SMART: A Self-Validating Multi-Dimensional Assessment Framework for Evaluating LLMs' Mathematical Problem-Solving Processes

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated impressive performance across various mathematical benchmarks. However, concerns persist over whether these high scores indicate genuine mathematical capability or merely superficial pattern recognition. Furthermore, we contend that the commonly used metric of final answer accuracy fails to capture the performance of LLMs on nuanced factors, as it reflects a composite outcome influenced by multiple factors. This motivates us to introduce SMART (Self-Validating Multi-Dimensional Assessment Framework), which deconstructs the problem-solving process into four key dimensions: understanding, reasoning, arithmetic, and reflection & refinement. Crucially, SMART does not evaluate based on final answer accuracy but instead designs separate tasks and evaluation methods for each dimension, enabling detailed and controllable assessments that decouple individual factors. Additionally, we propose a self-validating mechanism that iteratively generates and verifies test data, ensuring benchmark reliability and scalability. We evaluate 13 open-source and closed-source LLMs using SMART, and our findings reveal that final answer accuracy is insufficient for evaluating true mathematical problem-solving capabilities. Our analysis highlights symbolic reasoning and reflection & refinement as the key factors that distinguish LLM performance. We hope these insights will provide valuable guidance for advancing LLMs' true mathematical competence, and we will release our code and benchmark upon acceptance.

1 Introduction

Large language models (Achiam et al., 2023; Wei et al., 2022) have demonstrated impressive performance across various natural language processing tasks and are increasingly being integrated into real-world applications (Singhal et al., 2023; Wang et al., 2024; Jiang et al., 2024), showcasing their ability to perform complex reasoning with human-level accuracy (OpenAI, 2024; Guo et al., 2025). Given their widespread deployment, it is essential to assess LLMs' reasoning abilities to ensure their reliability, and effectiveness in real-world tasks.

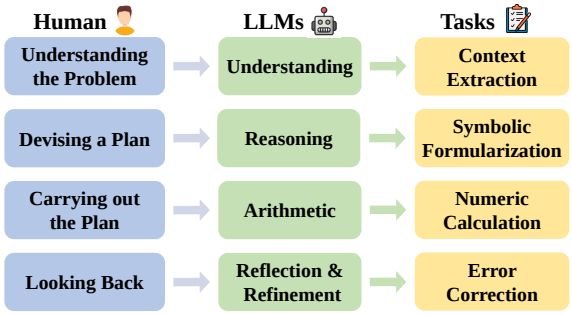

Figure 1: Illustration of the SMART framework for evaluating LLMs. According to Polya's theory, humans solve mathematical problems through four steps: understanding the problem, devising a plan, carrying out the plan, and looking back. Based on these steps, we define four evaluation dimensions and corresponding tasks to systematically assess LLMs.

Numerous benchmarks (Cobbe et al., 2021; Ling et al., 2017; Patel et al., 2021; Miao et al., 2020; Koncel-Kedziorski et al., 2016) have been proposed to evaluate LLMs on mathematical reasoning tasks. However, a significant concern with these public benchmarks is their susceptibility to data contamination, which can lead to inflated performance and skewed evaluations (Oren et al., 2023; Zhu et al., 2023, 2024; Li et al., 2024a). Moreover, most benchmarks primarily focus on final answer accuracy, neglecting the fact that this metric is indeed an aggregate effect of multiple factors, and fail to adequately measure the underlying reasoning process. (Cobbe et al., 2021; Ling et al., 2017; Patel et al., 2021; Miao et al., 2020; Koncel-Kedziorski et al., 2016). With the rapid advancement of LLMs, traditional accuracy-based metrics have reached saturation, limiting the effectiveness of final-answer accuracy as a differentiating metric. For instance,

Llama-3.1-70B achieves an impressive 95.1% accuracy on the GSM8k dataset (AI@Meta, 2024). Thus, it is imperative to broaden the scope of evaluation metrics to not only assess the accuracy of the answers but also to evaluate the depth of reasoning. Furthermore, the construction of high-quality benchmark datasets is resource-intensive, requiring significant human labor and time for annotation (Mirzadeh et al., 2024; Kurtic et al., 2024). When leveraging advanced LLMs like GPT-4o for data generation, ensuring the correctness and quality of the generated content remains a major challenge, as LLMs may produce errors or inconsistencies. To address this, existing approaches often rely on human annotators to rigorously review and refine the generated question variations and answers (Zhu et al., 2024; Zheng et al., 2023). However, this manual refinement process is not scalable, creating a significant obstacle to generating reliable, high-quality datasets at scale.

In light of these issues, we propose a novel evaluation framework called the Self-Validating Multi-Dimensional Assessment Framework (SMART). Inspired by Polya's problem-solving theory (Polya, 2014), we deconstruct the process of tackling mathematical problems into four key evaluation dimensions: understanding, reasoning, arithmetic, and reflection & refinement, each addressing a distinct aspect of problem-solving performance. We design dimension-specific tasks with well-defined expected outputs for each evaluation dimension of SMART, as shown in Fig. 1. This hierarchical structure allows SMART to break down the complex cognitive task in depth and assess different aspects of mathematical problem-solving independently. This comprehensive evaluation framework offers valuable insights into the strengths and limitations of models, guiding improvements in both algorithmic design and applications. Furthermore, we introduce a self-validating mechanism that automatically verifies the quality of generated dimension-specific testing data. This iterative process continues until the generated dataset meets the required standards, ensuring both scalability and reliability of the SMART benchmark.

We evaluate 13 recently released open- and closed-source LLMs of various scales using our SMART framework. Experimental results reveal that while all models perform well in the understanding dimension, they struggle significantly in the reasoning and reflection & refinement dimension. Additionally, we investigate the key factors influencing performance across different dimensions. Finally, we deeply analyze the relationship between final answer accuracy and the decomposed dimensions, and propose a new metric for measuring the truly mathematical capability of LLMs.

Our main contributions are as follows:

- We propose SMART, a self-validating multi-dimensional assessmemt framework designed to breakdown the problem-solving process of LLMs into four key dimensions: understanding, reasoning, arithmetic, and reflection & refinement. This framework allows for a deeper understanding of the cognitive processes involved in problem-solving, rather than solely emphasizing the accuracy of the final answer.

- We introduce a self-validating mechanism that automatically verifies the quality of generated test data, reducing reliance on human annotations. This self-validating approach ensures the benchmark's scalability and reliability.

