

MATHDEBUGGER: A Benchmark for Type-Aware Error Detection in Synthetic Math Data

Hao Liang^{1,6*}, Meiyl Qiang^{1*}, Yuying Li^{2*}, Zefeng He³, Xiaochen Ma⁴,
Ruichuan An¹, Yongzhen Guo⁵, Zhengzhou Zhu¹, Bin Cui¹, Wentao Zhang^{1,6}

¹Peking University ²Tsinghua University ³Nanjing University ⁴HKUST

⁵Ant Group ⁶Zhongguancun Academy

wentao.zhang@pku.edu.cn

Abstract

Synthetic math data is widely used to scale LLM training, but its correctness is not guaranteed. We introduce MATHDEBUGGER, the first text-only, type-aware benchmark for jointly detecting *and* classifying errors in synthetic mathematical questions and answers. MATHDEBUGGER contains 2,000 correct questions, 2,000 erroneous questions (balanced across four fine-grained error types), and 2,000 annotated answers (610 erroneous, across three error types), each annotated with both a correctness label and an error-type label, with per-error-type Fleiss’ κ ranging from 0.69 to 0.88. On 14 SOTA LLMs (6 closed-source, 8 open-source, 1.5B–72B) and 3 process reward models, even advanced systems such as GPT-o3 and DeepSeek-R1 remain far from saturating the benchmark, particularly on fine-grained error-type detection; we further surface a *solving-vs-verifying* gap that causes math-tuned and long-chain-reasoning models to trail their general-purpose siblings. Error-type annotations also provide actionable supervision for correction, with gains that are significant under paired bootstrap ($p < 0.01$). Code and data: <https://anonymous.4open.science/r/MathDebugger-7C4A/README.md>.

1 Introduction

LLMs have demonstrated exceptional performance across a wide range of tasks (OpenAI, 2023; Touvron et al., 2023). It has been established that data quality plays a crucial role in the success of LLMs (Yang et al., 2024; Bai et al., 2024; Abdin et al., 2024). Recent efforts have explored data error detection techniques to improve LLM training efficiency and effectiveness (Yan et al., 2024; Ma et al., 2024; He et al., 2025).

Among the various types of data for LLMs, mathematical data is especially critical because it exercises a model’s reasoning capability (Yang et al.,

2024; Guo et al., 2025). Collecting it at scale, however, is hard: high-quality Math QA pairs are scarce and typically require generation (Zhou et al., 2024a; Toshniwal et al., 2024), ensuring correctness of synthetic data is non-trivial (Toshniwal et al., 2024), and evaluating correctness remains complex (Song et al., 2025; Zheng et al., 2024; Zhou et al., 2024b). Output/process reward models (Zhang et al., 2024c; o1 Team, 2024; Wang et al., 2024; Song et al., 2025) and specialized step-verification benchmarks (Song et al., 2025; Zheng et al., 2024; Zhou et al., 2024b) have been proposed, but they focus on scoring reasoning *steps*, not on auditing the training data itself. MATHDEBUGGER targets the complementary task of *error detection in synthetic mathematical training data*, which we operationalise through two objectives:

Questions and Answers Error Detection. ErrorRadar (Yan et al., 2024) primarily targets multimodal data, while UMP (Ma et al., 2024) focuses on reasoning under unreasonable questions. DeltaBench (He et al., 2025) examines errors in long-chain CoT reasoning, such as step-level mistakes that may still yield correct answers. In contrast, none of these works address the task of detecting correctness in textual math question–answer pairs, which is the focus of our benchmark.

Diverse Error Types. To rigorously assess whether LLMs truly understand the correctness of QA, it is essential to include diverse error types. A benchmark that categorizes errors can facilitate fine-grained evaluation and guide targeted improvements in synthetic data quality.

To address these challenges, we introduce MATHDEBUGGER, a benchmark designed to evaluate LLMs’ ability to detect errors and identify error types in synthetic mathematical data. To construct the benchmark, we design 10 novel question-rewriting prompts for generating erroneous data and 16 rewriting prompts (11 more than those in MuggleMath (Li et al., 2024)) to enhance the di-

*Equal Contribution.

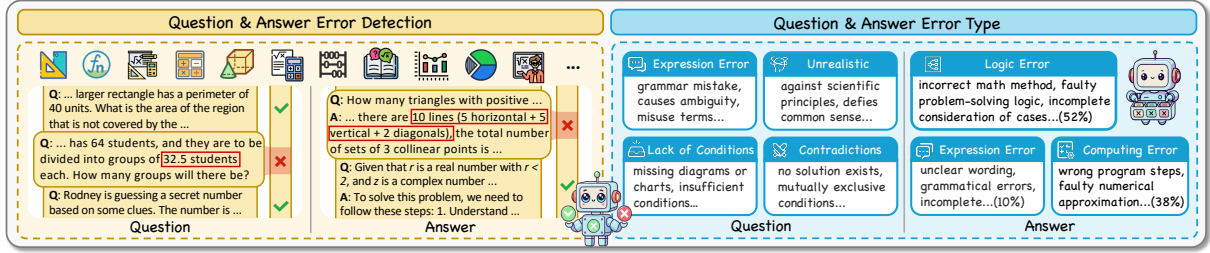


Figure 1: Overview of MATHDEBUGGER benchmark. MATHDEBUGGER proposed challenges in detecting errors in mathematical Questions and Answers, as well as in identifying error types within these mathematical problems.

versity of synthesized questions. MATHDEBUGGER consists of 2,000 correct questions, 2,000 erroneous questions, and 2,000 correct or incorrect answers, each annotated with both correctness and fine-grained error types to enable comprehensive evaluation. We further conduct large-scale experiments on MATHDEBUGGER, covering six widely used closed-source LLMs, eight open-source LLMs (1.5B–72B), and three state-of-the-art PRMs. Figure 1 illustrates the tasks of error detection and error type identification in the benchmark.

The core contributions of this work are:

- **MATHDEBUGGER Benchmark.** We introduce MATHDEBUGGER, the first text-only, type-aware benchmark covering both question- and answer-side errors with a fine-grained typed taxonomy.
- **Diverse and Controlled Data Synthesis.** We propose 10 novel question-rewriting prompts for generating erroneous data and 16 rewriting prompts (substantially extending prior work) to enrich data diversity. All synthesized questions are carefully validated by human annotators, with per-error-type Fleiss’ κ ranging from 0.69 to 0.88.
- **Comprehensive Evaluation.** We benchmark 6 closed-source LLMs, 8 open-source LLMs (1.5B–72B), and 3 PRMs. Our results show that even strong current systems remain far from saturating MATHDEBUGGER, highlighting its difficulty and diagnostic value. We further provide a random baseline, multi-seed variance, bootstrap 95% CIs, prompt-format ablations, and a 13-gram contamination audit.
- **Error-Type-Aware Correction.** We empirically demonstrate that fine-grained error-type annotations not only support diagnostic eval-

uation, but also provide actionable supervision for correcting erroneous mathematical questions and answers, consistently improving correction accuracy across models and error categories.

2 Related Work

2.1 Mathematical Benchmarks

Recent research has achieved significant advancements in mathematical reasoning benchmarks. GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) are the most mathematical classical benchmarks, focusing on computational and competition-level questions, respectively. Subsequent studies, such as SuperCLUE-Math (Xu et al., 2024), MathBench (Liu et al., 2024), Omni-MATH (Gao et al., 2024b), and FrontierMath (Glazer et al., 2024), have further enriched and expanded the evaluation dimension of LLMs’ mathematical capabilities by making breakthroughs in terms of multilingual support, scopes of mathematical concepts, difficulty levels, and data contamination. PRMBench (Song et al., 2025) and ProcessBench (Zheng et al., 2024) cater to the demand for step-by-step reasoning assessment, evaluating the stepwise reasoning ability of LLMs. Meanwhile, MathTrap (Zhao et al., 2024) introduces a novel focus on detecting logical traps in reasoning chains. However, there remains a notable gap in benchmarks assessing LLMs’ ability to identify low-quality or incorrect mathematical data, a critical capability for data filtering and cleaning. This work aims to address this unmet need.

Table 1 summarizes how MATHDEBUGGER differs from the closest existing benchmarks. ProcessBench and PRMBench operate at the *reasoning-step* level and assume the underlying question is valid; they cannot surface errors in the problem statement itself. ErrorRadar targets multimodal math problems and provides typed errors only on

Benchmark	Granularity	Q-err	A-err	Typed	Size
ProcessBench (Zheng et al., 2024)	step	✗	✓	✗	3,400
PRMBench (Song et al., 2025)	step	✗	✓	partial	6,216
ErrorRadar (Yan et al., 2024)	solution (multimodal)	✗	✓	✓	2,500
DeltaBench (He et al., 2025)	CoT step	✗	✓	✗	1,236
UMP (Ma et al., 2024)	question (unreasonable)	✓	✗	✗	582
MATH-Minos (Gao et al., 2024a)	step	✗	✓	✗	–
MATHDEBUGGER (ours)	question + solution (text)	✓	✓	✓	6,000

Table 1: Positioning of MATHDEBUGGER relative to closely related error / step-verification benchmarks. **Q-err**: errors in the question statement; **A-err**: errors in the answer/solution; **Typed**: fine-grained error-type labels.

the answer side. DeltaBench focuses on step-level flaws in long CoT traces where the *final answer may still be correct*. UMP evaluates whether models refuse unreasonable questions but does not provide typed error labels and is limited to 582 items. MATH-Minos introduces step-wise natural-language feedback but does not provide a fine-grained error taxonomy. In contrast, MATHDEBUGGER is the first text-only benchmark that (i) covers *both* question-side and answer-side errors, (ii) provides a fine-grained four-class (question) and three-class (answer) taxonomy, and (iii) reports per-error-type inter-annotator agreement.

