in the second month increased to 3*60 = <<3*60=180>>180. In the first two months, the total number of downloads of the program was 180+60 = <<180+60=240>>240. In the third month, the number of downloads of the program reduced by 30/100*180 = <<30/100*180=54>>54. There were 180-54 = <<180/54=126>>126 downloads in the third month. In the three months, the total number of downloads of the program was 126+240 = <<126+240=366>>366. #### 366.

Figure 10: Example of CoT with different errors.

System Prompt:
You are an expert evaluator tasked with assessing the semantic similarity between a reference answer and a model-generated answer.

User Prompt:
Your goal is to determine how closely the model-generated answer aligns with the reference answer in terms of meaning, content. Instructions:
Read both answers carefully: Understand the core information, details, and intent conveyed in each answer.
Evaluate semantic similarity: Compare the model-generated answer to the reference answer based on content and completeness.
Score the similarity: Provide a score between 1 and 100, where 100.0 indicates nearly identical meaning, structure, and content, and 1.0 indicates no relation. Scores can be fractional (e.g., 85.5).
Output Format:[Score]This prompt ensures that only the score is output, allowing for fractional values to reflect nuanced differences in similarity.

The Ground-truth Structed Key Information:
[GT SKI]

The LLM Generated Structed Key Information :
[SKI]

The Answer of Task:
[Score]

Figure 11: The prompt for LLM-as-a-Judge for evaluating the Understanding task.

System Prompt:
You are a helpful assistant and good at following instructions.

User Prompt:
You are a helpful assistant that solves notation-based arithmetic problem. You will be given a notation-based arithmetic problem, and your task is to provide the final answer which is enclosed in []. The final answer should not include any units or formulas. Additionally, ensure that your answer is rounded to five decimal places.

Three-shot Examples
[Examples]

The Given Question:
[Question]

Now, analyze the next math problem. Generate answer of the math problem. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Answer of Task:
[Answer]

Figure 12: The prompt for LLMs to solve the arithmetic notation problem.

System Prompt:
You are a helpful assistant and good at following instructions.

User Prompt:
I will provide a math problem with its corresponding CoT (Chain-of-Thought) reasoning. We have deliberately introduced one of the following six types of errors into the CoT:

1. Change Number: Randomly selects a number in the CoT and replaces it with a different, randomly generated number.
2. Delete Segment: Randomly removes one sentence (a segment delimited by a period) from the CoT.
3. Insert Segment: Randomly selects a sentence from another CoT (cot2) and inserts it into a random position in the primary CoT.
4. Swap Segments: Randomly selects two sentences in the CoT and swaps their positions.
5. Duplicate Segment: Randomly selects a sentence in the CoT, duplicates it, and inserts the copy into a random position.
6. Modify Operator: Replaces a mathematical operator (+, -, ×, ÷) in the CoT with a different, randomly selected operator.

Please carefully review each step in the CoT and determine whether any of the above types of errors are present. If you find errors, output the corresponding error type name and numbers.

Three-shot Examples
[Examples]

The Given Question:
[Seed question]

The Given CoT:
[CoT]

Now, analyze the next math word problem and its CoT. Find the errors in the CoT. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Answer of Task:
[Answer]

Figure 13: The prompt for LLMs to detect mistakes in the CoT.

System Prompt:
You are a helpful assistant and good at following instructions.

User Prompt:
I will provide a math problem with its corresponding CoT (Chain-of-Thought) reasoning. We have deliberately introduced one of the following six types of errors into the CoT:

1. Change Number: Randomly selects a number in the CoT and replaces it with a different, randomly generated number.
2. Delete Segment: Randomly removes one sentence (a segment delimited by a period) from the CoT.
3. Insert Segment: Randomly selects a sentence from another CoT (cot2) and inserts it into a random position in the primary CoT.
4. Swap Segments: Randomly selects two sentences in the CoT and swaps their positions.
5. Duplicate Segment: Randomly selects a sentence in the CoT, duplicates it, and inserts the copy into a random position.
6. Modify Operator: Replaces a mathematical operator (+, -, ×, ÷) in the CoT with a different, randomly selected operator.

Please carefully review each step in the CoT and determine whether any of the above types of errors are present. **More than one errors are in the CoT**. Output all the corresponding error type namse and numbers.

Three-shot Examples
[Examples]

The Given Question:
[Seed question]

The Given CoT:
[CoT]

Now, analyze the next math word problem and its CoT. Find **all errors** in the CoT. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Answer of Task:
[Answer]

Figure 14: The prompt for LLMs to detect more than one mistake in the CoT.

System Prompt:
You are a helpful assistant and good at following instructions.

User Prompt:
You are a highly skilled mathematician. I will provide you with an math word problem, and its corresponding solution steps (CoT) with error. Your task is to provide the corrected CoT (solution steps) with the error fixed.