- We conduct extensive evaluation and highlight significant disparities in the mathematical capabilities of different LLMs, with a detailed, dimension-specific, interpretable analysis. We anticipate that these insights will serve as a valuable foundation for future advancements in the development and optimization of LLMs.

2 Related Work

Mathematic 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, like GSM8K (Cobbe et al., 2021), to high-school-level datasets, like MATH (Hendrycks et al., 2021), and extend to expert-level datasets, like FrontierMath (Glazer et al., 2024). Their scope covers a broad range of mathematical domains, including geometry, number theory, and real analysis. However, despite the increasing difficulty of these benchmarks, it remains unclear whether LLMs truly comprehend mathematical concepts or merely rely on probabilistic pattern-matching to generate correct answers (Mirzadeh et al., 2024). Furthermore, the widespread use of these benchmarks increase the risk of data contamination, potentially inflating performance evaluations (Li et al., 2024a).

Dynamic evaluation. To address these concerns, recent studies have explored dynamic evaluation ap-

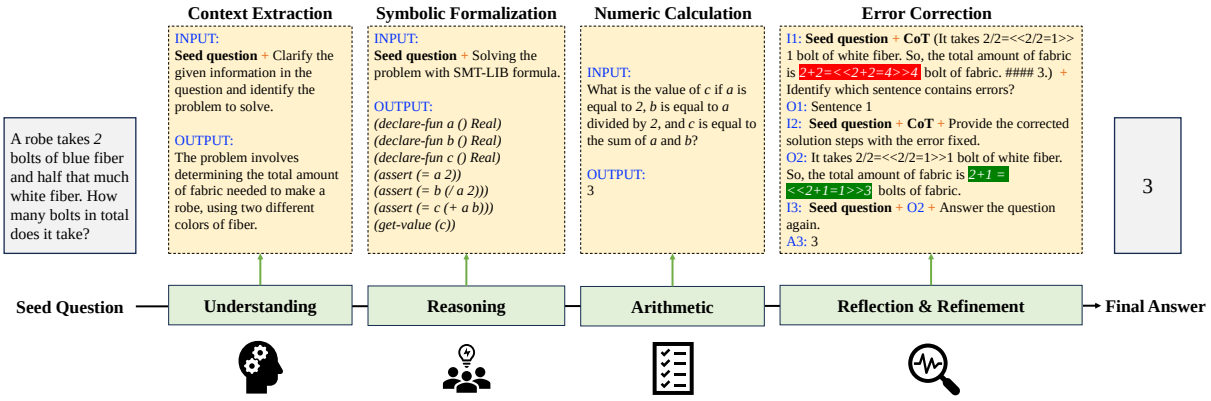

Figure 2: An overview of the SMART framework for evaluating the mathematical problem-solving process, which is broken down into four distinct steps: understanding, reasoning, arithmetic, and reflection & refinement. Each step is evaluated using dimension-specific task and metrics, ensuring a thorough and comprehensive assessment of the model's problem-solving capabilities.

proaches (Zhu et al., 2023, 2024; Fan et al., 2023), which generate adaptive test data based on predefined transformation rules. For example, GSM-Plus (Li et al., 2024b) introduces eight perturbation strategies, such as numerical and arithmetic variations, while GSM-Symbolic (Mirzadeh et al., 2024) generates diverse problem variants using symbolic templates. These approaches aim to mitigate data leakage and improve robustness in assessments. Despite these advancements, current evaluation methodologies remain limited. They predominantly focus on final-answer accuracy, which fails to capture an LLM's reasoning process in a systematic manner. Additionally, they do not explicitly assess the diverse cognitive skills required for mathematical problem-solving, nor do they provide fine-grained insights into the underlying logical reasoning steps.

These limitations underscore the need for a comprehensive, interpretable, and scalable evaluation framework that can decompose the problem-solving process, assess reasoning at multiple levels, and minimize reliance on human verification. To bridge this gap, we introduce SMART, a self-validating multi-dimensional assessment framework designed to systematically evaluate LLMs' mathematical reasoning abilities while addressing benchmark reliability and scalability challenges.

3 Methodology

We introduce SMART, a novel evaluation paradigm inspired by Polya's problem-solving theory (Polya, 2014). SMART systematically assesses the reasoning process in solving mathematical word problems by deconstructing problem-solving process into a sequence of logical and systematic steps, as illustrated in Fig. 2. The framework evaluates mathematical problem-solving across four key dimensions: understanding, reasoning, arithmetic, and reflection & refinement. To minimize cross-dimensional interference, we design distinct evaluation tasks and metrics tailored to each dimension, ensuring a more granular and interpretable analysis of LLMs' problem-solving abilities.

3.1 SMART Framework

3.1.1 Understanding

The first dimension of the SMART framework is understanding, which assesses how well LLMs can comprehend the problem with clarity and completeness before attempting to solve it. To measure this, we design a context extraction task that asks LLMs to identify the given information or conditions in the problem and clearly determine what needs to be solved. This task focuses solely on assessing LLMs' ability on understanding of the problem, without requiring reasoning or calculation. The context extracted by GPT-4o (Achiam et al., 2023) serves as the ground truth. To evaluate the understanding dimension performance of LLMs, we use sentence similarity (Reimers, 2019) between the contexts generated by the models and the ground truth as the understanding dimension metric.

3.1.2 Reasoning

The second dimension of the SMART framework is reasoning, which evaluates the ability of LLMs to construct a coherent and logical plan for solving a given problem. To assess this, we design a symbolic formalization task where LLMs are asked to model the math word problems via formal sym-

bolic expressions. This task measures the LLM's mathematical modeling ability to capture the problem's logical structure rather than solving it.

We adopt the SMT-LIB language (Barrett et al., 2010), a widely used standardized notation, which is compatible with SMT solvers such as Z3 (De Moura and Bjørner, 2008), SymPy (Meurer et al., 2017), and MathSAT (Cimatti et al., 2013). These solvers can compute results from the generated symbolic formulas, allowing us to compare the results of different SMT-LIB expressions to the ground truth answers of the problems. We calculate the accuracy of the correct SMT-LIB formula results as the reasoning metric, which serves as an indicator of how well the model has captured the logical structure and formulation of the problem.

In our evaluation, LLMs are asked only to generate the logical formulas, without being required to compute the final answer. By decoupling reasoning from arithmetic computation, we ensure that the evaluation captures the LLM's true reasoning capabilities. By focusing on symbolic formalization, we assess the model's ability to identify and represent the underlying logic, offering a more precise evaluation of reasoning performance.

