Exposing the Achilles' Heel: Evaluating LLMs Ability to Handle Mistakes in Mathematical Reasoning

Joykirat Singh Akshay Nambi Vibhav Vineet
Microsoft Research
{akshayn, vivineet}@microsoft.com

Abstract

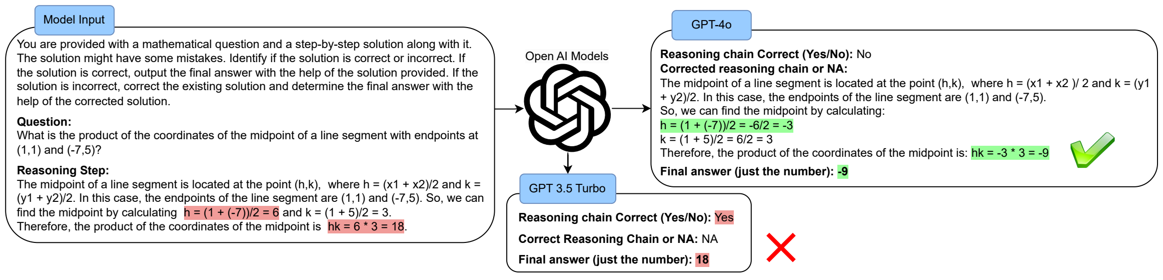
Large Language Models (LLMs) have significantly impacted the field of Math Word Problems (MWPs), transforming how these problems are approached and solved, particularly in educational contexts. However, existing evaluations often focus on final accuracy, neglecting the critical aspect of reasoning capabilities. This work addresses that gap by evaluating LLMs' abilities to detect and correct reasoning mistakes. We present a novel dataset, MWP-MISTAKE, containing MWPs with both correct and incorrect reasoning steps generated through rule-based methods and smaller language models. Our comprehensive benchmarking of state-of-the-art models such as GPT-4o and GPT4 uncovers important insights into their strengths and limitations. While GPT-4o excels in mistake detection and rectification, gaps remain, particularly in handling complex datasets and novel problems. Additionally, we identify concerns with data contamination and memorization, which affect LLM reliability in real-world applications. While OpenAI' O1 model demonstrates 90% accuracy in reasoning and final answers on complex tasks, it remains weak in mistake detection. Our findings highlight the need for improved reasoning evaluations and suggest ways to enhance LLM generalization and robustness in math problem-solving.

1 Introduction

Large Language Models (LLMs) have transformed AI applications across diverse domains, including healthcare, agriculture, and education (OpenAI, b,a). Their prowess in natural language understanding, question answering, and mathematical problem-solving has shown potential to revolutionize various human endeavors (Liu et al., 2024b). Recent advancements have fueled extensive research into applying LLMs to interpret and solve a wide array of mathematical tasks, from basic arithmetic to complex algebraic equations and calculus problems (Hendrycks et al., 2021; Zhang et al., 2024a). Math Word Problems (MWPs) involve interpreting narrative scenarios to extract mathematical concepts and apply reasoning for solutions (Srivatsa and Kochmar, 2024). Studies (Xu et al., 2024; He-Yueya et al., 2023; Deb et al., 2023) show LLMs can convert text into mathematical expressions and generate accurate results, but a critical element *mathematical reasoning* is often underexplored.

Despite achieving remarkable accuracy rates exceeding 90% on datasets like GSM-8K (Grade School Math dataset with linguistically diverse word problems) (Cobbe et al., 2021a), foundational LLMs such as Claude-3-Opus (Anthropic, 2024), Gemini Ultra (Team et al., 2024), and OpenAI GPT-4 (OpenAI et al., 2024) reveal a significant gap in our understanding of their capabilities in mathematical reasoning (Deb et al., 2023). Current research predominantly focuses on evaluating the final accuracy of MWPs (Luo et al., 2023; Yu et al., 2024), neglecting the intricate reasoning processes necessary to derive solutions. We argue that the reasoning steps play a pivotal role, and it is imperative to assess them to comprehensively analyze the foundational capabilities of these models. This necessity is further underscored by the increasing utilization of LLMs in domains such as education (Gan et al., 2023), where they serve as personalized tutors for students, aiding in teaching concepts and solving mathematical problems. Simply deriving the final answer is insufficient; the ability to guide students through correct steps, identify errors in their reasoning, and provide corrective guidance is paramount for such applications.

This paper addresses this gap by providing a comprehensive benchmark and evaluation of LLMs' performance on math word problems, including their capabilities in mistake detection and correction within the reasoning steps (Figure 1). Analyzing their error detection and rectification



You are provided with a mathematical question and a step-by-step solution along with it. The solution might have some mistakes. Identify if the solution is correct or incorrect. If the solution is correct, output the final answer with the help of the solution provided. If the solution is incorrect, correct the existing solution and determine the final answer with the help of the corrected solution.

Question:
What is the product of the coordinates of the midpoint of a line segment with endpoints at (1,1) and (-7,5)?

Reasoning Step:
The midpoint of a line segment is located at the point (h,k), where h = (x1 + x2)/2 and k = (y1 + y2)/2. In this case, the endpoints of the line segment are (1,1) and (-7,5). So, we can find the midpoint by calculating h = (1 + (-7))/2 = 6 and k = (1 + 5)/2 = 3.
Therefore, the product of the coordinates of the midpoint is hk = 6 * 3 = 18.

Reasoning chain Correct (Yes/No): No
Corrected reasoning chain or NA:
The midpoint of a line segment is located at the point (h,k), where h = (x1 + x2)/ 2 and k = (y1 + y2)/2. In this case, the endpoints of the line segment are (1,1) and (-7,5).
So, we can find the midpoint by calculating:
h = (1 + (-7))/2 = -6/2 = -3
k = (1 + 5)/2 = 6/2 = 3
Therefore, the product of the coordinates of the midpoint is: hk = -3 * 3 = -9
Final answer (just the number): -9

Reasoning chain Correct (Yes/No): Yes
Correct Reasoning Chain or NA: NA
Final answer (just the number): 18

Figure 1: Model is prompted with a question along with incorrect reasoning steps to detect any mistake and correct the reasoning step to get to the correct final answer. GPT-4o generates the correct output, while GPT-3.5Turbo fails to identify any mistake in the reasoning step. (Task - T1).

provides insights into overall problem-solving abilities. Our objectives are threefold: (1) evaluate LLMs' mathematical reasoning, emphasizing mistake detection and correction, (2) identify their strengths and weaknesses in handling different mathematical challenges, and (3) propose potential directions for enhancing their capabilities in this domain. To achieve this comprehensive evaluation, we have developed our own mistake dataset, which includes errors in the reasoning steps. This dataset allows the assessment of models' proficiency not only in providing correct solutions but also in detecting and correcting mistakes within the reasoning steps. We evaluate 12 different foundational models, including large, small and fine-tuned on math, language models on our curated dataset `MWP-MISTAKE`. We are releasing this dataset for further evaluation and benchmarking[1].