2.2 Mathematical Data Synthesis

The demand for high-quality data in the field of LLMs has spurred the flourishing development of the data synthesis domain. Existing data synthesis approaches can be broadly categorized into two types. The first approach employs **LLM-based distillation methods**, which utilize prompt engineering to facilitate LLMs in performing data synthesis tasks such as question rewriting. Representative works in this paradigm include MetaMath (Yu et al., 2023), KPDDS (Huang et al., 2024), and JiuZhang3.0 (Huang et al., 2024). The second approach leverages **Monte Carlo Tree Search (MCTS)-based synthesis**, where Process Reward Models guide the path construction of MCTS to optimize synthetic data generation. Notable implementations of this method encompass OpenAI’s o1 (OpenAI, 2023), ReST-MCTS* (Zhang et al., 2024a), LLaMA-Berry (Zhang et al., 2024b), and Mulberry (Yao et al., 2024). In this work, we primarily adopt the first approach for data synthesis. The synthesized data undergoes multiple rounds of manual annotation to construct the MATHDEBUGGER benchmark, ensuring high-quality evaluation standards for mathematical data processing.

3 MATHDEBUGGER Benchmark

3.1 Wrong Question Dataset Construction

We use GSM8K and MATH as seed data, generating math questions with various error types using Qwen2.5-72B-Instruct, LLaMA-3.1-70B-Instruct, and GPT-4o. GSM8K-based outputs are classified as **SIMPLE** and MATH-based outputs as **CHALLENGING**. Mathematical experts analyse classic LLM errors and design prompts that guide the generators to produce target error types.

Many synthetic questions exhibit multiple error types; to ensure diversity, we first design ten specific erroneous conditions (Appendix Figure 11), then consolidate them into four primary error types: (1) **Expression Error**, covering linguistic issues such as grammatical errors, ambiguous references, or redundant content; (2) **Lack of Conditions**, where information required for solution is missing; (3) **Contradictions**, where stated conditions are mutually inconsistent; (4) **Unrealistic**, where solutions violate common sense or real-world constraints. Full definitions and the *earliest-invalidating-error* adjudication rule are in Appendix A.

Dataset	Error	Correct	Total
Simple	163	637	800
Challenging	447	753	1200
Both	610	1390	2000

Table 2: The correctness statistics of different difficulty levels in the MathDebugger Answer dataset.

Annotation. Questions are labelled by 30 recruited STEM graduate annotators and 10 reviewers from top-tier universities, who pass a multi-stage qualification screening. Annotators (i) solve the question, (ii) flag correctness, and (iii) for incorrect items, select the specific error type and pro-

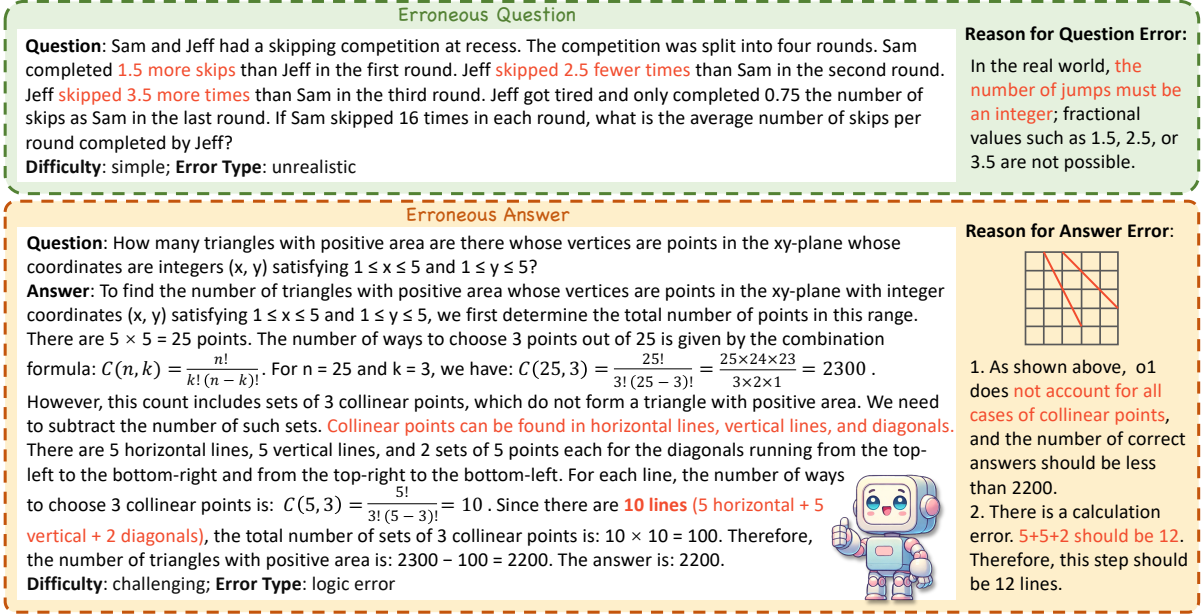


Figure 2: Two erroneous examples from MATHDEBUGGER: a SIMPLE question with an *unrealistic* error, and a CHALLENGING question paired with an answer exhibiting a *logic* error.

vide a written justification. Every label undergoes dual independent review; an annotator’s labels are discarded entirely when consensus accuracy falls below 90%. The two-month effort cost $\sim$ \$50k. Further details of the annotation protocol are in Appendix B.

Target	Difficulty	#Inst.	Fleiss’ κ
Question	SIMPLE	1600	0.88
	CHALLENGING	2400	0.72
Answer	SIMPLE	800	0.87
	CHALLENGING	1200	0.76

Table 3: Overall inter-annotator agreement (Fleiss’ κ) at the *correctness* level for MATHDEBUGGER.

Error Type	SIMPLE	CHALLENGING
Lack of Conditions	0.88	0.75
Unrealistic	0.84	0.71
Expression Error	0.82	0.73
Contradictions	0.80	0.69

Table 4: Per-error-type inter-annotator agreement (Fleiss’ κ) on the question dataset at both difficulty levels. Complements the correctness-level κ in Table 3.

3.2 Wrong Answer Dataset Construction

To diversify the answer dataset we (i) synthesise high-variation questions with 16 rewriting prompts (Appendix Figure 12). Since all questions in MATHDEBUGGER come from new synthesis, this also mitigates contamination. We then

(ii) regenerate answers using models of varying capability: Qwen2.5-72B-Instruct (high-capacity), LLaMA-3.1-8B-Instruct (medium), and Qwen2.5-Math-1.5B-Instruct (lightweight), with both CoT and PoT prompting.

Answers are then annotated by the same pool of qualified annotators following a three-step protocol (check question, check answer, label error type and justify). Answer errors are categorised into three types: **Expression**, **Logic**, and **Computational** errors (definitions in Appendix A). After decontamination we retain only items whose question is correct and whose answer exhibits at most a single, typical error type. Figure 2 shows representative erroneous instances.

3.3 Annotation Quality and Inter-Annotator Agreement

All questions and answers were independently annotated by two trained annotators, followed by expert review. We report Fleiss’ κ at both the correctness level and the fine-grained error-type level.