3.1.3 Arithmetic

The arithmetic dimension assesses the model's ability to perform pure numerical calculations involving basic operations such as addition, subtraction, multiplication, and division. Unlike the symbolic formalization task, where LLMs are required to generate SMT-LIB formulas from the seed question, this phase directly evaluates the model's ability to solve arithmetic problems that follow the same reasoning logic and yield the same final answer as the seed question.

To generate these arithmetic problems, we utilize GPT-4o to convert the SMT-LIB formula into a corresponding arithmetic expression. This simplification reduces the math word problem to one that focuses solely on numerical values and their relationships, omitting any background information. In Fig. 2, the numeric calculation task is to compute the sum of a and b if $a = 2$ and $b = a/2$.

This design ensures that the numeric calculation task focuses exclusively on basic mathematical operations, effectively isolating the evaluation of arithmetic capabilities while minimizing the influence of other dimensions, such as reasoning and understanding. As a result, this task provides a precise and targeted assessment of the model's arithmetic proficiency, accurately reflecting its numerical computation abilities.

3.1.4 Reflection & Refinement

The final dimension of the SMART framework is reflection and refinement, which evaluates LLMs' ability to review their problem-solving process and improve their answers through self-correction. In the error correction task, we deliberately introduce incorrect steps into a chain-of-thought (CoT) solution (Wei et al., 2022), such as altering $2 + 2 = 4$ to $2 + 1 = 3$. LLMs are then tasked with identifying the erroneous statements, correcting them, and generating a refined answer. If the LLMs fail to detect all errors in the CoT, they will not proceed to the subsequent refinement task. LLMs with strong reflection and refinement capabilities can enhance their performance and stability through self-correction, demonstrating the ability to iteratively improve their problem-solving processes.

3.2 The Self-validating Mechanism

3.2.1 Dataset construction

To construct the SMART benchmark, we aggregate five widely used math word problem datasets: GSM8k (Cobbe et al., 2021), SVAMP (Patel et al., 2021), ASDiV (Miao et al., 2020), AQuA (Ling et al., 2017), and MAWPS (Koncel-Kedziorski et al., 2016). These datasets collectively form the seed question dataset, comprising 6,862 testing samples. Using GPT-4o, we generate variations of the seed questions tailored to the evaluation dimensions of the SMART framework, with each variation designed to test a specific aspect of the problem-solving process. Together, the seed questions and their dimension-specific variants constitute the SMART benchmark, which includes a total of 34,310 testing samples. Further details on the data generation process and examples can be found in the Appendix B.

3.2.2 Dataset Verification

While many prior works (Wang et al., 2023; Zhu et al., 2024; Li et al., 2024a,b,c) have employed LLMs to generate variations of seed questions, they often require extensive manual verification by human annotators to ensure the correctness of both the questions and their ground-truth answers. This manual approach is resource-intensive and limits scalability. To overcome this limitation, we propose a self-validating mechanism that automates the verification of generated datasets and their an-

Task	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
SMT-LIB	86.29	93.37	96.88	97.81	98.17	98.92
Context	88.72	95.78	98.11	99.21	\	\
Arithmetic Problem	85.21	92.42	95.66	97.24	98.32	99.05

Table 1: The pass rate of self-validating process across six iterations.

notations, reducing reliance on human intervention while maintaining accuracy and consistency.

For the context extraction task, our self-validating mechanism decomposes the question into a SMT-LIB formula and context, then recombines them to regenerate the question, enabling a cycle consistency check. The SMT-LIB formula is validated using the Z3 solver to ensure it produces the same answer as the original seed question. Subsequently, the context and the validated SMT-LIB formula are input into an LLM to regenerate a math word problem. If the regenerated question yields the same answer as the seed question, the context is deemed valid, as it contains clear and complete background information about the problem. Otherwise, the context is discarded and regenerated until it meets this consistency criterion.

For the numeric calculation task, LLMs are required to re-extract the SMT-LIB formula from the generated arithmetic question. The generated arithmetic question is deemed consistent if the solution obtained from the extracted SMT-LIB formula matches the answer to the original seed question. Otherwise, it is discarded and regenerated until the extracted formula aligns with the original seed question's solution.

Through this self-validating mechanism, we generate high-quality SMT-LIB formulas, context extractions, and arithmetic problems for all seed questions. Tab. 1 presents the pass rates for generating testing data during the self-validation process. After six iterations, a small number of failed cases are manually corrected to ensure benchmark reliability.

4 Experiments

4.1 Experimental Setup

We evaluate eight open-source models of varying sizes, ranging from 7B to 72B parameters, including Llama3 (AI@Meta, 2024), Qwen2.5 (Yang et al., 2024; Team, 2024), Mistral (MistralAITeam, 2024a,b), Phi4 (Abdin et al., 2024), and Gemma2 (Team et al., 2024). Additionally, we assess five state-of-the-art closed-source models, including o1-mini (OpenAI, 2024), GPT-4o (Achiam et al., 2023), Gemini 1.5 Pro (Reid et al., 2024), Qwen Max (Team, 2024), and DeepSeek-V3 (Liu et al., 2024). To ensure a fair comparison, we set the generation temperature to 0.1 for all models.

4.2 Performance of LLMs on the SMART Benchmark

We evaluate 13 open-source and closed-source models on the SMART benchmark, and the results are presented in Table 2. The final answer dimension measures the accuracy of the model's direct response to the problem (ACC@An). The understanding dimension quantifies the similarity between the extracted context and the ground truth context (SIM@Un). In the reasoning dimension, we use the Z3 solver to verify the correctness of the generated SMT-LIB formula, with the accuracy of the SMT-LIB output serving as the reasoning evaluation metric (ACC@Reason). The arithmetic dimension evaluates the accuracy of the model in solving pure arithmetic problems (ACC@Ar). The Reflection & Refinement (R & R) dimension combines both mistake detection and correction, with accuracy calculated as (ACC@RR). Specifically, the reflection dimension measures the model's ability to identify mistakes (ACC@Reflect), while the refinement dimension evaluates the accuracy of generating refined answer (ACC@Refine). More details and examples can be found in Appendix C.