Our analysis reveals several key insights into the performance of LLMs on MWPs. Firstly, detecting mistakes, even trivial ones, remains a significant challenge for these models. Secondly, LLMs often derive correct answers despite this difficulty in mistake detection. This can be attributed to data memorization and potential contamination in training datasets, where models may have encountered similar/same problems before. However, the ability to recover from or correct errors in the reasoning process is generally poor across most models. Our contributions to this paper are as follows:

1. We collect and release to the research community `MWP-MISTAKE`, a dataset containing MWPs with both correct and incorrect reasoning obtained from state-of-the-art MWP datasets such as SVAMP (Patel et al., 2021), GSM-8K (Cobbe et al., 2021b), MATH (Hendrycks et al., 2021), MATHBENCH (Liu et al., 2024a), and JEEBENCH (Arora et al., 2023). Incorrect reasoning is derived through meticulously crafted rules to alter the reasoning steps and using smaller models, leveraging their inherent limitations in solving MWPs.
2. We provide benchmark results for our dataset to evaluate the reasoning capabilities of LLMs such as O1 (OpenAI, 2024) GPT-4o (OpenAI, a), GPT-4 (OpenAI et al., 2024), GPT-3.5Turbo (noa), Claude (Anthropic, 2024), as well as smaller language models like Llama (Touvron et al., 2023), Phi (Abdin et al., 2024a), and Mixtral (Jiang et al., 2024) and also models fine-tuned on Math datasets. Our analysis demonstrates that all SOTA LLMs struggle with mistake detection and correction.
3. Through meticulous evaluation and comparison of different LLMs, we offer a detailed analysis of their strengths and weaknesses in handling mathematical reasoning tasks.

2 MWP-Mistake Dataset

Most Math Word Problem (MWP) datasets provide math problems with final answers, occasionally including correct reasoning steps. To evaluate LLMs' ability to detect and correct errors, we created the `MWP-MISTAKE` dataset using five sources: SVAMP (Patel et al., 2021), GSM-8K (Cobbe et al., 2021b), MATH (Hendrycks et al., 2021), MATHBENCH (Liu et al., 2024a), and JEEBENCH (Arora et al., 2023), with MATHBENCH and JEEBENCH being more recent. These five datasets form the basis of the MWP-MISTAKE dataset, covering a wide range of complexities from middle, high school to college levels. While GSM-8K and MATH offer ground truth corect reasoning steps, the others do not. For

[1]Repository for source code and dataset: `https://aka.ms/MWP-MISTAKE`

Table 1: MWP-MISTAKE Dataset details with the total number of questions and reasoning steps.

Dataset	Default reasoning		Smaller model reasoning			
	# Questions with correct reason (GT)	# Questions with incorrect reason (Rules)	# Questions with incorrect reasoning			Total
			Llama-2-7b-chat	Mixtral-8x7B	Phi-3-mini	
SVAMP	154	924	60	20	20	1178
GSM-8K	93	558	100	100	100	951
MATH	150	900	150	150	150	1500
MATHBENCH	100	600	100	100	100	1000
JEEBENCH	38	228	12	19	35	332

those, we used GPT-4 to generate chain-of-thought reasoning steps, which were then extensively manually verified for correctness. The final dataset includes MWP questions, correct reasoning steps, and final answers from all five sources (see Appendix A for additional details). To create incorrect reasoning steps, we propose two approaches: (i) meticulously crafted rules, and (ii) using smaller models as bad reasoners, which we describe next.

2.1 Rules to Programmatically Inject Errors

These rules are motivated and derived from common mistakes observed in educational settings, ensuring the errors introduced are realistic and representative of actual student errors. The six rules are, **(i) Shuffle numerical values:** Swap numerical values to test if the model correctly selects relevant numbers from the question. **(ii) Replace numerical values:** Replace numbers with random values (0–100) to assess if the model correctly extracts numerical information. **(iii) Shuffle operations:** Swap mathematical operators to evaluate the model's ability to apply correct operations. **(iv) Insert random reasoning steps:** Add an unrelated reasoning step to check if the model detects inconsistencies. **(v) Shuffle reasoning steps:** Rearrange steps to introduce ambiguity, testing the model's ability to recognize logical order. **(vi) Delete reasoning steps:** Remove a step in solutions to assess if the model identifies missing reasoning.

These rules replicate real-world student behavior, capturing common mistakes such as misordering steps, skipping steps, misinterpreting numerical values, using incorrect numbers, applying the wrong operations, and adding irrelevant steps. While rules 5 and 6 do not introduce explicit reasoning errors, they are included to help models identify unclear scenarios. Table 1 details the number of questions selected from five datasets, with each question generating seven reasoning variations (one correct + six incorrect). Real-world examples are provided in Appendix A.1.

2.2 Smaller Models as Bad Reasoners

Recently, SLMs are gaining popularity with increased performance, however, they still lack several capabilities, including advanced mathematical reasoning, resulting in poorer performance on MWPs. To curate incorrect reasoning steps, we use SLMs to generate Chain-of-Thought (COT) reasoning and final answers for all dataset questions. Questions with incorrect final answers, identified by comparing them to the ground truth, are retained, and their reasoning steps are classified as incorrect. We then perform an extensive human validation of the answer and reasoning steps using 5 independent volunteers to make sure their is a mistake (as there could few instances where the answer can be incorrect, but reasoning steps could be correct). Each volunteer was provided with clear guidelines on how to assess the correctness of reasoning chains. The dataset was partitioned into five equal subsets, and each volunteer was tasked with labeling each reasoning chain as correct or incorrect. We employ state-of-the-art SLMs, such as Llama-2-7b-chat, Phi-3-mini, and Mixtral-8x7B, to generate COT reasoning steps and Appendix D provides examples of such incorrect reasoning steps with final wrong answer. Table 1 provides statistics for each model across datasets. The entire dataset, including reasoning steps, was exhaustively manually verified to eliminate errors.

Our dataset includes questions with original correct reasoning, rule-based incorrect reasoning, and SLM-generated incorrect reasoning. For evaluation, we split the data into two parts: (1) **Default**, with correct reasoning and rule-based incorrect steps, and (2) **SLM reason**, featuring SLM-generated incorrect reasoning. Table 1 provides the complete details of the curated MWP-MISTAKE dataset with the above two splits.

3 Experimental Setup

Task Details. Our aim is to assess the performance of LLMs on MWPs, focusing on their ability to detect and correct mistakes within the reasoning steps. Below are two task variants to accomplish this are: **(i) Task-1 (T1):** Given a question with reasoning steps, the model must determine correctness, rectify errors if present, and compute the final answer (Figure 1).**(ii) Task-2 (T2):** The model only identifies whether the reasoning steps are correct or incorrect and provides the final answer, without explicitly correcting errors. In essence, T1 evaluates the model's ability to detect mistakes, rectify them, and derive the correct answer, while T2 focuses solely on detecting mistakes-models must identify errors but are not explicitly required to correct them before providing the final answer. Both tasks operate under few-shot settings (prompt details - Appendix E).

Models. To evaluate LLMs' mathematical reasoning capabilities, we utilize foundational LLMs, SLMs and math-finetuned SLMs.

1. **LLMs:** We utilize 6 LLMs that have shown great performance in MWPs such as GPT-4o, GPT-4, GPT-3.5Turbo, Claude-3-Opus, Llama-2-70b (Touvron et al., 2023), Llama-3-70B (Dubey et al., 2024).
2. **SLMs.** Additionally, we evaluate six popular SLMs—Phi-3-mini (Abdin et al., 2024b), Mixtral-8x7B (Jiang et al., 2024), Llama-2-7b-chat (Touvron et al., 2023), Qwen2-7B (Yang et al., 2024), Llama-3-8B (Dubey et al., 2024), and Llama-3-8b-finetuned (Chen and Li, 2024) trained on high-quality data to assess their reasoning capabilities. Additional details of models in Appendix G.