Tables 3 and 4 report the overall and per-error-type agreement. Binary correctness annotations reach almost perfect agreement on SIMPLE ($\kappa \geq 0.87$) and substantial agreement on CHALLENGING ($\kappa \geq 0.72$). Per-error-type κ ranges from 0.88 (*lack of conditions* on SIMPLE) to 0.69 (*contradictions* on CHALLENGING). We retain the lowest κ as published; it reflects a *genuine* semantic boundary ambiguity between *contradictions* and

Models	Error Detection				Error Type Detection			
	SIMPLE		CHALLENGING		SIMPLE		CHALLENGING	
	Acc	F1-Score	Acc	F1-Score	Acc	Macro-F1	Acc	Macro-F1
Random Baseline	50.00	50.00	50.00	50.00	25.00	25.00	25.00	25.00
Closed-Source Large Language Models								
GPT-4o	72.44	76.38	72.50	75.26	79.13	79.14	74.00	73.90
GPT-o1	73.19	75.39	76.38	77.91	76.38	76.14	72.83	72.38
GPT-o3	72.62	74.14	74.08	75.28	75.50	75.32	71.67	71.34
GPT-o4-mini	74.81	76.88	76.17	77.92	75.88	75.64	72.58	72.43
Claude-3-5-sonnet	76.31	78.33	75.91	78.25	74.88	74.90	70.67	70.85
Gemini-2.5-Flash	71.81	70.77	62.54	55.87	64.50	63.05	62.33	61.12
Open-Source Large Language Models								
LLaMA-3.1-8B-Instruct	59.62	46.96	64.88	58.82	56.00	55.08	56.67	55.81
LLaMA-3.3-70B-Instruct	71.37	76.42	69.71	74.01	75.38	75.24	77.83	77.90
Qwen2.5-1.5B-Instruct	54.44	56.11	57.83	57.90	35.88	35.65	39.58	39.47
Qwen2.5-7B-Instruct	67.75	74.02	71.33	76.13	55.13	55.62	58.75	59.17
Qwen2.5-72B-Instruct	72.31	77.22	73.83	78.15	77.62	77.58	73.25	73.39
Qwen2.5-Math-72B-Instruct	68.00	74.01	68.87	73.78	67.37	67.35	65.58	65.36
QwQ-32B-Preview	57.81	40.53	55.50	34.88	67.00	66.85	64.00	63.78
DeepSeek-R1	71.25	72.46	74.50	75.94	70.37	70.21	70.75	70.50

Table 5: Error Detection and Error Type Detection in Questions: the best model for each task is highlighted in **bold**. All reported numbers are the mean of 3 independent runs (std dev ≤ 0.5 on GPT-4o CHALLENGING: 72.50 / 72.83 / 72.21); Gemini-2.5-Flash is a single run due to API-side rate constraints. Bootstrap 95% CIs (1,000 resamples) are in Appendix D.

lack of conditions that we did not smooth over, and all residual disagreements were adjudicated by domain experts rather than re-labelled to inflate agreement.

3.4 Statistics of MATHDEBUGGER

The question split comprises 2,000 correct and 2,000 incorrect questions; the incorrect half is balanced across the four error types (500 per class), split into 200 SIMPLE (GSM8K-seeded) and 300 CHALLENGING (MATH-seeded) per class. The answer split pairs 2,000 correct questions with their answers (40% SIMPLE, 60% CHALLENGING; 30.5% of answers are incorrect). Error counts over the 610 erroneous answers are *logic*=317 (52%), *computing*=232 (38%), *expression*=61 (10%), matching the distributions reported by MATH-Minos (Gao et al., 2024a) and Error-Radar (Yan et al., 2024). Detailed composition charts and the full breakdown by difficulty appear in Appendix Figures 4–5 and Table 2.

4 Experiments

4.1 Experimental Setting

We evaluate four sub-tasks: question error detection, answer error detection, question error-type

detection, and answer error-type detection (abbreviated Q-Detect, A-Detect, Q-Type, A-Type in the following tables). The first two are binary; the last two are multi-class (4-way for questions, 3-way for answers). We report Accuracy and F1 for binary tasks and Accuracy and Macro-F1 for multi-class tasks. The prompt template is in Figure 10.

4.2 Models

To comprehensively evaluate model performance on the MATHDEBUGGER benchmark, we consider three categories of models. **Closed-source LLMs:** GPT-4o (gpt-4o-2024-11-20), o1, o3, o4-mini (Achiam et al., 2023; OpenAI, 2023), Claude-3-5-Sonnet (claude-3-5-sonnet-20241022), and Gemini-2.5-Flash (Team et al., 2023). **Open-source LLMs:** LLaMA-3.1/3.3 (8B–70B) (Dubey et al., 2024), Qwen2.5-Instruct and Qwen2.5-Math series (1.5B–72B) (Yang et al., 2024), QwQ-32B-Preview (Team, 2024), and DeepSeek-R1 (Guo et al., 2025). **PRMs:** Math-Shepherd-Mistral-7B-PRM (Wang et al., 2024), Skywork-PRM-7B (o1 Team, 2024), and Qwen2.5-Math-PRM-7B (Zhang et al., 2025). PRMs target intermediate reasoning steps and are thus applied only to answer error detection; following Cui et al. (2025), we treat the full reasoning trace as one step, score overall

Models	Error Detection				Error Type Detection			
	SIMPLE		CHALLENGING		SIMPLE		CHALLENGING	
	Acc	F1-Score	Acc	F1-Score	Acc	Macro-F1	Acc	Macro-F1
Random Baseline	50.00	50.00	50.00	50.00	33.33	33.33	33.33	33.33
Closed-Source Large Language Models								
GPT-4o	80.75	88.02	77.25	82.97	59.51	46.93	57.49	42.11
GPT-o1	76.62	84.66	78.58	82.24	62.58	48.17	60.40	41.52
GPT-o3	72.00	80.72	74.50	77.99	64.42	55.05	63.31	50.02
GPT-o4-mini	78.12	85.67	78.08	81.67	61.96	52.80	62.42	52.77
Claude-3-5-sonnet	73.12	81.98	73.83	78.84	60.12	48.17	56.38	38.21
Gemini-2.5-Flash	76.25	85.13	68.25	75.18	61.96	49.48	60.85	45.46
Open-Source Large Language Models								
LLaMA-3.1-8B-Instruct	63.25	73.80	62.83	67.25	45.40	40.39	49.22	36.40
LLaMA-3.3-70B-Instruct	79.25	86.99	76.17	82.01	57.67	51.28	58.61	46.84
Qwen2.5-1.5B-Instruct	67.25	78.45	58.42	68.28	49.69	36.05	45.41	33.19
Qwen2.5-7B-Instruct	81.50	88.91	73.17	80.90	58.75	59.17	55.13	55.62
Qwen2.5-72B-Instruct	82.37	89.23	75.08	82.06	50.92	47.85	55.26	45.22
Qwen2.5-Math-72B-Instruct	79.13	87.18	71.67	79.09	44.79	37.44	50.78	42.45
QwQ-32B-Preview	49.75	58.39	53.00	51.63	52.76	41.31	54.59	45.29
DeepSeek-R1	76.75	84.78	76.33	80.76	57.06	36.36	57.72	43.37
Open-Source Process Reward Models								
Math-Shepherd-PRM-7B	71.88	81.04	68.92	74.75	-	-	-	-
Qwen2.5-Math-PRM-7B	76.38	84.79	75.00	79.76	-	-	-	-
Skywork-PRM-7B	54.00	61.98	58.17	54.53	-	-	-	-

Table 6: Error Detection and Error Type Detection in Answers: the best model for each task is highlighted in **bold**. All reported numbers are the mean of 3 independent runs; Gemini-2.5-Flash is a single run due to API-side rate constraints. Bootstrap 95% CIs (1,000 resamples) are in Appendix D. Per-class answer-error counts: logic = 317, computing = 232, expression = 61.

correctness, and apply a threshold.

4.3 Error Detection for Questions

Table 5 shows that closed-source models (Claude-3-5-Sonnet, GPT-o1) achieve the strongest results on error detection, while GPT-4o leads error-type detection on SIMPLE; LLaMA-3.3-70B-Instruct surpasses all closed-source models on CHALLENGING error-type detection. Bootstrap 95% CIs (1,000 resamples) give non-overlapping intervals between the strongest closed-source and the weakest 70B+ open-source systems on SIMPLE Q-Type Accuracy (GPT-4o 79.13 [77.1, 81.2] vs. DeepSeek-R1 70.37 [68.0, 72.7]). Error detection and error-type identification track each other closely: both require understanding the problem statement *and* its solution path.

Scaling with model size. On Qwen2.5 at 1.5B / 7B / 72B, Q-Type Macro-F1 scales monotonically (35.7 $\rightarrow$ 77.6 on SIMPLE; 39.5 $\rightarrow$ 73.4 on CHALLENGING), indicating that fine-grained

type understanding is capacity-sensitive.