The SMART benchmark reveals significant performance gaps among LLMs. Open-source models with 8B parameters exhibit poor score in reasoning, arithmetic, and reflection & refinement dimensions. As the scale of LLMs increases, the performance across these dimensions improves, aligning with the empirical findings of the scaling law (Kaplan et al., 2020). For closed-source models, while both DeepSeek-V3 and o1-mini achieve saturated scores in ACC@An, DeepSeek exhibits a significant gap in ACC@Reason (75.17%) and ACC@RR (19.09%) when compared to o1-mini, which scores 92.84% in ACC@Reason and 61.21% in ACC@RR. Gemini1.5-pro shows a 12.29% lower ACC@An than DeepSeek-V3, primarily due to its 15.55% lower score in ACC@Ar. Thus, the SMART benchmark effectively illustrates

Model	Answer	Understanding	Reasoning	Arithmetic	R & R	Reflection	Refinement
	ACC@An	SIM@Un	ACC@Reason	ACC@Ar	ACC@RR	ACC@Reflect	ACC@Refine
Open-source models							
Qwen2.5-7B	57.75	81.62	31.04	55.02	3.67	4.08	90.50
Llama3.1-8B	63.16	76.47	25.58	60.78	0.57	4.41	12.87
Mistral-Nemo	70.33	**83.69**	31.79	78.61	6.21	8.62	72.08
Phi-4-14B	93.66	79.47	71.47	95.48	7.22	20.40	35.43
Mistral-Small	70.83	83.66	31.57	78.59	11.63	15.78	71.66
Gemma2-27B	68.62	81.44	61.42	62.36	13.58	14.47	93.86
Qwen2.5-72B	77.48	83.22	**79.85**	76.48	27.06	28.11	96.31
Llama-3.3-70B	**94.08**	79.56	76.21	**95.92**	**36.38**	**37.62**	**96.71**
Closed-source models							
o1-mini	**94.23**	88.56	**92.84**	**96.52**	**61.21**	**61.81**	**99.02**
GPT-4o	84.86	**90.36**	86.29	86.84	32.76	33.44	97.96
Gemini1.5-pro	81.34	80.21	81.78	80.12	29.22	30.08	97.14
Qwen-max	79.23	82.98	76.82	75.16	21.33	22.68	94.04
DeepSeek-V3	93.53	85.19	75.17	95.67	19.09	19.78	96.53

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

dimension-specific performance gaps that are not captured by previous benchmarks.

Data contamination. In the arithmetic dimension, we ask LLMs to solve pure arithmetic problems that remove the background information and simplify the relationships between variables, in contrast to math word problems. Therefore, LLMs should theoretically achieve higher ACC@Ar scores than ACC@An. However, both Qwen2.5-72B and Gemini1.5-pro exhibit lower ACC@Ar scores than ACC@An, suggesting the possibility of data contamination.

Reasoning and reflection & refinement as the bottleneck. Most LLMs achieve over 80% SIM@Un in the context extraction task, indicating that they can grasp the relevant information of the question and interpret the problem statement. However, the scores for the reasoning and reflection & refinement dimensions are significantly lower than those for other dimensions. For example, Llama3.3-70B achieves 76.21% ACC@Reason and 36.38% ACC@RR, while reaching 95.92% in ACC@Ar. Although DeepSeek-V3 achieves similar scores to o1-mini in ACC@An and ACC@Ar, its performance in reasoning and R & R lags significantly behind that of o1-mini. Specifically, ACC@Reason for DeepSeek-V3 is 17.67% lower than o1-mini, and ACC@RR is 42.12% lower. These findings highlight that the primary bottleneck in mathematical problem-solving lies in the reasoning and reflection & refinement dimensions.

4.3 How Does Task Difficulty Impact Different Dimensions of SMART?

In this section, we investigate the factors influencing the performance of each dimension in the SMART framework. To do so, we generate several new, dimension-specific questions with varying difficulty levels and evaluate the performance of LLMs on this new testing data.

Difficulty setting. For the understanding dimension, we progressively introduce irrelevant sentences from other problems as noise into the seed question. The number of noise sentences controls the difficulty of the context extraction task. For the reasoning dimension, we define the complexity levels of questions based on the number of operations (e.g., $+, -, \times, \div, mod$) and classify the questions from the symbolic reasoning task into four levels. In the arithmetic dimension, we alter the number of digits rather than the magnitude of the numbers to generate different levels of complexity. In the reflection & refinement dimension, the difficulty is determined by the number of mistakes in the chain-of-thought solution. Further details of the difficulty settings are presented in the Appendix D.2.

Impact of task difficulty on LLMs. Fig. 4 illustrates the performance of five models across different difficulty settings. As the complexity level increases, the performance across all dimensions decreases sharply, indicating that our settings effectively control the difficulty of the testing data. The SIM@Un drops significantly at first and then generally levels off as the number of noise sentences increases. GPT-4o achieves only 40% in ACC@Reason and ACC@Ar when the number of reasoning steps exceeds six or the number of digits reaches nine. Therefore, mathematical problems with more reasoning steps and greater number of digits present a significant challenge for LLMs. The primary reason for the drop in ACC@RR as the number of mistakes in the CoT increases is