Three-shot Examples
[Examples]

The Given Question:
[Seed question]

The Given CoT:
[CoT]

Now, analyze the next math problem and its CoT. Generate fixed CoT of the math problem. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Answer of Task:
[Answer]

Figure 15: The prompt for LLMs to correct the mistakes in the CoT.

System Prompt:
You are a helpful assistant and good at following instructions.

User Prompt:
Your task is to solve the problem and then refine the **Reasoning** aspect of your solution.
STEP 1 – Initial Solution
• Solve the problem.
• Present a concise chain of reasoning steps (R1, R2, …) followed by your boxed final answer.

STEP 2 – Reasoning Check
• Re-read your own reasoning steps.
• Mark the single step whose logical justification is the weakest or most doubtful. Label it "Weak-R".
• Briefly explain in one sentence why this step might be fragile.

STEP 3 – Reasoning Refinement
• Rewrite only the Weak-R step, providing clearer logic or the missing justification.
• Update any downstream reasoning if necessary.
• Re-state the boxed final answer (change it if the refinement alters the result).
• End with the line: "Refinement complete – reasoning strengthened."

The Given Question:
[Question]

Now, analyze the next math problem. Generate the answer of the question. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Answer of Task:
[Answer]

Figure 16: The prompt for LLMs to self-refine the Reasoning dimension.

System Prompt:
You are a helpful assistant and good at following instructions.

User Prompt:
Your task is to solve the problem and then refine the **Arithmetic** aspect of your solution.

STEP 1 – Initial Solution
• Solve the problem.
• Present your reasoning steps (R1, R2, …) and arithmetic calculations (A1, A2, …).
• End with a boxed final answer.

STEP 2 – Arithmetic Check
• Re-examine each arithmetic step (A k).
• Identify the single calculation that is most error-prone or uncertain. Label it "Weak-A".
• In one sentence, explain why this calculation might be unreliable (e.g., multi-digit operation, fraction simplification).

STEP 3 – Arithmetic Refinement
• Recompute only the Weak-A step carefully, showing the detailed calculation and a quick sanity check.
• Propagate any changes to later steps if needed.
• Re-state the boxed final answer (update if the refinement changes it).
• End with: "Refinement complete – arithmetic strengthened."

The Given Question:
[Question]

Now, analyze the next math problem. Generate the answer of the question. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Answer of Task:
[Answer]

Figure 17: The prompt for LLMs to self-refine the Arithmetic dimension.

The Seed Question:
Adam bought 13 boxes of chocolate candy and gave 7 to his little brother . If each box has 6 pieces inside it , how many pieces did Adam still have ?

The Question with one noise sentence:
Adam bought 13 boxes of chocolate candy and gave 7 to his little brother. If each box has 6 pieces inside it , how many pieces did Adam still have ? Mary is baking a cake. She already put in 12 cups of flour.

The Question with three noise sentences:
Adam bought 13 boxes of chocolate candy and gave 7 to his little brother. If each of them had the same number of cookies. Anna is able to buy 5 more articles for $300 after the price of each article decreased by 15%. If each box has 6 pieces inside it , how many pieces did Adam still have ? Mary is baking a cake. She already put in 12 cups of flour.

The Question with five noise sentences:
He has a total of 40 cannolis in his house. Adam bought 13 boxes of chocolate candy and gave 7 to his little brother. If each of them had the same number of cookies. At Allan's house, there is twice as much corn as cannolis. Anna is able to buy 5 more articles for $300 after the price of each article decreased by 15%. Sam 's dog had puppies and 8 had spots. If each box has 6 pieces inside it , how many pieces did Adam still have?

Figure 18: Example of questions with a different number of noise sentences.

The Question with One or Two Reasoning Steps:
James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?

The Question with Three or Four Reasoning Steps:
Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?

The Question with Five or Six Reasoning Steps:
John drives for 3 hours at a speed of 60 mph and then turns around because he realizes he forgot something very important at home. He tries to get home in 4 hours but spends the first 2 hours in standstill traffic. He spends the next half-hour driving at a speed of 30mph, before being able to drive the remaining time of the 4 hours going at 80 mph. How far is he from home at the end of those 4 hours?

The Question with more than Six Reasoning Steps:
Adrien's total salary was 30 percent higher than Lylah's. Four years later, his salary had increased, and he was earning 40% more than what he was making four years ago. If Adrien's and Lylah's salary increased simultaneously, and Adrien earned $40000 four years ago, calculate the total salary the two were receiving four years later?

Figure 19: Example of questions with different reasoning steps.

The Arithmetic Question:
What is the value of d if a is equal to 36, b is equal to 13, c is equal to 49, and d is the sum of a, b, and c?