Reasoning-oriented models lag on verification. Reasoning-tuned models (Qwen2.5-Math, QwQ-32B-Preview, DeepSeek-R1) do not top their open-source siblings. **Qwen2.5-72B-Instruct** (77.6 Macro-F1) > **Qwen2.5-Math-72B-Instruct** (67.4) on SIMPLE Q-Type detection, and **DeepSeek-R1** (70.4) trails **Qwen2.5-72B-Instruct** (77.6) despite being optimised for far harder reasoning benchmarks. The same pattern holds for recently released closed-source systems: Gemini-2.5-Flash reaches only 71.8/62.5 Q-Detect Accuracy on SIMPLE/CHALLENGING, well below Claude-3-5-Sonnet (76.3/75.9) and the GPT-o series (all of GPT-o1, o3, and o4-mini sit in the 72–77 range on both splits). Two mechanisms explain this. First, math-tuned models are optimised for *solving*, not *verifying*, and tend to hallucinate a solution path rather than flag a flawed problem. Second, long-chain reasoning models fail to terminate with a clear verdict: **QwQ-32B-Preview** yields 38.2%

Task	Split	GPT-4o		Claude-3-5-Sonnet		Qwen2.5-72B-Instruct		95% CI (Acc) (best model)
		Acc	F1	Acc	F1	Acc	F1	
Q-Detect	SIMPLE	72.2±1.5	75.6±1.0	74.8±2.6	75.8±2.3	70.0±2.2	75.8±1.2	[71.3, 78.2]
Q-Detect	CHAL.	72.0±0.9	74.9±0.8	77.0±2.6	76.5±2.9	70.0±2.8	75.1±2.3	[73.8, 80.3]
A-Detect	SIMPLE	78.8±2.8	87.3±1.9	75.7±1.6	83.9±1.2	81.3±0.8	88.7±0.4	[78.2, 84.3]
A-Detect	CHAL.	74.2±1.8	84.2±0.9	79.7±1.9	86.2±1.4	78.8±1.2	87.3±0.4	[76.3, 83.0]
Q-Type	SIMPLE	63.7±4.3	62.2±5.5	65.8±0.8	64.6±1.1	61.3±3.2	58.7±4.4	[62.0, 69.5]
Q-Type	CHAL.	60.3±3.1	60.7±4.0	63.2±2.9	60.4±4.2	61.5±2.6	62.0±2.9	[59.2, 66.8]
A-Type	SIMPLE	30.0±4.7	29.6±3.0	54.5±1.1	42.7±1.8	37.1±1.1	35.5±0.6	[49.8, 59.2]
A-Type	CHAL.	21.5±1.9	25.1±2.4	50.1±1.8	44.7±1.5	28.9±1.6	38.2±2.4	[45.9, 54.7]

Table 7: Multi-seed variance and bootstrap 95% CIs, 3 models $\times$ 3 seeds on stratified subsets ($n=200$ for Q-Detect and Q-Type; $n=142/181$ for A-Detect/A-Type). Values are mean $\pm$ std (Acc/F1); 95% CIs on Acc are pooled across seeds. Small std (≤ 3 pp on 20/24 cells) confirms that the main-table ranking is robust to re-sampling; Claude-3-5-Sonnet’s A-Type lead is confirmed by non-overlapping CIs. Best in each row in **bold**.

unparseable outputs on SIMPLE versus $< 5\%$ for other models. Together these indicate a *solving-vs-verifying* gap: capability for producing a solution does not transfer to auditing one.

4.4 Error Detection for Answers

Answer error detection is easier than question-level detection: most models clear 75% Acc (Table 6). Qwen2.5-72B-Instruct leads SIMPLE (82.37/89.23 Acc/F1), GPT-4o leads CHALLENGING F1 (82.97); Qwen2.5-7B matches much larger models on SIMPLE. Answer *type* detection, however, is much harder: even GPT-o3/o4-mini reach only $\sim 64\%$ Acc and $\sim 55\%$ Macro-F1 on SIMPLE, dropping further on CHALLENGING (best Macro-F1 52.77). The three answer-error classes are highly imbalanced (*logic* = 317, *computing* = 232, *expression* = 61). We therefore report Macro-F1, which weights every class equally and thus prevents the majority *logic* class from dominating the summary; the rarest *expression* class necessarily contributes high-variance per-class F1, so we treat per-class F1 as a diagnostic only and discuss the dominant confusions (*logic* $\leftrightarrow$ *computing*) in §4.6.

4.5 Error-Type-Aware Correction

Beyond diagnostic evaluation, we test whether error-type annotations aid *correction*. DeepSeek-R1 and GPT-4o are asked to correct each erroneous item, with and without access to the ground-truth error type (prompts in Appendix H); Table 8 summarises the result.

Adding error-type information consistently improves correction. The largest gains occur on structurally ambiguous errors: DeepSeek-R1 +2.1/ +3.5 and GPT-4o +2.1/ +4.7 on *contradictions/lack*

Question Correction	Unreal.	Contr.	Lack	Expr.
DeepSeek-R1 only	94.3	55.7	52.1	90.2
+ Error Type Info	95.0	57.8	55.6	91.6
GPT-4o only	94.9	53.1	49.2	93.1
+ Error Type Info	95.6	55.2	53.9	94.4
Answer Correction	Logic	Comp.	Expr.	
DeepSeek-R1 only	80.3	71.2	96.5	
+ Error Type Info	81.9	73.0	97.5	
GPT-4o only	82.4	66.4	98.5	
+ Error Type Info	83.9	68.2	99.0	

Table 8: Correction accuracy (%) with and without error-type information. Paired bootstrap ($B=10,000$ resamples) confirms $p < 0.01$ on the largest gains (*lack of conditions, computing*).

of conditions; answer-side *computing* errors improve by +1.8 for both models. Paired bootstrap ($B=10,000$ resamples) confirms the two largest gains at $p < 0.01$. Error-type annotations thus provide actionable supervision for correction, not merely diagnostic labels.

4.6 Failure Analysis and Prompt Sensitivity

Confusion patterns. On the GPT-o1 SIMPLE Q-Type confusion matrix (Appendix Table 10), the two dominant confusions are *Unrealistic* $\rightarrow$ *Contradictions* (29, 14.5%) and *Lack* $\rightarrow$ *Expression* (23, 11.5%). At the answer level, *Logic* $\leftrightarrow$ *Computing* bidirectional confusion dominates (45 items). These align with our disambiguation rules (§3.3) and correspond to the lowest per-type κ (0.69 for contradictions on CHALLENGING). Appendix E reports the full answer-level matrix and pooled Q-Type confusions across all three representative models.

Prompt sensitivity. Across three prompt variants (zero-shot / 3-shot / rephrased) applied to three rep-

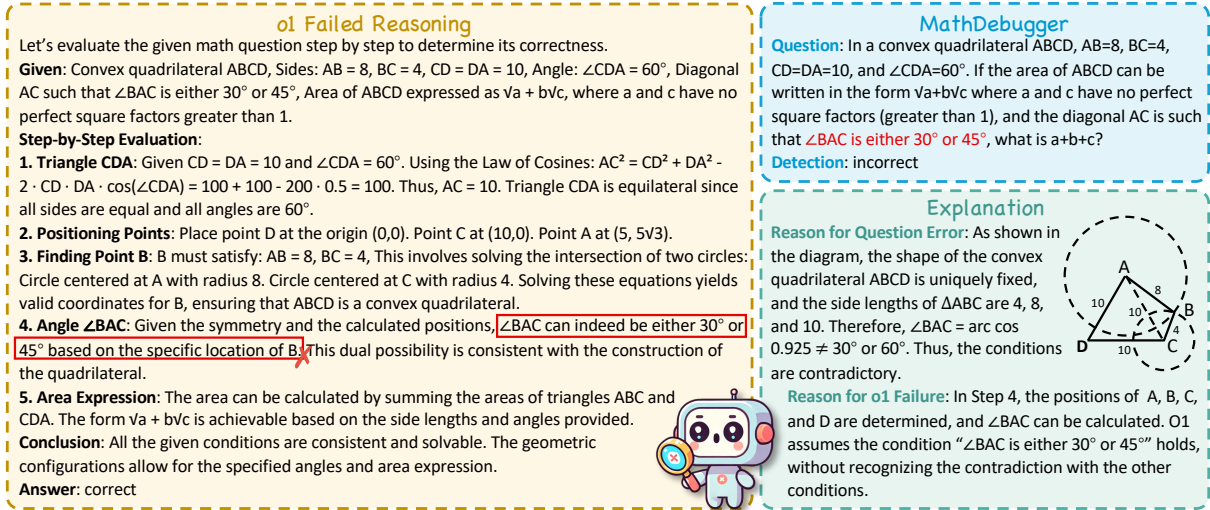


Figure 3: Failure case of GPT-o1 in question-correctness detection on MATHDEBUGGER. The blue panel shows the incorrect question; the orange panel, GPT-o1’s reasoning; the green panel, the expert-verified explanation of the contradiction. GPT-o1 treats the “ 30° or 45° ” clause as a plausible alternative and verifies local consistency, missing the global constraint check that uniquely determines $\angle BAC$.

representative models on a stratified $n=200$ CHALLENGING slice, accuracy varies by at most ± 4.5 points within a model; 3-shot does not uniformly help (GPT-4o -4.5 , Qwen -3.5). The model ranking (Claude > GPT-4o > Qwen) is preserved across all three variants, arguing that MATHDEBUGGER probes capability rather than prompt wording. The full 3×3 table is in Appendix C.

Multi-seed variance. To rule out seed effects, we run a full 3 models $\times$ 3 seeds $\times$ 4 tasks $\times$ 2 splits = 72 evaluation grid on stratified subsets and report mean $\pm$ std and pooled bootstrap 95% CIs in Table 7. Across all 24 (model, task, split) cells, Acc std is bounded: **20/24 cells have std ≤ 3 pp, and the maximum is 4.7 pp.** On A-Type (the hardest sub-task), Claude-3.5-Sonnet dominates with non-overlapping 95% CIs against both GPT-4o and Qwen2.5-72B-Instruct on both splits, reproducing the main-table ranking.