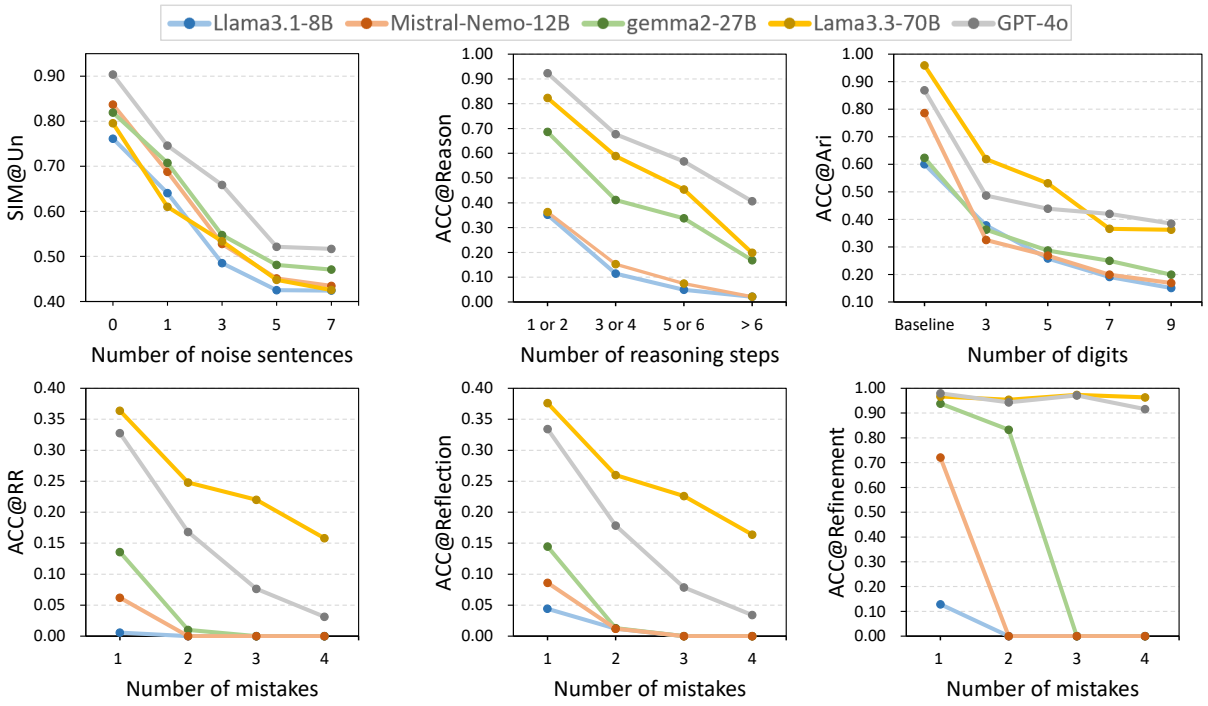

Figure 3: The performance of dimensions under different question difficulties.

that LLMs fail to detect all the mistakes. However, ACC@Refinement remains over 90% for GPT-4o even when the CoT contains four mistakes.

4.4 How do evaluation dimensions influence the performance drop of LLMs on questions with perturbations?

Many studies have demonstrated that LLMs experience significant performance drops when evaluated on question variations generated through perturbations (Li et al., 2024b; Zhu et al., 2023; Li et al., 2024a). However, perturbations that cause performance drops may affect multiple dimensions simultaneously. For example, adding operations may impact both the reasoning and arithmetic capabilities, making it difficult to pinpoint the primary factors driving the degradation.

In this section, we measure the performance change across the SMART dimensions when three perturbations (Li et al., 2024b) are applied to the seed questions. Table 3 shows that all dimensions experience noticeable PDR due to the perturbations. For the noise insertion perturbation, the understanding dimension is the most affected, with a 33.5% PDR for Llama3.3-70B. Both operation and numerical variation perturbations lead to significant PDR in the arithmetic dimension. Additionally, the reasoning and arithmetic dimensions of GPT-4o are more susceptible to perturbations than those of Llama3.3-70B, with GPT-4o experiencing higher PDR across these dimensions.

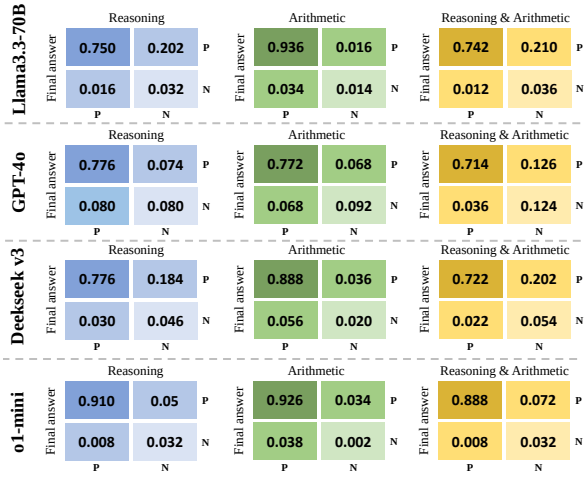

Figure 4: The confusion matrix of final answer and other dimensions. P means Positive, and N means Negative.

4.5 Is the Final Answer Accuracy Reliable for Measuring mathematical Capability?

We present the confusion matrix for final answer and other dimensions in Fig. 4, which categorizes math word problems into four classes based on the dimension results of SMART. True Positive (TP) indicates that the problem is solved correctly in both the final answer and its corresponding dimension-specific evaluation. False Negative (FN) refers to cases where the final answer is correct, but the model fails in the dimension-specific task.

Model	Perturbation	Answer		Understanding		Reasoning		Arithmetic	
		ACC@An↑	PDR↓	SIM@Un↑	PDR↓	ACC@Reason↑	PDR↓	ACC@Ar↑	PDR↓
Llama3.3-70B	Seed question	93.8	/	79.4	/	75.8	/	95.3	/
	+ Noise insertion	80.5	14.2	52.8	**33.5**	58.8	22.4	78.8	17.3
	+ Adding operation	79.8	14.9	68.4	13.9	59.8	21.1	71.6	**24.9**
	+ Numerical variation	37.2	**60.3**	75.4	5.0	49.5	30.7	60.4	**36.6**
GPT-4o	Seed question	84.8	/	90.3	/	86.2	/	86.8	/
	+ Noise insertion	80.7	4.8	65.3	**27.7**	63.8	26.0	77.8	10.4
	+ Adding operation	64.5	23.9	83.4	7.6	64.2	25.5	57.2	**34.1**
	+ Numerical variation	33.4	**60.6**	85.9	4.9	56.8	34.1	49.6	**42.9**

Table 3: The performance degradation of evaluation dimensions when three types of perturbations are added to the seed questions. PDR refers to the performance drop rate.

False Positive (FP) represents instances where the model arrives at the correct dimension evaluation answer but produces incorrect results in final answer. True Negative (TN) denotes scenarios where the model fails to solve both the original problem and its dimension-specific variants.

The FN values are nonzero across all confusion matrices, indicating that LLMs can sometimes arrive at correct answers through shortcuts or unknown mechanisms when their intermediate reasoning process is incorrect. Notably, the FN in the reasoning dimension for Llama3.3-70B (0.202) and DeepSeek-V3 (0.184) is significantly higher than their FN in the arithmetic dimension (0.016 and 0.036). This suggests that errors in reasoning contribute more to incorrect problem-solving than errors in arithmetic. Meanwhile, the FP scores are consistently low across all matrices, demonstrating the effectiveness of our dimension-specific evaluation design in SMART.

The TP score in the final answer-reasoning & rithmetic confusion matrix represents cases where LLMs correctly solve both the question and the intermediate process of reasoning and arithmetic tasks. We consider this score a measure of how well LLMs truly master math problem-solving ability. In Table 4. Although DeepSeek-V3 and Llama3.3-70B achieve similar score in ACC@An compared to o1-mini, their TP scores (72.2% and 74.23%) are significantly lower than o1-mini's (88.08%). o1-mini also exhibits the smallest performance drop rate of 5.76%, indicating the most reliable evaluation results with no significant overestimation.