Metrics. For mistake detection, the model outputs either "yes" (indicating correct reasoning) or "no" (indicating incorrect reasoning). The ground truth labels are similarly "yes" for correct reasoning and "no" for incorrect reasoning, and the model's predictions are compared against these labels to calculate the F1 score. This F1 score measures path-level detection performance for the complete reasoning chain. For performance evaluation, the generated final answer is compared to the ground truth final answer to compute the accuracy.

4 Results and Analysis

4.1 OpenAI o1 model performance on MWP-MISTAKE dataset?

OpenAI's O1 model enhances reasoning capabilities by allocating more processing time before responding. Our analysis comparing O1 and GPT-4O on GSM-8K, MATH, MATHBENCH, and JEEBENCH reveals that O1 outperforms GPT-4O in deriving correct final answers, demonstrating superior reasoning across complex tasks (Table 2). This focused comparison between O1 (a "thinking" model) and GPT-4o (a "non-thinking" model) is to assess how enhanced reasoning time impacts mistake detection. For all other models, we maintained consistency by benchmarking against GPT-4o.

Table 2: Performance of o1 vs GPT4o on questions from `MWP-MISTAKE`

		O1	GPT4o	O1	GPT4o
Model	Metric	D	D	SLM	SLM
GSM-8K	Mistake Detection	0.70	0.85	0.84	0.86
	Final Answer	0.96	0.98	0.86	0.84
MATH	Mistake Detection	0.78	0.83	0.94	0.94
	Final Answer	0.91	0.89	0.84	0.79
MATHBENCH	Mistake Detection	0.72	0.80	0.98	0.99
	Final Answer	0.45	0.87	0.38	0.62
JEEBENCH	Mistake Detection	0.80	0.79	0.99	0.99
	Final Answer	0.86	0.38	0.89	0.41

Rule-based mistake identification. Both models perform similarly, with GPT-4O often matching or surpassing O1 in final answer accuracy. However, O1 struggles more with mistake detection, showing an average accuracy drop of 0.0675. Notably, O1 underperforms on MATHBENCH, scoring only 0.45 accuracy compared to GPT-4O's 0.87. Despite this, O1's advanced reasoning abilities are evident in other datasets.

SLM-generated mistakes. Both models achieve high mistake detection performance, especially in complex datasets like JEEBENCH and MATHBENCH. While O1 excels in reasoning and generalization, it struggles to derive the correct final answer in MATHBENCH, falling behind GPT-4O with accuracy of 0.38 and 0.62, respectively.

In summary, while both models perform similarly in mistake identification, O1 excels in final answer generation and rectification, especially for SLM-based mistakes. These findings highlight O1's superior reasoning capabilities, making it more effective for complex tasks. However, challenges like potential data contamination and inefficiencies in processing time and token usage require further optimization.

4.2 Can LLMs Effectively Identify Mistakes in Reasoning Steps?

We evaluate the ability of various models to detect mistakes in the reasoning steps of MWPs, with F1 scores across five datasets, as shown in Table 3 for both default (D) and smaller models (SLM).

GPT-4o Performance. GPT-4o outperforms models like GPT-4, GPT-3.5Turbo, and smaller counterparts, achieving an overall F1 score of 87%. However, it struggles with consistent mistake detection, particularly on simpler datasets like SVAMP and more complex ones like JEEBENCH, where its F1 score drops by 6% compared to GSM-8K.

Table 3: Mistake Detection Performance (F1 score) on MWP-MISTAKE dataset for Task T1. (D-Default reasoning steps, SLM-Smaller model reasoning steps) (Bold: Best)

	GSM8K		MATH		MATHBENCH		JEEBENCH		SVAMP		AVERAGE		
Model	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**Overall**
GPT-4o	**0.85**	**0.86**	**0.83**	**0.94**	**0.80**	**0.99**	**0.79**	**0.99**	0.74	0.92	**0.80**	**0.94**	**0.87**
GPT-4	0.72	0.72	0.78	0.90	0.51	0.90	0.81	0.87	0.61	0.89	0.69	0.86	0.77
GPT-3.5Turbo	0.80	0.70	0.80	0.60	0.50	0.34	0.54	0.46	0.75	0.69	0.68	0.56	0.62
Claude-3-Opus	0.79	0.89	0.73	0.90	0.68	0.92	0.69	0.88	0.77	**0.93**	0.73	0.90	0.82
Qwen2-7B	0.59	0.26	0.64	0.53	0.49	0.67	0.60	0.60	0.53	0.61	0.57	0.53	0.55
Phi-3-mini	0.70	NA	0.65	NA	0.54	NA	0.55	NA	0.70	NA	0.63	NA	0.63
Mixtral-8x7B	0.73	NA	0.79	NA	0.62	NA	0.70	NA	0.64	NA	0.70	NA	0.70
Llama-2-7b-chat	0.07	NA	0.16	NA	0.08	NA	0.36	NA	0.12	NA	0.16	NA	0.16
Llama-2-70b	0.63	0.73	0.77	0.61	0.81	0.98	0.54	0.75	0.71	0.45	0.69	0.70	0.70
Llama-3-8B	0.79	0.81	0.79	0.79	0.56	0.58	0.50	0.67	0.78	0.81	0.68	0.73	0.71
Llama-3-8b-finetuned	0.83	0.82	0.83	0.76	0.77	0.80	0.55	0.74	**0.81**	0.68	0.76	0.76	0.76
Llama-3-70B	0.79	0.74	0.76	0.78	0.55	0.76	0.59	0.61	0.70	0.82	0.68	0.74	0.71

This indicates that while GPT-4o excels in many areas, its precision in mistake detection remains inconsistent across different complexities.

Rule-based vs. SLM-generated Mistakes. GPT-4o and other models detect SLM-generated mistakes more accurately than rule-based ones, achieving an F1 score of 94% versus 80%. This disparity arises because SLM errors propagate across multiple steps, forming identifiable patterns, whereas rule-based errors are isolated to single steps (e.g., number/operator shuffling), making them harder to detect due to their lack of broader contextual cues.

Performance of GPT-4 vs. GPT-3.5Turbo. While GPT-3.5Turbo matches or surpasses GPT-4 on datasets like GSM-8K, it struggles with errors from smaller models. In contrast, GPT-4 handles these mistakes more effectively, possibly due to data contamination or overfitting. For instance, GPT-4 achieves an F1 score of 86% on smaller model-generated mistakes, compared to GPT-3.5Turbo 's 56%.

Smaller Models and Fine-tuning. Fine-tuned smaller models like Llama-3-8b-finetuned perform competitively, achieving an F1 score of 76%, rivaling GPT-4 (77%) in some cases. This underscores the impact of domain-specific fine-tuning, particularly for mathematical tasks, where targeted training enhances mistake detection accuracy.