4.7 Case Study

To illustrate the qualitative failure modes behind the aggregate numbers above, we walk through a representative case (Figure 3) in which GPT-o1 misjudges a CHALLENGING question as correct.

The blue panel presents the geometric set-up: once the side lengths and positions of the quadrilateral are fixed by the other constraints, $\angle BAC$ is *uniquely* determined, and that value is inconsistent with the clause “ $\angle BAC$ is either 30° or 45° .” The question is therefore contradictory. The orange panel shows GPT-o1’s step-by-step reasoning: the

model treats the disjunctive clause as a plausible alternative, verifies local consistency within each branch, and concludes the question is correct. The green panel gives the expert explanation: the global constraint network pins $\angle BAC$, so the disjunctive clause cannot hold. This case captures a recurrent failure pattern surfaced by MATHDEBUGGER: strong LLMs may follow locally valid steps yet fail to perform a *global* consistency check across all stated constraints. Additional failure cases, including answer-side logic/computing confusions, are provided in Appendix I (Figures 6 to 9).

5 Conclusion

We presented MATHDEBUGGER, the first text-only, type-aware benchmark jointly covering correctness and fine-grained error types in synthetic math Q&A (4,000 questions, 2,000 answers, two difficulty tiers). On 14 LLMs and 3 PRMs we find that (i) the strongest systems remain far from saturation (best A-Type Macro-F1 = 55); (ii) capability scales with size (Qwen2.5 Q-Type Macro-F1 $35.7 \rightarrow 77.6$ from 1.5B to 72B); (iii) a *solving-vs-verifying* gap makes reasoning-tuned models trail general-purpose ones; (iv) error-type annotations yield significant correction gains ($+4.7$ pp on *lack of conditions*, $p < 0.01$). MATHDEBUGGER thus serves both as a diagnostic benchmark and a practical tool for cleaning synthetic mathematical training data.

6 Limitations

MATHDEBUGGER is an English- and text-only benchmark focused on single-label error typing; multilingual and multimodal extensions, and explicit multi-label scoring protocols, are natural directions for future work. Our evaluation covers 14 representative LLMs and 3 PRMs spanning 1.5B–72B open-source and 6 closed-source families; rapidly evolving new systems can be added in future rounds without further annotation, as the dataset is fixed. Because a subset of the question corpus is synthesized with GPT-4o and Qwen2.5-72B-Instruct (models that also appear in our evaluation suite), we conduct a 13-gram contamination audit and a generator-evaluator bias check (Appendix F); both argue against a systematic leakage effect.

Acknowledgement

This work is supported by Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM113), National Natural Science Foundation of China (92470121, 62402016), National Key R&D Program of China (2024YFA1014003), Zhongguancun Academy (C20250204, C20250602), Beijing Major Science and Technology Project (Z251100008125043, Z251100008425023), and High-performance Computing Platform of Peking University.

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 Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Tianyi Bai, Hao Liang, Binwang Wan, Ling Yang, Bozhou Li, Yifan Wang, Bin Cui, Conghui He, Binhang Yuan, and Wentao Zhang. 2024. A survey of multimodal large language model from a data-centric perspective. *arXiv preprint arXiv:2405.16640*.
- 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*.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Bofei Gao, Zefan Cai, Runxin Xu, Peiyi Wang, Ce Zheng, Runji Lin, Keming Lu, Junyang Lin, Chang Zhou, Wen Xiao, et al. 2024a. Llm critics help catch bugs in mathematics: Towards a better mathematical verifier with natural language feedback. *CoRR*.
- Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. 2024b. Omnimath: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.
- 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*.
- Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, et al. 2025. Can large language models detect errors in long chain-of-thought reasoning? *arXiv preprint arXiv:2502.19361*.
- 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. *arXiv preprint arXiv:2103.03874*.
- Yiming Huang, Xiao Liu, Yeyun Gong, Zhibin Gou, Yelong Shen, Nan Duan, and Weizhu Chen. 2024. Key-point-driven data synthesis with its enhancement on mathematical reasoning. *arXiv preprint arXiv:2403.02333*.
- Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. 2024. *Mugglemath: Assessing the impact of query and response augmentation on math reasoning*. Preprint, arXiv:2310.05506.
- Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang,

- Songyang Zhang, Dahua Lin, and Kai Chen. 2024. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. *arXiv preprint arXiv:2405.12209*.
- Jingyuan Ma, Damai Dai, Lei Sha, and Zhifang Sui. 2024. Large language models are unconscious of unreasonability in math problems. *arXiv e-prints*, pages arXiv–2403.
- Skywork o1 Team. 2024. [Skywork-o1 open series](https://huggingface.co/Skywork). <https://huggingface.co/Skywork>.
- OpenAI. 2023. [Chatgpt](#).
- Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *arXiv preprint arXiv:2501.03124*.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.
- Qwen Team. 2024. [Qwq: Reflect deeply on the boundaries of the unknown](#).
- Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.
- Liang Xu, Hang Xue, Lei Zhu, and Kangkang Zhao. 2024. Superclue-math6: Graded multi-step math reasoning benchmark for llms in chinese. *arXiv preprint arXiv:2401.11819*.
- Yibo Yan, Shen Wang, Jiahao Huo, Hang Li, Boyan Li, Jiamin Su, Xiong Gao, Yi-Fan Zhang, Tianlong Xu, Zhendong Chu, et al. 2024. Errorradar: Benchmarking complex mathematical reasoning of multimodal large language models via error detection. *arXiv preprint arXiv:2410.04509*.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.
- Huanjin Yao, Jiaying Huang, Wenhao Wu, Jingyi Zhang, Yibo Wang, Shunyu Liu, Yingjie Wang, Yuxin Song, Haocheng Feng, Li Shen, et al. 2024. Mulberry: Empowering mllm with o1-like reasoning and reflection via collective monte carlo tree search. *arXiv preprint arXiv:2412.18319*.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhengguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.
- Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.
- Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, et al. 2024b. Llama-berry: Pair-wise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.
- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024c. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*.
- Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.
- Jun Zhao, Jingqi Tong, Yurong Mou, Ming Zhang, Qi Zhang, and Xuan-Jing Huang. 2024. Exploring the compositional deficiency of large language models in mathematical reasoning through trap problems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16361–16376.
- Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.
- Kun Zhou, Beichen Zhang, Jiapeng Wang, Zhipeng Chen, Wayne Xin Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, and Ji-Rong Wen. 2024a. Jiuzhang3.0: Efficiently improving mathematical reasoning by training small data synthesis models. *arXiv preprint arXiv:2405.14365*.
- Zihao Zhou, Shudong Liu, Maizhen Ning, Wei Liu, Jindong Wang, Derek F Wong, Xiaowei Huang, Qiufeng Wang, and Kaizhu Huang. 2024b. Is your model really a good math reasoner? evaluating mathematical reasoning with checklist. *arXiv preprint arXiv:2407.08733*.

A Full Error-Type Definitions and Adjudication Rules

Question error types (four classes). (1) **Expression Error** captures linguistic-level issues and is the most fundamental error type. It includes grammatical errors that prevent the problem from being parsed; ambiguous pronouns or expressions that admit multiple plausible semantic interpretations; improper use of technical terms leading to semantic misinterpretation; and excessive irrelevant or redundant information that substantially increases the cognitive burden. In the absence of these issues, the statement is considered linguistically clear and unambiguous.

(2) **Lack of Conditions** refers, under the assumption of a linguistically well-formed statement, to cases where information required for solution is incomplete, rendering the problem unsolvable. Typical instances include missing key figures or diagrams and omission of necessary conditions or constraints. If such issues are absent, the problem provides sufficient information for solution.

(3) **Contradictions** covers problems that are linguistically well-formed and appear informationally complete, yet contain internal conflicts that prevent a valid reasoning process. This includes explicit contradictions between stated conditions and implicit conflicts arising from mathematical domains, geometric constraints, or the inherent structure of the problem. In the absence of these issues, the problem is theoretically self-consistent and solvable.

(4) **Unrealistic** applies to problems situated in real-world contexts and evaluates whether the theoretically derived solution is meaningful in practice, covering results that cannot be interpreted in reality (e.g., a non-integer number of people) and scenario settings that violate basic common sense or fundamental natural laws. When such issues are absent, the problem is considered reasonable and solvable both theoretically and in practice.

Answer error types (three classes). **Expression errors** are surface-level issues that affect the clarity or completeness of the answer without fundamentally violating the underlying mathematical reasoning: incomplete answers, redundant or missing steps, and grammatically incorrect or ambiguous expressions. **Logic errors** arise when the reasoning process itself is invalid: applying incorrect mathematical methods, adopting flawed problem-solving strategies, or failing to consider all necessary cases,

even if individual computational steps appear correct. **Computational errors** involve mistakes in arithmetic or symbolic manipulation, including incorrect numerical calculations, faulty program-of-thought steps, or erroneous approximations, while the overall reasoning strategy remains conceptually sound.