To enhance the actual mathematical reasoning abilities of LLMs, we employ a reflection and refinement mechanisms via prompt engineering, with results presented as TP+RR scores in Table 4. Notably, o1-mini achieves a TP+RR score of 92.08%, which closely approaches its ACC@An of 94.23%.

Model	ACC@An↑	TP↑	PDR↓	TP+RR↑	PIR↑
Llama3.1-8B	63.16	41.42	34.42	42.58	2.80
gemma2-27B	68.62	43.12	37.16	46.58	8.02
Llama3.3-70B	94.08	74.23	21.10	80.75	**8.78**
DeepSeek-V3	93.53	72.20	22.81	76.11	5.42
GPT-4o	84.34	71.41	15.44	76.54	7.18
o1-mini	**94.23**	**88.08**	**5.76**	**92.08**	4.54

Table 4: The TP score of the final answer-reasoning & arithmetic confusion matrix. TP+RR refers to LLMs that incorporate reflection and refinement mechanisms. PDR denotes the performance drop rate between ACC@An and TP, while PIR indicates the performance increase rate from TP to TP+RR.

5 Conclusion

In this paper, we introduce SMART evaluation framework to systematically assess the mathematical problem-solving capabilities of LLMs. SMART deconstructs problem-solving into four key dimensions: understanding, reasoning, arithmetic, and reflection & refinement, enabling a fine-grained and interpretable evaluation. To ensure benchmark reliability and scalability, we propose a self-validating mechanism that iteratively verifies the correctness of generated test data. Through comprehensive experiments on 13 LLMs, we reveal that reasoning and reflection & refinement are the primary bottlenecks in mathematical problem-solving. Furthermore, our findings demonstrate that final answer accuracy alone is insufficient for assessing true mathematical competence, as models can arrive at correct answers through flawed reasoning process. We believe that SMART provides a more rigorous, interpretable, and scalable evaluation paradigm for advancing LLMs' mathematical reasoning abilities.

6 Limitations

First, SMART primarily focuses on grade-school-level mathematical problems. More complex mathematical problems, such as those requiring advanced symbolic manipulation or multi-step proofs, cannot be effectively analyzed within the SMART framework. Because their solution processes involve additional evaluation dimensions beyond the current four (understanding, reasoning, arithmetic, and reflection & refinement). Extending SMART to handle higher-level mathematical reasoning remains an open challenge.

Second, Z3 and SMT-LIB are effective tools for solving problems involving linear and integer equations, and certain types of nonlinear constraints. However, due to the limitations of their problem-solving scope, Z3 and SMT-LIB are not well-suited for addressing highly complex nonlinear problems, and certain NP-complete combinatorial problems. In the future, we plan to use Lean (Moura and Ullrich, 2021) to formalize and prove complex mathematical theorems, particularly in areas that involve higher-level logic and more intricate proof structures, which are beyond the capabilities of SMT solvers.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. 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, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 model card.

Clark Barrett, Aaron Stump, Cesare Tinelli, et al. 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.

Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. 2013. The mathsat5 smt solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 93–107. Springer.

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*.

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.

Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. 2023. Nphardeval: Dynamic benchmark on reasoning ability of large language models via complexity classes. *arXiv preprint arXiv:2312.14890*.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 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. *NeurIPS*.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

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.

Eldar Kurtic, Amir Moeini, and Dan Alistarh. 2024. Mathador-lm: A dynamic benchmark for mathematical reasoning on large language models. *arXiv preprint arXiv:2406.12572*.

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. *arXiv preprint arXiv:2405.19740*.

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. *arXiv preprint arXiv:2402.19255*.

Zenan Li, Zhi Zhou, Yuan Yao, Yu-Feng Li, Chun Cao, Fan Yang, Xian Zhang, and Xiaoxing Ma. 2024c. Neuro-symbolic data generation for math reasoning. *arXiv preprint arXiv:2412.04857*.

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*.

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

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

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.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

MistralAITeam. 2024a. Mistral-nemo-instruct-2407. https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407.

MistralAITeam. 2024b. 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.

Yonatan Oren, Nicole Meister, Niladri Chatterji, Faisal Ladhak, and Tatsunori B Hashimoto. 2023. Proving test set contamination in black box language models. *arXiv preprint arXiv:2310.17623*.

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

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.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

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*.

Yiming Wang, Zhuosheng Zhang, Pei Zhang, Baosong Yang, and Rui Wang. 2023. Meta-reasoning: Semantics-symbol deconstruction for large language models. *arXiv preprint arXiv:2306.17820*.

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.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 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. Dyval 2: Dynamic evaluation of large language models by meta probing agents. *arXiv preprint arXiv:2402.14865*.

A Polya's Problem-solving Theory

Polya first propose the four step problem solving method in the book 'How to solve it' (Pólya and Conway, 1957) to systematically answer questions. According to his theory, it breaks down the process of problem-solving into four steps (Fig. 1): 1) understanding the problem. 2) make a plan. 3) carry out the plan. 4) look back and review.

Inspired by Polya's problem-solving theory, we evaluate the mathematical capability in four key dimensions:

- The understanding dimension assesses the model's ability to comprehend the problem accurately and clearly.
- The reasoning dimension evaluates the model's capacity for symbolic reasoning and logical deduction.
- The arithmetic dimension measures the model's proficiency in performing numerical computations.
- The reflection & refinement dimension examines the model's ability to identify errors in the solution process and make corrections to improve the final answer.

B SMART Benchmark

B.1 Seed Question Dataset of SMART

We construct the seed question dataset of SMART benchmark containing 6862 testing data from five widely used grade-school-level math word problem dataset from GSM8k (Cobbe et al., 2021), SVAMP (Patel et al., 2021), ASDiV (Miao et al., 2020), AQuA (Ling et al., 2017) and MAWPS (Koncel-Kedziorski et al., 2016). The seed questions and their dimension-specific variations construct the SMART benchmark which contains 34310 testing questions.

The concrete data source distribution is illustrated in the Tab. 5. It is important to note that we convert the question type of AQuA testing dataset, which is a multiple-choice question, into an open-ended question type to ensure consistency with other datasets. We use the content of the correct option from the multiple-choice question as the ground truth for the question in the AQuA testing dataset. In addition, problems involving the greatest common divisor (GCD), least common multiple (LCM), or finding the maximum or minimum values cannot be expressed or automatically solved using SMT-LIB. Therefore, such questions will be excluded from the seed question dataset.

B.2 Dimension Specific Testing Dataset of SMART

We generate different question variants and its corresponding ground truth from each seed question to evaluate four problem-solving processes of LLMs.