Challenges with Newer Datasets. All models, including GPT-4o, struggle with newer and more complex datasets like MATHBENCH and JEEBENCH. For instance, GPT-4o 's F1 score drops by 6% on JEEBENCH compared to GSM-8K, highlighting a lack of generalization to unseen problem types. While GPT-4o remains the top performer, its limitations on these datasets emphasize the need for improved reasoning generalization. Appendix H provides a detailed F1 score analysis across different models and rule-based reasoning mistakes.

4.3 Can LLMs Accurately Derive Correct Answers Despite Mistakes?

We evaluate the models' ability to generate correct answers despite errors in reasoning steps. Table 4 reports the accuracy for Task 1, where models must detect and correct mistakes to compute the final answer.

GPT-4o Performance. GPT-4o achieves an overall accuracy of 75%, demonstrating strong answer derivation despite flawed reasoning. It excels in correcting rule-based errors on simpler datasets like GSM-8K (98%), MATH (89%), and MATHBENCH (87%) but drops to 38% on the more complex JEEBENCH dataset. A similar pattern emerges with SLM-generated mistakes—despite detecting them with over 90% accuracy, its correction ability remains inconsistent, with accuracy ranging from 70–80%. This suggests that while GPT-4o can handle simple rule-based errors, often through correction or memorization, it struggles with more intricate SLM-generated mistakes, highlighting limitations in its reasoning abilities.

Performance of Other Models. Similar trends emerge across other models. Claude-3-Opus and GPT-4 rank second and third, respectively, while SLMs like Phi, Llama, and Mixtral perform significantly worse, with accuracy ranging from 40–60%. This highlights the advantage of larger models like GPT-4o in mistake rectification over smaller and fine-tuned models.

Challenges with Complex Datasets. All models, including GPT-4o, struggle with complex datasets like JEEBENCH, where answer deriva-

Table 4: Performance in deriving correct answers (Accuracy) on MWP-MISTAKE dataset for Task T1. (D-Default reasoning steps, SLM-Smaller model reasoning steps) (Bold: Best)

	GSM8K		MATH		MATHBENCH		JEEBENCH		SVAMP		AVERAGE		
Model	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**Overall**
GPT-4o	**0.98**	**0.84**	**0.89**	0.79	0.87	**0.62**	0.38	**0.41**	**1.00**	0.77	**0.82**	**0.69**	**0.75**
GPT-4	0.96	0.73	0.77	0.63	0.87	0.40	0.28	0.23	0.97	0.72	0.77	0.54	0.66
GPT-3.5Turbo	0.85	0.40	0.62	0.20	0.72	0.19	0.18	0.11	0.94	0.35	0.66	0.25	0.46
Claude-3-Opus	0.97	0.86	0.85	**0.90**	**0.90**	0.50	**0.42**	0.32	0.96	**0.80**	0.82	0.68	0.74
Qwen2-7B-Instruct	0.90	0.44	0.76	0.33	0.82	0.25	0.39	0.17	0.85	0.29	0.75	0.29	0.52
Phi-3-mini	0.83	NA	0.40	NA	0.53	NA	0.16	NA	0.68	NA	0.52	NA	0.52
Mixtral-8x7b	0.82	NA	0.60	NA	0.66	NA	0.10	NA	0.86	NA	0.61	NA	0.61
Llama-2-7b-chat	0.72	NA	0.21	NA	0.32	NA	0.03	NA	0.57	NA	0.37	NA	0.37
Llama-2-70b	0.61	0.18	0.36	0.06	0.46	0.06	0.21	0.03	0.69	0.06	0.47	0.08	0.27
Llama-3-8b	0.89	0.34	0.43	0.10	0.74	0.14	0.35	0.09	0.93	0.17	0.67	0.17	0.42
Llama-3-8b-finetuned	0.89	0.34	0.43	0.10	0.74	0.14	0.35	0.09	0.93	0.17	0.67	0.17	0.42
Llama-3-70b	0.80	0.69	0.75	0.46	0.79	0.23	0.36	0.17	0.80	0.69	0.70	0.45	0.57

tion accuracy drops significantly. This highlights a fundamental limitation of current LLMs—their lack of robustness in handling deeper reasoning tasks and intricate problem sets.

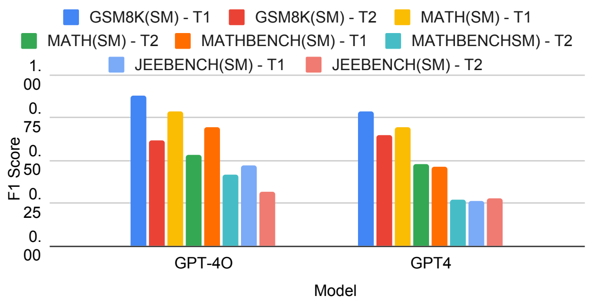


Figure 2: Performance in deriving final answer between T1 and T2. A significant drop in performance occurs when the model does not rectify the incorrect reasoning steps.

Comparing Performance on Task 2: Identifying Mistakes Without Correction.
In Task 2, models must identify the presence of a mistake but are not explicitly required to correct it before providing the final answer. Figure 2 presents the performance of GPT-4o, GPT-4, and GPT-3.5Turbo across all datasets for both Task 1 (detect and rectify mistakes) and Task 2 (identify mistakes and compute the answer without rectification), with detailed results in Appendix F.

GPT-4o's Performance drops noticeably from Task 1 to Task 2 across all datasets. In Task 1, where it is prompted to detect and correct mistakes, it achieves higher accuracy, especially on simpler datasets. However, in Task 2, where it only identifies mistakes, its F1 score declines significantly. This suggests that GPT-4o struggles to rectify errors unless explicitly instructed, highlighting a key weakness in its reasoning.

GPT-4 Performance. GPT-4 follows a similar trend but with a smaller performance drop between Task 1 and Task 2. While its overall accuracy is lower than GPT-4o, it is more consistent in mistake detection. However, like GPT-4o, it struggles to correct errors without explicit prompting, reinforcing its limitations in autonomous reasoning.

4.4 Exploring Data Contamination and Memorization Effects in Math Reasoning Tasks

In our analysis of LLMs' mathematical reasoning performance, we identify data contamination and memorization as key challenges affecting their effectiveness. Data contamination is where test data from downstream tasks appear in training, compromises real-world performance evaluation. Memorization is where models reproduce solutions without understanding underlying principles, limit generalization to novel problems. Signs of contamination emerge in unexpectedly high performance on certain datasets. For instance, GPT-3.5Turbo outperforms GPT-4 on GSM-8K suggests possible biases in GPT-4's training data. Similarly, the comparable performance of smaller and larger models hints at memorization. These findings highlight the need for rigorous evaluation to mitigate memorization and ensure LLMs' reliability in real-world applications.

Investigating data contamination and memorization poses challenges due to restricted pretraining data access and computational limitations. To tackle this, we employ an approach outlined in (Golchin and Surdeanu, 2024), utilizing an LLM to replicate individual instances of the dataset. This involves guiding the LLM with instructions containing unique identifiers from the source dataset, like dataset name, partition (e.g., train, test, or validation), and a fragment of the reference instance. By instructing the LLM to complete these partial instances, we can evaluate contamination and memorization. To detect contamination, a heuristic is applied comparing the average overlap score between generated completions and reference instances using ROUGE-L (Lin, 2004). This comparison is made between guided instructions (including dataset and partition identifiers) and general instructions (lacking such identifiers). If the overlap score is significantly larger with guided instructions, it suggests contamination. This method relies on the premise that the only distinction between the two instructions is the inclusion of dataset and partition names in guided instructions, implying any improvement can be attributed to contamination (Appendix K for more details).