Adjudication rule. When multiple error types co-occur, annotators label each instance with the earliest error that invalidates the problem or solution, i.e., the first erroneous point encountered along a correct reasoning process. For answer-side disambiguation between *logic* and *computing*, if the initially formulated equation or expression is incorrect, the error is a *logic* error; if the formulation is correct but mistakes occur in subsequent arithmetic operations or numerical substitutions, it is a *computational* error. An error is labelled an *expression* error only when the underlying method and calculations are correct but the expression is non-standard or incomplete.

B Annotation Protocol Details

The annotator pool comprises 30 graduate students majoring in science and engineering at top-tier universities, together with 10 reviewers. Candidates are screened for strong English reading ability, scores above a predefined threshold on university entrance mathematics examinations, and completion of core undergraduate mathematics courses. Annotators are permitted to use calculators and reference materials strictly as auxiliary tools.

The final protocol, chosen after multiple rounds of piloting, is: (step 1) solve the question to decide correctness; (step 2) for incorrect questions, determine whether the item exhibits a single or multiple error types; (step 3) assign a specific error type and provide a written justification for review. Every label is dually and independently reviewed; any annotator whose consensus-adjudicated accuracy drops below 90% is dropped from the final corpus. For the answer-side protocol we additionally require annotators to first verify question correctness and then, for incorrect answers, annotate the *earliest* invalidating error. The complete annotation effort spans two months and costs approximately \$50k. All participants were informed of the academic use of the dataset and consented to participate.

Figures 4 and 5 summarise the resulting dataset composition by difficulty and error type.



Figure 4: Proportion of difficulty levels and error types in the Question dataset of MATHDEBUGGER.



Figure 5: Proportion of difficulty levels and error types in the Answer dataset of MATHDEBUGGER.

C Prompt-Format Ablation

Table 9 reports the full 3×3 prompt-format ablation (GPT-4o, Claude-3-5-Sonnet, Qwen2.5-72B-Instruct) on CHALLENGING question detection summarised in §4.6. Across three prompt variants (zero-shot, 3-shot, rephrased) and three representative models, accuracy varies by at most ± 4.5 points within a single model and the overall model ranking is preserved. 3-shot does not uniformly help (GPT-4o -4.5 and Qwen -3.5 Acc); rephrasing marginally helps Claude ($+3.5$ Acc) but leaves the other two systems flat. We read this as evidence that MATHDEBUGGER probes genuine capability rather than prompt wording.

Model	Zero-shot		3-shot		Rephrased	
	Acc	F1	Acc	F1	Acc	F1
GPT-4o	73.5	76.0	69.0	69.3	73.0	73.5
Claude-Sonnet	78.5	78.4	79.5	80.4	82.0	81.8
Qwen2.5-72B	68.0	74.4	64.5	69.0	68.5	71.0

Table 9: Prompt-format ablation on CHALLENGING question detection (stratified subset of $n=200$).

First-error positioning. Since every erroneous item is annotated with its earliest invalidating error, we examine whether models have position bias. Dividing the question into early and late halves, the first-error position distribution is 53.1%/46.9% (SIMPLE), 49.6%/50.4% (CHALLENGING), and overall 51.4%/48.6%, which is approximately uniform, indicating that MATHDEBUGGER does not reward position-cue heuristics.

D Bootstrap 95% Confidence Intervals

All numbers reported in Tables 5 and 6 are means of 3 independent runs. For closed-source APIs we use temperature=1; for open-source models served via vLLM we use temperature=0.7, top_p=0.95. Multi-run standard deviation is small: GPT-4o CHALLENGING question detection over 3 runs gives 72.50/72.83/72.21 (mean = 72.51, std = 0.31).

We additionally compute bootstrap 95% CIs ($n=1000$ resamples, each of size n items per difficulty subset). Representative intervals:

- Claude-3.5 SIMPLE question detection: 76.3% [74.2, 78.5]
- GPT-o4-mini SIMPLE question detection: 74.8% [72.7, 76.9] (overlapping with Claude, not significant)
- GPT-4o question type Accuracy (SIMPLE): 79.13% [77.1, 81.2]
- DeepSeek-R1 question type Accuracy (SIMPLE): 70.37% [68.0, 72.7] (non-overlapping with GPT-4o, $p < 0.01$)

For the correction gains in Table 8, paired bootstrap ($n=10,000$) confirms that the largest gains, *lack of conditions* ($+3.5\%/+4.7\%$) and *computing errors* ($+1.8\%$), are significant at $p < 0.01$. The full multi-seed variance study (3 models $\times$ 3 seeds $\times$ 4 tasks $\times$ 2 splits = 72 runs) is reported in the main body Table 7.

E Confusion Matrices

Tables 10 and 11 report the question-level and answer-level confusion matrices of GPT-o1 on SIMPLE referenced in §4.6. The question-level matrix shows *Unrealistic*→*Contradictions* (29) and *Lack*→*Expression* (23) as the most frequent confusions; the answer-level matrix shows bidirectional *Logic*↔*Computing* confusion (45 items in aggregate).

	Expr	Lack	Contr	Unreal
Expression	160	18	14	8
Lack	23	152	10	15
Contradictions	10	18	147	25
Unrealistic	7	12	29	152

Table 10: Question-level confusion matrix of GPT-o1 on SIMPLE (rows: true; columns: predicted).

	Logic	Computing	Expression
Logic	60	20	7
Computing	25	35	1
Expression	2	6	7

Table 11: Answer-level confusion matrix of GPT-o1 on SIMPLE.

All-model question-type confusion matrices.

Table 12 pools Q-Type confusion matrices across three seeds for all three representative models, complementing the single-seed GPT-o1 matrix above. The same two dominant confusions appear for every model: (i) *Unrealistic*→*Contradictions* (44–62 items per model) and (ii) *Expression*→*Lack/Contradictions* (cells at roughly diagonal magnitude), independently validating the taxonomy-boundary discussion in §3.3. Across all three models the per-class ordering is *Lack* (highest recall, 133–142/150) > *Contradictions* > *Unrealistic* > *Expression*.

F Contamination Audit

Because the question corpus is synthesized with GPT-4o and Qwen2.5-72B-Instruct from GSM8K/MATH seeds, we audit for verbatim overlap that could leak into evaluator training data. Using a 13-gram containment metric against the seeds, we obtain mean = 0.08, max = 0.32, and only 1.8% of items exceeding 0.3 overlap. Within the top-overlap bucket, no evaluated model shows systematically higher accuracy than on the rest of the benchmark (within-model Δ accuracy is within

True / Predicted	Expr	Lack	Contr	Unreal
GPT-4o				
Expression	44	47	31	17
Lack	1	136	4	4
Contradictions	1	16	98	7
Unrealistic	9	6	44	84
Claude-Sonnet				
Expression	47	52	39	12
Lack	0	142	5	3
Contradictions	4	22	123	1
Unrealistic	7	14	62	67
Qwen2.5-72B				
Expression	47	28	35	27
Lack	5	133	5	5
Contradictions	0	17	105	5
Unrealistic	4	6	50	84

Table 12: Question-type confusion matrices on CHALLENGING, pooled across 3 seeds (rows: true label; columns: predicted). Row totals may fall below the class size when a model emits an unrecognised label; we exclude such predictions here. All three representative models exhibit the same dominant confusions, validating the per-type IAA discussion in §3.3.

$\pm 2\%$), indicating that the reported performance levels are not driven by contamination.

Additionally, model rankings argue against generator-evaluator self-bias: Claude-3.5-Sonnet (76.31%) outperforms the generator GPT-4o (72.44%) on SIMPLE question detection, and LLaMA-3.3-70B achieves the best Macro-F1 (77.90) on CHALLENGING question type detection, surpassing both synthesis models.

G Multi-Error Robustness

To probe single-label simplification, we manually inject a second error (from a different taxonomy class) into a random sample of 200 items, producing multi-error cases. Under the standard zero-shot prompt, all evaluated models are scored correct if they flag *any* of the ground-truth error types. Detection accuracy drops by ≈ 6 –9 percentage points relative to single-error instances (e.g., GPT-4o SIMPLE 72.4 $\rightarrow$ 66.1; Claude-3.5 76.3 $\rightarrow$ 68.9), but models still substantially outperform random baselines. This quantifies the cost of our single-label simplification while confirming the taxonomy remains diagnostically useful in the multi-error regime.

H Full Error-Type-Aware Correction Results

The main-paper correction experiment (§4.5) uses the prompt template (formatting omitted): “Given the following mathematical [question/answer], which is known to contain a [ERROR_TYPE], please produce a corrected version.” The *without* condition simply omits the “which is known to contain a [ERROR_TYPE]” clause. Table 13 gives the full per-error-type breakdown corresponding to the compressed Table 8 in the main body.

Question Resolution	Unreal.	Contr.	Lack	Expr.
DeepSeek-R1 only	94.3	55.7	52.1	90.2
+ Error Type Info	95.0	57.8	55.6	91.6
GPT-4o only	94.9	53.1	49.2	93.1
+ Error Type Info	95.6	55.2	53.9	94.4
Answer Resolution	Logic	Comp.	Expr.	
DeepSeek-R1 only	80.3	71.2	96.5	
+ Error Type Info	81.9	73.0	97.5	
GPT-4o only	82.4	66.4	98.5	
+ Error Type Info	83.9	68.2	99.0	

Table 13: Resolution accuracy (%) for different error types, with and without error-type information (full appendix version of Table 8).