B.2.1 Understanding

We use the GPT-4o the extract the context information as the ground truth. The prompt for GPT-4o to extract context as ground truth are shown in Fig. 7. We use the GPT-4o to check the quality of the extracted context. The self-validating mechanism as illustrated in the sec. 3.2. The prompt for verify the extracted context are shown in Fig. 10.

B.2.2 Arithmetic

We measure the arithmetic ability of LLMs in terms of its performance on solving pure numeric calculation problem, which has the same reasoning logic and final answer to the seed question. Directly converting the seed question to the arithmetic problem is challenging for LLMs because it requires simplifying complex natural language into structured mathematical operations while maintaining the logical relationships between variables. This transformation is not straightforward, as the model needs to accurately interpret the problem's intent, handle ambiguous phrasing, and correctly map it to arithmetic operations. Thus, we first generate the SMT-LIB of the seed question to simplify the reasoning logic among variables and then convert the SMT-LIB to the arithmetic problem via GPT-4o. The prompt for that process is shown in Fig. 8 and Fig. 9.

Dataset	Number
GSM8k-test	1325
SVAMP	997
ASDiV	1987
AQuA-test	187
MAWPS	2366
All	6862

Table 5: Seed question dataset source distribution

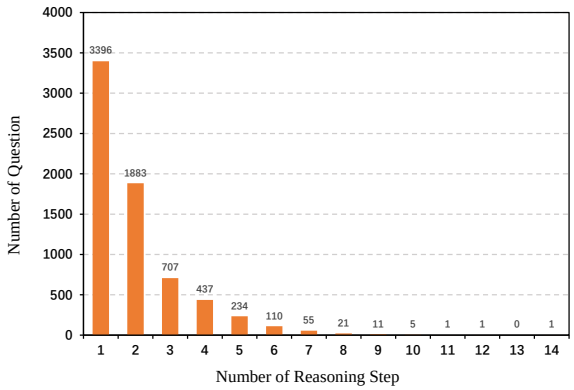

Figure 5: The number of reasoning step statistics of seed question dataset.

We verify the effectiveness of the generated arithmetic problem through the self-validating mechanism as illustrated in the sec. 3.2. The prompt for verify the generated arithmetic problem are shown in Fig. 8.

B.2.3 Reflection & Refinement

We randomly select a sentence from the chain-of-thought (CoT) and modify the number within that sentence to generate wrong CoT. The ID of the chosen sentence serves as the ground truth for the mistake detection task.

Fig. 6 presents a data sample in the SMART benchmark, which contains the seed question, the extracted context, the symbolic expression, the arithmetic question, the CoT and the final answer.

C Details for SMART Framework Evaluation

C.1 Understanding

We propose a context extraction task to measure the capability of LLMs in understanding math problems. The input for this task consists of seed questions, and the model is asked to extract the context of the question based on the prompt shown in Fig. 7. The evaluation metric for this task is the sentence similarity between the extracted context and the ground truth context.

C.2 Reasoning

"For the reasoning dimension, we introduce a symbolic formalization task to evaluate the symbolic reasoning capability of LLMs. The input 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. The evaluation metric for this task is the accuracy of the SMT-LIB formula results.

C.3 Arithmetic

For the arithmetic dimension, we introduce a numeric calculation task to assess the arithmetic ability of LLMs. The input for this task is an arithmetic problem, and LLMs are asked to solve it using the prompt shown in Fig. 11. The arithmetic problem is designed to have the same answer and reasoning logic as its corresponding seed question.

C.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. This task consists of three steps. The first step involves detecting errors in the CoT, given the question and CoT, with the answer being the position index of the erroneous sentence, as specified in the prompt shown in Fig. 12. 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 corrected CoT with given seed question and CoT with prompt shown in Fig.13. The final step is to refine the answer based on the corrected CoT and the seed question, using the prompt in Fig. 14. 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.

D Experiments

D.1 Experiment Setting

D.1.1 Environment Setting.

All experiments were conducted on a Linux server equipped with two NVIDIA H800 GPUs (80GB).

The Seed Question:
John invited 20 people to a birthday party. Each guest will eat 2 hot dogs. He already has 4 hot dogs left over from a previous party. If a pack of hot dogs contains 6 hot dogs and costs $2, how much does he need to spend on hot dogs?

The Extracted Context:
John is hosting a birthday party and needs to ensure he has enough hot dogs for his guests. He is calculating the cost based on the number of guests, the hot dogs he already has, and the price per pack

The Symbolic Expression:

```
(set-logic QF_NRA)
(declare-fun a () Real)
(declare-fun b () Real)
(declare-fun c () Real)
(declare-fun d () Real)
(declare-fun e () Real)
(declare-fun f () Real)
(declare-fun g () Real)
(declare-fun h () Real)
(declare-fun i () Real)
(assert (= a 20))
(assert (= b 2))
(assert (= c 4))
(assert (= d 6))
(assert (= e 2))
(assert (= f (* a b)))
(assert (= g (- f c)))
(assert (= h (/ g d)))
(assert (= i (* h e)))
(check-sat)
(get-value (i))
```

The Arithmetic Question:
What is the value of i, if a is equal to 20, b is equal to 2, c is equal to 4, d is equal to 6, e is equal to 2, f is equal to the product of a and b, g is equal to f minus c, h is equal to g divided by d, and i is equal to h multiplied by e?

The CoT:
He needs 2*20=<<2*20=40>>40 hot dogs. So he needs another 40-4=<<40-4=36>>36 hot dogs. So he needs to buy 36/6=<<36/6=6>>6 packs of hotdogs. That means he needs to spend 6*2=$<<6*2=12>>12. #### 12.

The Final Answer:
12.0

Figure 6: A data sample in the SMART benchmark.

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.

D.1.2 Prompt Setting.

For our experiments, we leverage prompt templates to generate evaluation questions, with examples provided in Figures 7 through 14. Once the model outputs are obtained, we convert them into JSON format for subsequent analysis, using regular expressions to ensure the transformation process is robust.

D.2 Examples for question with different difficulty setting

Fig.15 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.16 shows examples of question with different reasoning steps, which indicates different reasoning difficulties.

Fig.17 presents arithmetic questions with numbers in different digits. Numbers with more digits means more difficult for arithmetic evaluation task.

Fig.18 presents CoT with different number of error steps. CoT with more mistakes are more

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

User Prompt:
You are a highly skilled mathematician, NLP expert, and contextual analyzer. Your task is to analyze a natural language math problem and extracting the background information.
The context extraction task is to extract non-mathematical information that provides context for the problem. Avoid including any numbers, variables, or explicit mathematical details. Focus only on descriptive and contextual information (e.g., the scenario, characters, or events).