GPT-4o Performance. GPT-4o consistently exhibits the highest ROUGE-L scores across all datasets, suggesting significant contamination. This aligns with its earlier results—excelling in simpler tasks while struggling with more complex ones—indicating a reliance on memorized data rather than genuine reasoning capabilities.

Comparative Contamination Across Models. Both GPT-4 and GPT-3.5Turbo follow similar trends, displaying progressively lower ROUGE-L scores than GPT-4o, yet still indicating some degree of contamination. This aligns with their competitive performance, especially on datasets like GSM-8K, where memorization may have contributed to their high accuracy.

SLMs' Minimal Contamination. SLMs like Llama and Phi show negative ROUGE-L score differences, suggesting minimal or no contamination. These models appear to depend more on reasoning rather than memorization, as their performance does not benefit from exposure to test data. However, their struggles with complex tasks highlight their limitations in advanced reasoning capabilities compared to larger models.

4.5 Can LLMs Correctly Rectify Mistakes in Reasoning Steps?

In Task 1, the model not only detects mistakes but also attempts to rectify them to arrive at the correct answer. We evaluate the model's ability to rectify mistakes once detected by examining the number of questions where mistakes were identified and calculating how many times the model produced the correct answer after rectification. The assumption is that if the model reaches the correct final answer after detecting a mistake, it has successfully rectified the incorrect reasoning step. For instance, if the model identifies mistakes in 90 out of 100 questions and rectifies them in 45 cases (resulting in a final correct answer), the rectification score would be 50% (45/90). Table 5 illustrates the performance of different models in rectifying reasoning steps and producing the correct final answer across various datasets.

GPT-4o demonstrates strong proficiency in mistake rectification, achieving a 78% rectification score, 11% higher than GPT-4 and over 35% above other models, including SLMs. It excels in correcting rule-based reasoning errors but struggles with more complex datasets like MATHBENCH and JEEBENCH. On simpler datasets (GSM-8K, MATH, SVAMP), its high rectification accuracy may stem from data contamination or the simpler nature of rule-based mistakes. As noted earlier, Claude-3-Opus performs on par with GPT-4o in mistake rectification, while other models lag with scores between 30-50%. Notably, Llama-3-70B matches GPT-4, demonstrating strong rectification abilities. To analyze the rectification process, we compute the percentage of cases where the model corrected reasoning steps but still reached incorrect answers. On the `MWP-MISTAKE` dataset, GPT-4o failed in 17% of such cases, while GPT-4, Llama-2-7b-chat, Mixtral-8x7B, and Phi-3-mini had failure rates of 30%, 43.5%, 80.9%, 40.2%, and 55.6%, respectively.

We also compared rectified reasoning steps with ground-truth steps using NLP metrics like BERTScore to assess alignment and effectiveness (details in Appendix I and J). Additionally, we leveraged ReasonEval (Xia et al., 2024), which employs an LLM-based Validity Metric to check for logical and calculation errors. Across all models, rectified reasoning steps scored higher than mistake-detected steps, indicating improved accuracy post-rectification. GPT-4o achieved the highest validity score, showcasing its superior mistake detection and correction abilities (Appendix M). We found that, across all models, the rectified reasoning chains consistently achieve higher validity metric scores compared to the mistake-detected reasoning steps, indicating improved accuracy after rectification. GPT-4o demonstrates the highest validity score, reflecting its ability to detect and rectify mistakes effectively.

4.6 How does model performs on self generated incorrect reasoning steps?

Most SOTA models struggle to detect errors in reasoning steps, but how well do they evaluate their

Table 5: Ability to rectify mistakes and derive the correct final answer on `MWP-MISTAKE` dataset for Task T1. (D-Default reasoning steps, SLM-Smaller model reasoning steps) (Bold: Best)

	GSM8K		MATH		MATHBENCH		JEEBENCH		SVAMP		AVERAGE		
Model	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**D**	**SLM**	**Overall**
GPT-4o	**0.98**	**0.91**	**0.87**	**0.84**	**0.90**	**0.64**	0.42	**0.42**	**1.00**	**0.86**	**0.83**	**0.73**	**0.78**
GPT-4	0.96	0.88	0.72	0.70	0.83	0.45	0.10	0.24	0.94	0.77	0.71	0.61	0.66
GPT-3.5Turbo	0.81	0.56	0.54	0.34	0.62	0.34	0.16	0.05	0.93	0.57	0.61	0.37	0.49
Claude-3-Opus	0.97	0.94	0.84	0.89	0.87	0.57	0.27	0.33	0.96	0.85	0.78	0.72	0.75
Qwen2-7B-Instruct	0.83	0.51	0.77	0.47	0.69	0.29	0.29	0.21	0.78	0.50	0.67	0.40	0.53
Phi-3-mini	0.79	NA	0.37	NA	0.41	NA	0.03	NA	0.63	NA	0.45	NA	0.45
Mixtral-8x7b	0.77	NA	0.56	NA	0.57	NA	0.17	NA	0.83	NA	0.58	NA	0.58
Llama-2-7b-chat	0.73	NA	0.21	NA	0.11	NA	0.04	NA	0.52	NA	0.32	NA	0.32
Llama-2-70b	0.57	0.25	0.34	0.07	0.46	0.06	0.02	0.03	0.60	0.21	0.40	0.12	0.26
Llama-3-8b	0.77	0.51	0.39	0.24	0.58	0.08	**0.49**	0.06	0.65	0.39	0.58	0.26	0.42
Llama-3-8b-finetuned	0.85	0.41	0.33	0.10	0.69	0.18	0.25	0.13	0.91	0.26	0.60	0.21	0.41
Llama-3-70b	0.80	0.88	0.72	0.62	0.73	0.33	0.21	0.21	0.83	0.81	0.66	0.57	0.61

own reasoning? To investigate, we analyzed LLMs' ability to detect and rectify mistakes in their self-generated reasoning steps. We selected questions where the corresponding model produced incorrect final answers, using these faulty reasoning chains to assess mistake detection, rectification, and correctness. Our findings show that while models like GPT-4 and GPT-4o perform similarly in detecting mistakes in both self-generated and SLM-generated reasoning, their rectification ability drops significantly for self-generated errors. For instance, on the MATH dataset, GPT-4 maintains mistake detection performance but rectification drops to 53% compared to 70% for SLM-generated mistakes. Further details are in Appendix L.

5 Related Work

Current research on large language models (LLMs) for solving math word problems (MWPs) primarily prioritizes answer accuracy over evaluating reasoning processes. Studies like MathPrompter (Imani et al., 2023) and WizardMath (Luo et al., 2023) demonstrate strong performance by generating complex reasoning steps but focus mainly on correctness rather than verifying the validity and relevance of each step. Similarly, works such as (Liu et al., 2024b; Yuan et al., 2023; Schulman et al., 2017) enhance LLMs' accuracy but do not assess whether their reasoning aligns with logical problem-solving paths. Recent studies have begun shifting focus toward reasoning quality, but their scope remains limited. Works like (Sawada et al., 2023) assess reasoning by comparing generated and reference reasoning, while (Xia et al., 2024) introduces metrics like validity and redundancy to evaluate reasoning steps. ROSCOE(Golovneva et al., 2023) expands on this by providing unsupervised metrics for assessing semantic consistency and logicality beyond just the final answer. However, these approaches often overlook the detection and correction of specific reasoning mistakes in MWPs, leaving a gap in understanding how well LLMs handle flawed reasoning.