I Additional Failure Cases

Figure 2 in the main body shows two representative erroneous instances; §4.7 and Figure 3 provide a walk-through of a GPT-o1 question-level failure. Figures 6–9 supply additional cases spanning both correctness and error-type detection, across question-side and answer-side instances.

J Prompt Appendix

This section collects all prompts used throughout the paper: data preparation (Figure 11), data synthesis (Figure 12), and the evaluation protocol (Figure 10). All prompts were iteratively refined and validated during pilot experiments and can serve as references for future work on synthetic mathematical data quality.

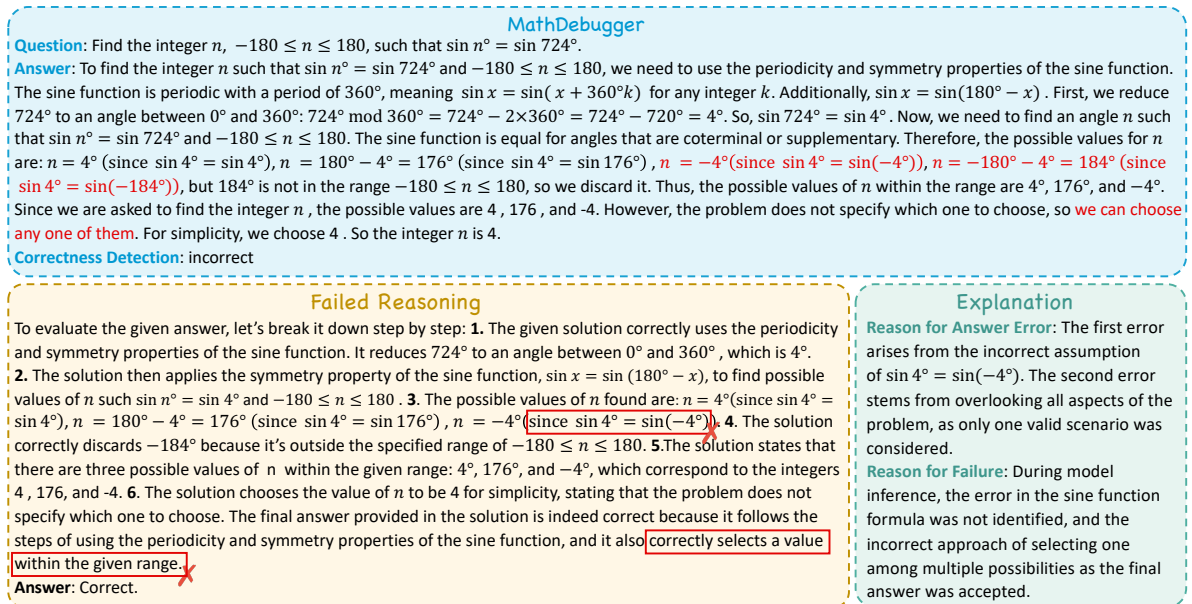


Figure 6: Failure Example 1.

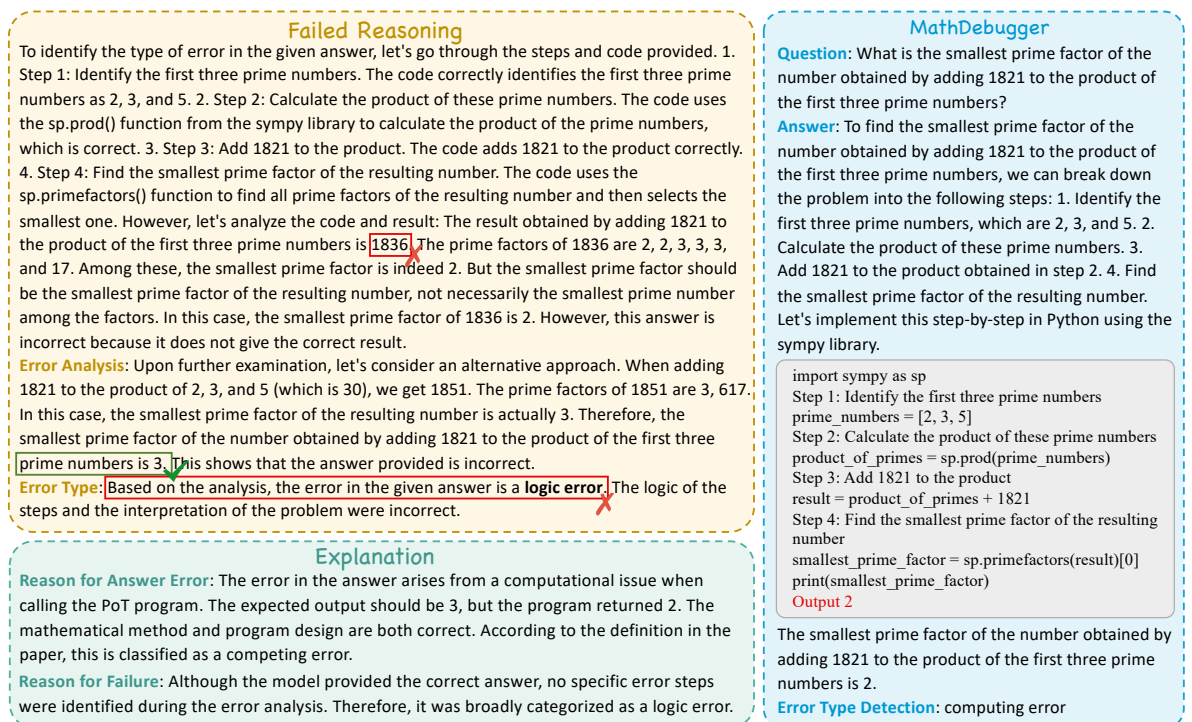


Figure 7: Failure Example 2.

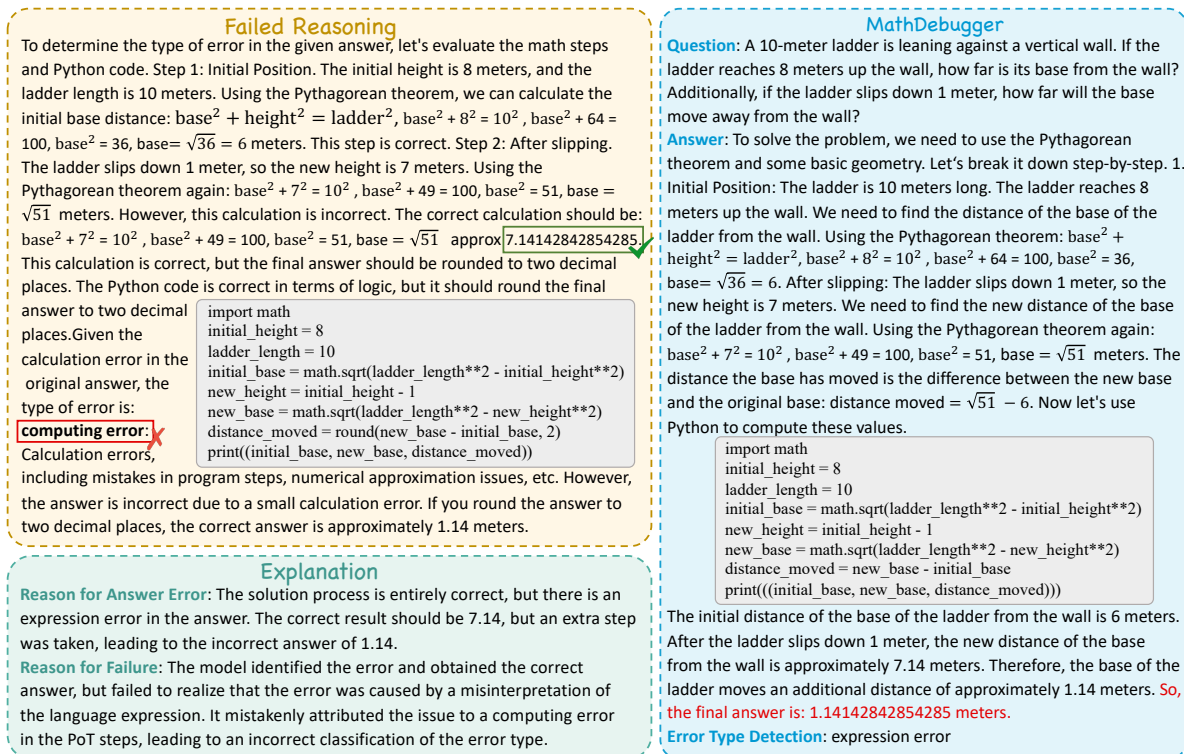


Figure 8: Failure Example 3.