The Given Question:
[Seed question]

Now, analyze the next math word problem. Extract the background information, strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Extracted Context:
[Question context]

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

difficult for reflection evaluation task.

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.

The Given Question:
[Seed question]

Now, analyze the next math word 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 Symbolic Expression:
[SMT-LIB]

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

Perturbation	Answer		Understanding		Reasoning		Arithmetic	
	ACC@An↑	PDR↓	ACC@Un↑	PDR↓	ACC@Reason↑	PDR↓	ACC@Ar↑	PDR↓
Seed question	68.3	/	81.4	/	61.2	/	62.5	/
+ Noise insertion	54.6	20.1	56.1	**31.1**	47.3	13.9	60.0	4
+ Adding operation	23.7	65.3	74.1	9.0	41.8	19.3	40.2	**35.7**
+ Numerical variation	21.6	**68.4**	78.9	3.1	38.6	22.5	38.4	**38.6**

Table 6: The performance degradation of evaluation dimensions for gemma2-27B when three types of perturbations are added to the seed questions. PDR refers to the performance drop rate.

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

User Prompt:
You are a highly skilled mathematician, NLP expert. You will be given a math problem, and your task is to convert the original SMT-LIB formula into a pure arithmetic problem in natural language, 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 word problem is same to the SMT-LIB' answer. Do not mention answer in the new generated question.

The Given Symbolic Expression :
[SMT-LIB]

The Given Answer:
[Final answer]

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

The Arithmetic Problem:
[Arithmetic problem]

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

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

User Prompt:
You are a highly skilled mathematician, NLP expert. You will be given a SMT-LIB expression, and a question context. Your task is to create a math word problem based on the provided SMT-LIB expression and background information. Follow these guidelines:

1. Use the SMT-LIB expression: The expression defines variables, equations, and constraints. Extract the logic and numerical values from the SMT-LIB expression.
2. Incorporate the background information: Use the provided context to create a realistic and meaningful story for the word problem. Align the narrative with the relationships and values in the SMT-LIB expression.
3. Ensure correctness: The math word problem must logically match the SMT-LIB expression.
4. Structure the problem: Include all necessary details for solving the problem in a clear and concise manner. Avoid introducing additional irrelevant information.

The Given Symbolic Expression :
[SMT-LIB]

The Given Context:
[Context]

Now, analyze the next SMT-LIB expression. Generate a math word problem based on the SMT-LIB expression and background information. Strictly following the steps and formatting provided. Be precise, logical, and concise in your responses.

The Generated Math Word Problem:
[Math word problem]

Figure 10: The prompt for LLMs to generated math word problem based on the SMT-LIB expression and the extracted context.

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

User Prompt:
You are a highly skilled mathematician, NLP expert. You will be given a math problem, and your task is to calculate the answer of the problem. The answer should not include any units or formulas. If there are multiple answers, separate them with a comma and a space. Additionally, ensure that your answer is rounded to five decimal places.

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 Question:
[Answer]

Figure 11: The prompt for LLMs to solve the Mathematic problem.

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

User Prompt:
You are a highly skilled mathematician, NLP expert. You will be given a math problem, ant its corresponding solution steps (Chain-of-Thought, CoT). If there is an error in the solution steps, identify which sentence contains the error. The error sentence number should be based on the order of sentences in the CoT, starting from 0. If no error is found, return False.

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 Position of Error:
[Sentence ID]

Figure 12: The prompt for LLMs to detect mistakes in the CoT.

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

User Prompt:
You are a highly skilled mathematician, NLP expert. 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.

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 Corrected of CoT:
[C-CoT]

Figure 13: The prompt for LLMs to correct the mistaks in the CoT.

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

User Prompt:
You are a highly skilled mathematician, NLP expert. I will provide you with a math word problem, its revised solution steps (CoT). Your task is to provide new answer of the problem. The answer should not include any units or formulas. If there are multiple answers, separate them with a comma and a space. Additionally, ensure that your answer is rounded to five decimal places.

The Given Question:
[Seed question]

The Given Corrected CoT:
[C-CoT]

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

The Answer of Question:
[Answer]

Figure 14: The prompt for LLMs to refine the anwer of the seed question based on the correct CoT.

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 15: Example of questions with 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 16: 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 17: Example of arithmetic questions with number in different digits.

The Seed Question:
Peter has 54 tickets and 4 pencils. If he shares the tickets among 9 friends, how many tickets does each friend get?", "background": "Peter has a collection of tickets and pencils. He decides to share his tickets with his friends, specifically dividing them among a group. The scenario revolves around the distribution of these tickets and how many each friend receives.

The CoT with One Mistake:
To determine how many tickets each of Peter's 12 friends receives, we need to divide the total number of tickets by the number of friends. First, we identify the total number of tickets Peter has, which is 54. Next, we note that Peter is sharing these tickets among 9 friends. Now, we perform the division to find out how many tickets each friend gets: 54 tickets ÷ 9 friends = 6 tickets per friend. Thus, each friend receives 6 tickets. #### 6.0

The CoT with Two Mistakes:
To determine how many tickets each of Peter's 12 friends receives, we need to divide the total number of tickets by the number of friends. First, we identify the total number of tickets Peter has, which is 52. Next, we note that Peter is sharing these tickets among 9 friends. Now, we perform the division to find out how many tickets each friend gets: 54 tickets ÷ 9 friends = 6 tickets per friend. Thus, each friend receives 6 tickets. #### 6.0

The CoT with Three Mistakes:
To determine how many tickets each of Peter's 12 friends receives, we need to divide the total number of tickets by the number of friends. First, we identify the total number of tickets Peter has, which is 52. Next, we note that Peter is sharing these tickets among 9 friends. Now, we perform the division to find out how many tickets each friend gets: 54 tickets ÷ 9 friends = 8 tickets per friend. Thus, each friend receives 6 tickets. #### 6.0

The CoT with Four Mistakes:
To determine how many tickets each of Peter's 12 friends receives, we need to divide the total number of tickets by the number of friends. First, we identify the total number of tickets Peter has, which is 52. Next, we note that Peter is sharing these tickets among 9 friends. Now, we perform the division to find out how many tickets each friend gets: 54 tickets ÷ 9 friends = 8 tickets per friend. Thus, each friend receives 6 tickets. #### 12.0

Figure 18: Example of CoT with different number of error steps.