Another gap is limited exploration of LLMs' core reasoning abilities, particularly in mistake detection and rectification. While some studies propose LLMs as verifiers of their own reasoning (Zhang et al., 2024b; Zheng et al., 2023), they primarily assess correctness without addressing the deeper challenge of identifying and correcting logical mistakes. (Olausson et al., 2024) highlights LLMs' struggles in recognizing and fixing their own reasoning errors, especially in code generation tasks. (Zeng et al., 2023) shifts evaluation focus toward scoring solution correctness rather than just question answering. Work like Alice in Wonderland (Nezhurina et al., 2024) examines LLM reasoning, showing that small variations in common-sense tasks can drastically impact performance. However, rigorous benchmarking for mistake detection and correction remains unexplored.

Our work bridges this gap by introducing the MWP-Mistake dataset, which systematically incorporates diverse reasoning mistakes in MWPs. Unlike prior studies, our analysis evaluates not only models' ability to detect errors but also their capacity to correct them and generate accurate solutions. We present a comprehensive benchmark, comparing SOTA models across both simple and complex datasets. This work highlights current models' limitations in handling reasoning mistakes and establishes a framework for assessing LLMs' true reasoning abilities beyond mere answer accuracy.

6 Conclusions

This study evaluates large language models (LLMs) such as O1, GPT-4o, GPT-4, GPT-3.5Turbo, alongside smaller models like Llama-2-7b-chat, Mixtral-8x7B, and Phi-3-mini, on their ability to detect and correct errors in mathematical reasoning.

Using our MWP-MISTAKE dataset, which includes incorrect reasoning steps generated through both rule-based methods and smaller models, we comprehensively assess LLMs' performance in error detection and rectification. While GPT-4o outperforms other models, there remains a gap in its ability to consistently detect mistakes, as it struggles with several simple problems and its performance degrades on more complex tasks. We also uncover issues of data contamination and overfitting, especially in GPT-4's performance on GSM8K, and observe a performance drop on newer datasets like MATHBENCH and JEEBENCH, highlighting generalization challenges. Addressing these limitations, such as enhancing generalization and minimizing data contamination, is essential for making LLMs more reliable and applicable to real-world mathematical problem-solving. Future research should focus on refining training processes and strengthening models' reasoning abilities to meet these challenges.

7 Limitations

While our study provides a comprehensive evaluation of LLMs' ability to detect and correct mistakes in mathematical reasoning, certain limitations warrant further exploration. First, our evaluation focuses on a curated dataset designed to systematically assess mistake detection and rectification. Although this dataset is diverse, real-world problem distributions may introduce additional complexities not fully captured in our benchmark. Expanding the dataset to cover a broader spectrum of reasoning errors across various mathematical domains is a potential avenue for future work.

Second, our analysis primarily evaluates models under few-shot prompting settings. While this aligns with practical deployment scenarios, alternative techniques such as fine-tuning or reinforcement learning may further enhance model performance. Investigating such methods is beyond the scope of this study, but remains an important direction.

Finally, while we discuss potential issues related to data contamination and memorization, precise quantification of their impact remains an open challenge. A more detailed study into the extent of memorization effects across different models and training paradigms would provide deeper insights into their reasoning capabilities.

Furthermore, our perturbation design was informed by graduate-level educators, ensuring alignment with common student mistakes. However, an LLM can exhibit more sophisticated errors. Expanding our perturbation framework to capture higher-order reasoning failures and domain-specific complexities is an important direction for future research.

Despite these limitations, our findings offer critical insights into LLMs' reasoning abilities, revealing fundamental challenges that must be addressed to improve robustness in mathematical problem-solving.

8 Ethical Considerations

Our study acknowledges the ethical considerations associated with evaluating LLMs' ability to detect and correct mistakes in mathematical reasoning.

First, while our dataset is curated to systematically assess reasoning capabilities, there is a possibility that biases in data selection may influence model performance. We have taken steps to ensure diversity in problem types and difficulty levels, but future work should explore broader, real-world datasets to mitigate potential biases.

Second, as LLMs are increasingly integrated into educational tools, there is a risk that users may over-rely on them without critically evaluating their outputs. Our analysis highlights limitations in mistake detection and correction, underscoring the need for responsible deployment, where models are used as assistive tools rather than definitive sources of truth. Finally, potential data contamination in LLM training poses concerns about fairness and reproducibility. While we discuss these issues and assess their impact, further transparency from model providers regarding training data sources is necessary to ensure ethical AI development.

References

OpenAI Platform.

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Qin Cai, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Yen-Chun Chen, Yi-Ling Chen, Parul Chopra, Xiyang Dai, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Victor Fragoso, Dan Iter, Mei Gao, Min Gao, Jianfeng Gao, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie

Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Ce Liu, Mengchen Liu, Weishung Liu, Eric Lin, Zeqi Lin, Chong Luo, Piyush Madan, Matt Mazzola, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Xin Wang, Lijuan Wang, Chunyu Wang, Yu Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Haiping Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Sonali Yadav, Fan Yang, Jianwei Yang, Ziyi Yang, Yifan Yang, Donghan Yu, Lu Yuan, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024a. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024b. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Anthropic. 2024. Introducing the next generation of Claude \ Anthropic.

Daman Arora, Himanshu Gaurav Singh, and Mausam. 2023. Have llms advanced enough? a challenging problem solving benchmark for large language models. *Preprint*, arXiv:2305.15074.

Wei Chen and Zhiyuan Li. 2024. Octopus v4: Graph of language models. *arXiv preprint arXiv:2404.19296*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021a. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021b. Training Verifiers to Solve Math Word Problems. *arXiv preprint*. ArXiv:2110.14168 [cs].

Aniruddha Deb, Neeva Oza, Sarthak Singla, Dinesh Khandelwal, Dinesh Garg, and Parag Singla. 2023. Fill in the blank: Exploring and enhancing llm capabilities for backward reasoning in math word problems. *Preprint*, arXiv:2310.01991.

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*.

Wensheng Gan, Zhenlian Qi, Jiayang Wu, and Jerry Chun-Wei Lin. 2023. Large language models in education: Vision and opportunities. *Preprint*, arXiv:2311.13160.

Shahriar Golchin and Mihai Surdeanu. 2024. Time travel in llms: Tracing data contamination in large language models. *Preprint*, arXiv:2308.08493.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Roscoe: A suite of metrics for scoring step-by-step reasoning. *Preprint*, arXiv:2212.07919.

Joy He-Yueya, Gabriel Poesia, Rose E. Wang, and Noah D. Goodman. 2023. Solving math word problems by combining language models with symbolic solvers. *Preprint*, arXiv:2304.09102.

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. *Preprint*, arXiv:2103.03874.

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

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

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.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024a. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. *Preprint*, arXiv:2405.12209.