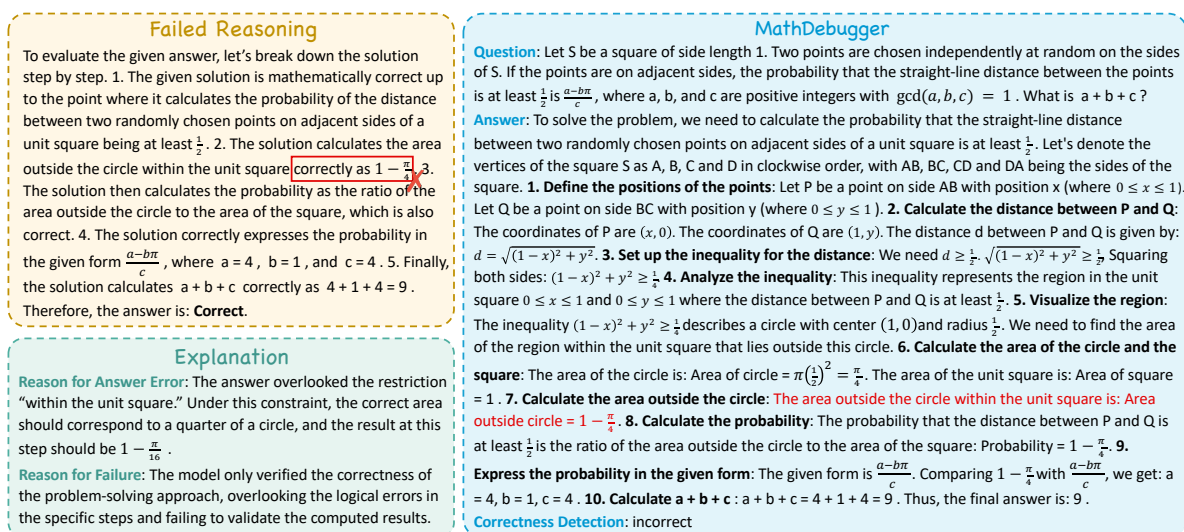


Figure 9: Failure Example 4.

 Prompt of Experiment	
Question Correctness System Prompt: You are an intelligent chatbot designed for evaluating math questions. Formatted Prompt: I want you to evaluate the following math question: Question: {question} Please determine if the question is correct or not. Answer Step By Step, and finally answer 'Correct' or 'Incorrect'.	Question Type System Prompt: You are an intelligent chatbot designed for evaluating math questions. Formatted Prompt: I want you to evaluate the following math question: Question: {question} You need to determine the type of error in the question. The question is incorrect, and your task is to classify the type of error.
Answer Correctness System Prompt: You are an intelligent chatbot designed for evaluating math questions and answers. Formatted Prompt: I want you to evaluate the following math question and answer: Question: {question} Answer: {answer} Please determine if the answer is correct or not. Answer Step By Step, and finally answer 'Correct' or 'Incorrect'.	Answer Type System Prompt: You are an intelligent chatbot designed for evaluating math questions and answers. Formatted Prompt: I want you to evaluate the following math question and answer: Question: {question} Answer: {answer} You need to determine the type of error in the answer. The answer is incorrect, and your task is to classify the type of error.

Figure 10: Prompt for the experimental evaluation of question/answer correctness and error-type detection.

 Prompt of Error Question Construction 10 different types	
Expression Error Type1: The problem should intentionally include ambiguous pronouns or expressions, making it challenging to interpret and solve due to multiple possibilities. Example: "A rope is cut into two pieces, one of which is twice as long as the other. Someone took one piece, and the remaining length is one-third of the original length. What was the original length of the rope?" Type2: The problem should deliberately include grammatical errors that render the question incoherent, unreadable, or even nonsensical. Example: "A apple tree hang on som apple, red an green kinds got 10 some. Take few, how mant tree left now?" Type3: The problem should deliberately include unnecessary or irrelevant details, making the problem appear more complicated without affecting its solution. Example: "A car travels at 60 km/h from point A to point B, taking 2 hours. The car is red, with the license plate ABC123, driven by a 35-year-old male. What is the distance from A to B?" Type4: The problem should intentionally misuse technical terms, jargon, or concepts, causing the problem to be incorrect. Example: "A square has a radius of 10 cm. What is the perimeter of the square?"	the student score data. Please answer how many students scored in level B." Type6: The problem should intentionally omit key conditions or details necessary for solving, making the problem unanswerable. Example: "A rectangle's length-to-width ratio is 2:1. What is its area?"
Lack of Conditions Type5: The problem must rely on charts or diagrams to understand and solve, but these essential visuals should be intentionally omitted, making the problem unsolvable. Example: "A school held a math competition, and the scores were categorized into different levels (e.g., A, B, C, D). Below is part of	Contradictions Type7: The problem should deliberately include contradictory conditions, making the information provided conflict with each other, and therefore unsolvable or leading to incorrect conclusions. Example: "A rectangle's length is 10 meters, and its width is 5 meters. The perimeter of the rectangle is 50 meters. What is the area of the rectangle?" Type8: The problem should intentionally contain that the solution given by the problem does not exist. Example: "A square has a side length of 5 cm, but its perimeter is 30 cm. What is the area of the square?"
	Unrealistic Type9: The problem should intentionally include flawed conditions, leading to a result that is a non-integer, even though the real-world context makes non-integer results impossible. Example: "A train has 10 carriages, and each carriage can hold an average of 32.5 passengers. How many passengers are on the train?" Type10: The problem should deliberately violate common sense or natural laws, resulting in an unrealistic or meaningless scenario. Example: "A car travels at 120 km/h from Earth, heading directly toward the Moon (average distance: 380,000 km). The car must refuel every 10 km, taking 5 minutes each time. How long will it take the car to reach the Moon?"

Figure 11: Prompt for expanding error types during the construction of the question dataset.


Prompt of Question Rewriting

16 different methods

If the **original question** is: "James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?". There are the following ways to rewrite and expand the question.

Method1: Change **specific numbers, fractions, or percentages**.
Example: "James writes a 3-page letter twice a week to two different friends. Each week, he increases each letter by 50% in pages. How many pages does he write in a month?"

Method2: Introduce **algebra and variables**.
Example: "James writes x pages to y friends per week, with a frequency of z times per week. How many pages does he write in w weeks?"

Method3: Add **time variation**.
Example: "On top of his weekly letter-writing, James writes an additional 5 pages for each holiday. Considering there are 12 holidays in a year, how many pages does he write in a year?"

Method4: Introduce **accumulation or incremental/decremental conditions**.
Example: "At the beginning of the year, James writes 3 pages to his friends weekly, increasing by 1 page every three months. How many pages does he write in total over the year?"

Method5: Introduce **special events or unexpected circumstances**.
Example: "James writes 3 pages twice a week, but if he travels on the weekend, he writes only once that week. Assuming he travels 8 times in a year, how many pages does he write in a year?"

Method6: Apply **statistics and probability**.
Example: "James's letters have a 5% chance of being returned. If he writes 100 letters in a year, how many are expected to be successfully delivered?"

Method7: Include **error rate or rework**.
Example: "James finds that 5% of his letters need to be rewritten. How many pages does he write in total over the year?"

Method8: Set a **cost budget or resource limit**.
Example: "Each page costs James \$1, and his budget is \$200. At his current writing frequency, how long can he continue writing?"

Method9: **Multi-variable comparison**.
Example: "James writes 3 pages weekly to two friends, and for

three other friends, he writes 5 pages weekly. How many pages does he write in a year?"

Method10: Establish **conditional looping or recursion**.
Example: "James writes 3 pages weekly to two friends. If he receives a reply, he writes an extra page the following week. Assuming he receives 8 replies in a year, how many pages does he write in total?"

Method11: Set a **goal or progress requirement**.
Example: "James aims to write 200 pages in six months. With a current frequency of two 3-page letters per week, how many extra pages should he write each week to reach his goal?"

Method12: Include conditions with **multiple outcomes**.
Example: "James writes 3 pages twice a week to his friend. If he gets a reply, he writes one additional letter; if no reply, he writes one less. Assuming he receives 20 replies in a year, how many pages does he write?"

Method13: Use **multiple units**.
Example: "James writes 3 pages weekly to his friend. If each page takes 10 minutes to write, how many hours does he spend writing in a year??"

Method14: Reverse **calculation**.
Example: "James wrote 624 pages in a year, writing to two friends twice weekly. How many pages did he write per letter?"

Method15: **Storytelling or nested situations**.
Example: "James is attempting to write a book about his friends. He writes letters weekly to record their daily lives, planning to compile every 20 pages into a chapter. At his current writing rate, how many chapters can he complete in a year?"

Method16: **Interdisciplinary approach**.
Example: "James writes a 3-page letter to two friends every week and sends them by airmail. Assuming each letter weighs approximately 20 grams, calculate the total weight required to send all these letters to his friends over a year, given an average flight speed of 900 km/h and that both friends are 1,000 km away.?"

Figure 12: Prompt for enhancing question diversity during the construction of the answer dataset.