The Arithmetic Question with Numbers in Three digits:
What is the value of d if a is equal to -29.6, b is equal to -6.34, c is equal to 976, and d is the sum of a, b, and c?

The Arithmetic Question with Numbers in Five digits:
What is the value of d if a is equal to 7938.7, b is equal to 74.180, c is equal to -21327, and d is the sum of a, b, and c?

The Arithmetic Question with Numbers in Seven digits:
What is the value of d if a is equal to 1,323,984, b is equal to 1,823,649, c is equal to 3.683971, and d is the sum of a, b, and c?

The Arithmetic Question with Numbers in Nine digits:
What is the value of d if a is equal to -732716160, b is equal to 2330874.42, c is equal to -340169802, and d is the sum of a, b, and c?

Figure 20: Example of arithmetic questions with numbers in different digits.

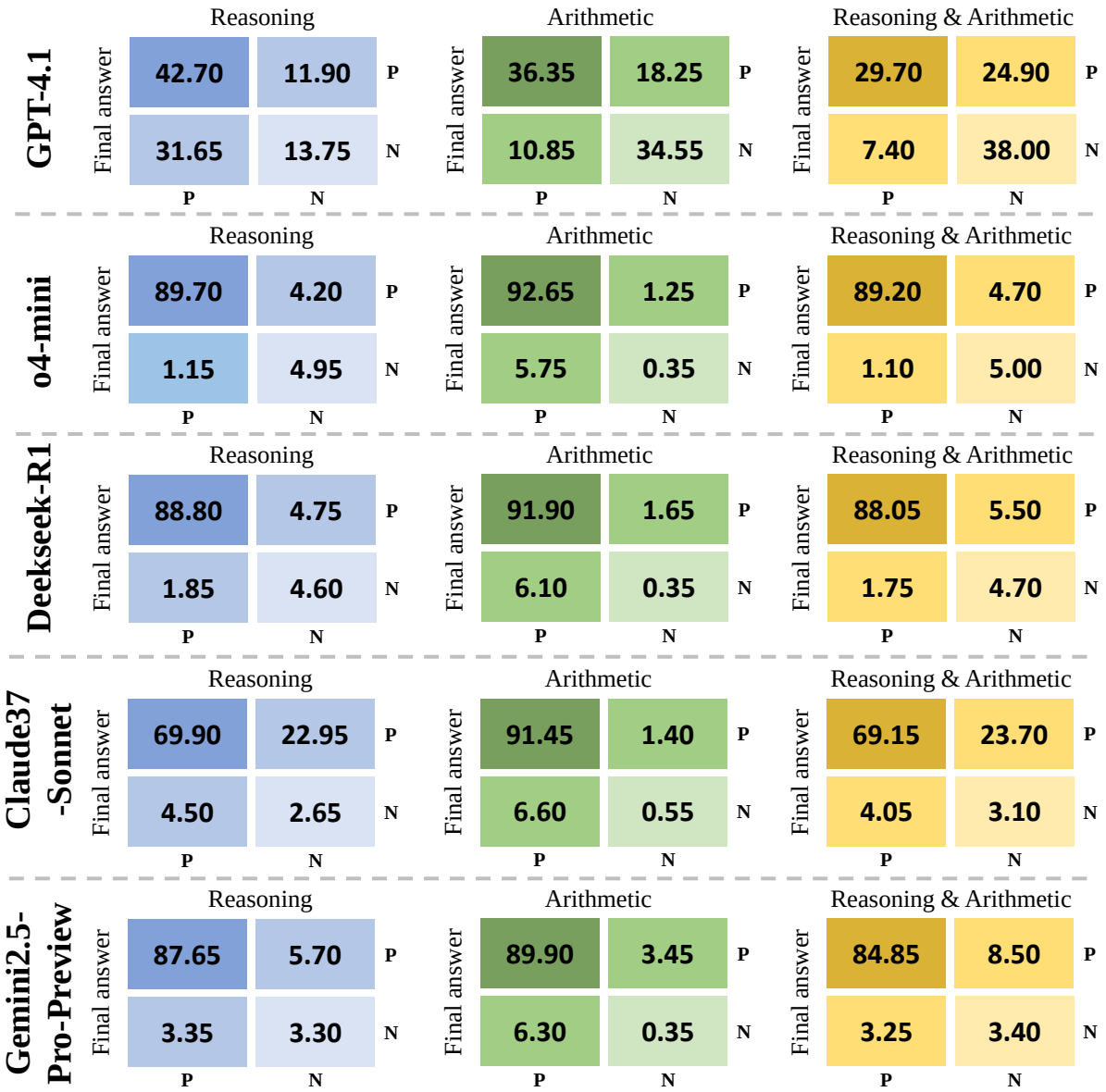

Figure 21: The confusion matrix of the final answer and other dimensions. P means Positive, and N means Negative.