Wentao Liu, Hanglei Hu, Jie Zhou, Yuyang Ding, Junsong Li, Jiayi Zeng, Mengliang He, Qin Chen, Bo Jiang, Aimin Zhou, and Liang He. 2024b. Mathematical language models: A survey. *Preprint*, arXiv:2312.07622.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. WizardMath: Empowering Mathematical Reasoning for Large Language Models via Reinforced Evol-Instruct. *arXiv preprint*. ArXiv:2308.09583 [cs].

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.

Marianna Nezhurina, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. 2024. Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models. *Preprint*, arXiv:2406.02061.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2024. Is Self-Repair a Silver Bullet for Code Generation? *arXiv preprint*. ArXiv:2306.09896 [cs].

OpenAI. a. Hello GPT-4o.

OpenAI. b. Introducing ChatGPT.

OpenAI. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, and Sam Altman. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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

Tomohiro Sawada, Daniel Paleka, Alexander Havrilla, Pranav Tadepalli, Paula Vidas, Alexander Kranias, John J. Nay, Kshitij Gupta, and Aran Komatsuzaki. 2023. ARB: Advanced Reasoning Benchmark for Large Language Models. *arXiv preprint*. ArXiv:2307.13692 [cs].

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

KV Aditya Srivatsa and Ekaterina Kochmar. 2024. What makes math word problems challenging for llms? *Preprint*, arXiv:2403.11369.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, and Jiahui Yu. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2024. Evaluating mathematical reasoning beyond accuracy. *arXiv preprint arXiv:2404.05692*.

Xin Xu, Tong Xiao, Zitong Chao, Zhenya Huang, Can Yang, and Yang Wang. 2024. Can llms solve longer math word problems better? *Preprint*, arXiv:2405.14804.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. *arXiv preprint*. ArXiv:2309.12284 [cs].

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *Preprint*, arXiv:2308.01825.

Zhongshen Zeng, Pengguang Chen, Shu Liu, Haiyun Jiang, and Jiaya Jia. 2023. Mr-gsm8k: a metareasoning benchmark for large language model evaluation. *arXiv preprint arXiv:2312.17080*.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, Sean Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. 2024a. A careful examination of large language model performance on grade school arithmetic. *Preprint*, arXiv:2405.00332.

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. 2024b. Small language models need strong verifiers to self-correct reasoning. *Preprint*, arXiv:2404.17140.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv preprint*. ArXiv:2306.05685 [cs].

Appendix

The dataset and code to run all experiments are provided in this repository.

A MWP-MISTAKE Dataset

MWP-MISTAKE dataset is curated using 4 different types of well-known datasets. Below are the details of each of the datasets.

- SVAMP (Patel et al., 2021): SVAMP is a MWP dataset created by applying a carefully chosen variation over examples sampled from existing datasets, AsDiv-A (Miao et al., 2020) and MAWPS (Koncel-Kedziorski et al., 2016).
- GSM-8K (Cobbe et al., 2021b):GSM-8K is a dataset of diverse grade school math word problems created by human writers, involving basic arithmetic operations. Released in November 2021.
- MATH (Hendrycks et al., 2021): The MATH dataset is divided into seven categories, each with five difficulty levels. For our study, we used levels 1, 2, and 3 from the algebra and counting and probability categories. Released in November 2021. We focused on Levels 1 to 3 because the problems in Levels 4 and 5 are more complex, requiring specific notations, symbols, and equations. Injecting reasoning mistakes into such complex problems is non-trivial and would require expert knowledge to ensure accuracy in the reasoning chains.
- MATHBENCH (Liu et al., 2024a): MATHBENCH is a recent dataset with questions divided by educational stages, from basic arithmetic to college levels. For our experiment, we chose middle and high-school-level single-choice and multiple-choice questions. Released in May 2024.
- JEEBENCH (Arora et al., 2023): JEEBENCH is a challenging benchmark dataset for evaluating LLM problem-solving abilities, containing 515 pre-engineering math, physics, and chemistry problems from the IIT JEE-Advanced Exam. For our experiment, we chose mathematics single-choice questions only. Released in October 2023.

A.1 Rule based Examples

The rules we have applied are grounded in discussion with grad-school educators or who provided insights into common mathematical reasoning errors made by students. Furthermore, prior research systematically analyses student mistakes, revealing key patterns such as data misinterpretation, logical fallacies, and incomplete proofs. Our heuristics were carefully designed to replicate these real-world errors, aligning with both expert knowledge and empirical findings. While some rules may appear trivial at first glance, they capture authentic patterns of human errors that arise during problem-solving. Below, we provide examples to clarify the real-world applicability of these rules:

- **Shuffle Numerical Values:** A student might misread a problem or inadvertently swap numbers, such as treating "5 apples and 3 oranges" as "3 apples and 5 oranges." This is a typical misstep when transcribing or interpreting numerical data.
- **Replace Numerical Values:** This error mimics cases where students substitute incorrect values, e.g., replacing "radius = 7" with "radius = 5" when solving for the area of a circle. This often occurs due to lapses in focus or misreading.
- **Shuffle Operations:** A common reasoning error is misunderstanding the order of operations, such as treating "$2 + 3 \times 4$" as "$(2 + 3) \times 4$." This reflects students' struggles with applying mathematical precedence correctly.
- **Insert Random Reasoning:** Students sometimes include irrelevant steps or overcomplicate their work, such as writing "Because the train moves faster than the car, $2 \times 5 = 10$." These distractions, while not inherently erroneous, disrupt logical flow and mirror overthinking.
- **Shuffle Reasoning Steps:** The sequence of reasoning is critical in problem-solving. For instance, solving for the area of a triangle by calculating "base × height" before dividing by 2 is correct, but reversing this order introduces confusion and can mislead students. Such missteps highlight the importance of logical step sequencing.
- **Delete Reasoning Step:** Omitting steps often reflects missed applications of formulas or logical progression, e.g., skipping the application of the Pythagorean theorem when solving for the hypotenuse in a right triangle. While this omission might not always lead

to an incorrect answer, it risks undermining clarity and correctness in reasoning.

These transformations do not solely aim to generate incorrect answers; they evaluate the model's robustness in navigating incomplete, shuffled, or flawed reasoning to derive the correct solution. Particularly, "shuffle reasoning" and "delete reasoning" mimic challenges faced by students, such as disrupted clarity or skipped steps, emphasizing the importance of logical step-by-step progression. Thus, the rules are carefully designed to reflect genuine reasoning errors while testing models' ability to detect, rectify, and reason coherently. Examples of MWPs with correct, rule-based, incorrect and smaller model-based incorrect reasoning steps are present in Figure 3

A.2 Prompts to curate reasoning steps in `MWP-MISTAKE` dataset

GSM-8K and MATH already contain MWP questions, a chain of thought reasoning steps and a final answer. To curate a chain of thought reasoning step for MATHBENCH and JEEBENCH, we made use of GPT-4. While prompting GPT-4 we made sure that reasoning steps did not contain the final answer, so that the final answer is not picked directly from the reasoning step. Listing 1 prompt is used to curate the reasoning steps.

Listing 1: Prompt to curate reasoning chain without answers.

```
Strictly follow the conditions \
below.
1. Output format: \nReasoning \
Chain: \nFinal Answer:
2. Reasoning Chain should be \
separated by a new line only.
3. A reasoning chain cannot have\
 the final answer. (Replace the \
final answer  in the reasoning \
chain with its calculation or \
####)
4. Do not include any additional\
 information in the final answer\
 (only the answer).
```

Table 6 shows examples of default reasoning steps from GSM-8K dataset.

B Mistake Detection Analysis with Simple MWPs

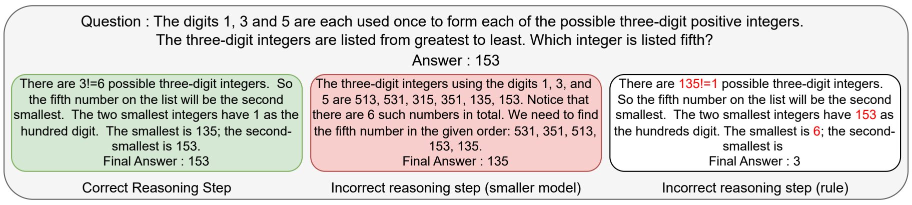
We evaluated the models' ability to detect reasoning mistakes using the SVAMP dataset, which contains simple arithmetic word problems (up to a 4th-grade level) and variations testing question sensitivity, reasoning ability, and structural invariance (Appendix C for more details on the variations included). Mistakes were introduced using both rule-based methods and outputs from SLMs, with human validation ensuring accuracy.

Table 7 shows presents the models' mistake detection performance across these variations. The results show that none of the models consistently detected mistakes, with F1 scores across all variations falling below 80%. The highest F1 score, 81%, was achieved by Llama-3-8b-finetuned, a fine-tuned model specifically trained on 13 math-related datasets, which outperformed even more advanced models like GPT-4o and GPT-4. This suggests that fine-tuning on domain-specific data offers significant benefits for mathematical tasks.

Despite these improvements, even the fine-tuned model showed significant sensitivity to problem variations. When question sensitivity variations were introduced, performance dropped by 0.08, while reasoning ability and structural invariance variations resulted in reductions of 0.06 and 0.02, respectively. GPT-4o exhibited a similar performance decline, suggesting that even the most advanced models are vulnerable to small variations in problem structure (See Table 7, Appendix C).

These findings highlight a key gap: even on relatively simple problems, models fail to generalize when minor variations are introduced. This suggests that fine-tuning, while beneficial, is insufficient to fully address the deeper issues in mistake detection across mathematical reasoning tasks. More robust methods are needed to improve generalization.

Comparison between JEEBENCH and SVAMP: Comparing SVAMP with JEEBENCH, mistake detection for SLM-generated errors in JEEBENCH appears better because the incorrect reasoning chains were generated by smaller models that perform poorly on such complex tasks. These low-quality chains are easier for advanced models like GPT-4o to identify as flawed. On SVAMP, however, the reasoning chains are of higher quality, and errors tend to be localized and nuanced. Detecting such subtle errors is more challenging,


Question : The digits 1, 3 and 5 are each used once to form each of the possible three-digit positive integers. The three-digit integers are listed from greatest to least. Which integer is listed fifth?
Answer : 153

There are 3!=6 possible three-digit integers. So the fifth number on the list will be the second smallest. The two smallest integers have 1 as the hundred digit. The smallest is 135; the second-smallest is 153.
Final Answer : 153

The three-digit integers using the digits 1, 3, and 5 are 513, 531, 315, 351, 135, 153. Notice that there are 6 such numbers in total. We need to find the fifth number in the given order: 531, 351, 513, 153, 135.
Final Answer : 135

There are 135!=1 possible three-digit integers. So the fifth number on the list will be the second smallest. The two smallest integers have 153 as the hundreds digit. The smallest is 6; the second-smallest is
Final Answer : 3

Correct Reasoning Step
Incorrect reasoning step (smaller model)
Incorrect reasoning step (rule)

Figure 3: Examples of MWPs with correct reasoning, rule-based incorrect and smaller model-based incorrect reasoning from MATH.

Table 6: Example of rule-based incorrect reasoning step (GSM-8K dataset)

Question	Gerald spends $100 a month on baseball supplies. His season is 4 months long. He wants to use the months he's not playing baseball to save up by raking, shovelling, and mowing lawns. He charges $10 for each. How many chores does he need to average a month to save up for his supplies?
Final Answer	5
Gold Reasoning step	He needs to save up $400 because 4 x 100 = 400 He has 8 months to earn this money because 12 - 4 = 8 He needs to earn $50 a month because 400 / 8 = 50 He needs to do 5 tasks a month because 50 / 10 = 5
Shuffle the reasoning step	He needs to earn $50 a month because 400 / 8 = 50 He needs to save up $400 because 4 x 100 = 400 He needs to do 5 tasks a month because 50 / 10 = 5 He has 8 months to earn this money because 12 - 4 = 8
Delete the reasoning step	He needs to save up $400 because 4 x 100 = 400 He needs to earn $50 a month because 400 / 8 = 50 He needs to do 5 tasks a month because 50 / 10 = 5
Shuffle numerical values	He needs to save up $400 because 4 x 100 = 400 He has 50 months to earn this money because 8 - 8 = 4 He needs to earn $12 a month because 400 / 8 = 50 He needs to do 5 tasks a month because 50 / 10 = 5
Replace numerical values	He needs to save up $400 because 4 x 100 = 400 He has 8 months to earn this money because 12 - 4 = 8 He needs to earn $6 a month because 32 / 8 = 50 He needs to do 76 tasks a month because 50 / 10 = 5
Shuffle Operations	He needs to save up $400 because 4 x 100 = 400 He has 8 months to earn this money because 12 * 4 = 8 He needs to earn $50 a month because 400 - 8 = 50 He needs to do 5 tasks a month because 50 / 10 = 5
Insert Random Reasoning step	He needs to save up $400 because 4 x 100 = 400 Therefore, Faye has $60 - $30 = $30 left. He has 8 months to earn this money because 12 - 4 = 8 He needs to earn $50 a month because 400 / 8 = 50 He needs to do 5 tasks a month because 50 / 10 = 5

Table 7: Mistake Detection Performance (F1 score) on SVAMP dataset with all variations

	Question Sensitivity			**Reasoning Ability**			**Structural invariance**		
Model	**Same Object, Different Structure**	**Different Object, Same Structure**	**Different Object, Different Structure**	**Add relevant information**	**Change Information**	**Invert Operation**	**Change order of objects**	**Change order of phrases**	**Add irrelevant information**
GPT-4o	0.78	0.70	**0.77**	0.74	0.78	0.72	0.74	0.75	0.76
GPT-4	0.65	0.58	0.64	0.62	0.66	0.55	0.62	0.60	0.64
GPT-3.5Turbo	0.76	0.77	0.71	0.75	0.77	0.74	0.78	0.72	0.75
Llama-2-7b-chat	0.14	0.09	0.13	0.10	0.17	0.08	0.13	0.15	0.13
Phi-3-mini	0.77	0.64	0.68	0.72	0.73	0.67	0.69	0.68	0.69
Qwen2-7B-Instruct	0.57	0.42	0.62	0.54	0.59	0.46	0.61	0.51	0.59
Llama-3-8b	0.80	0.79	0.79	0.82	0.79	0.75	0.81	0.75	0.78
Llama-3-70b	0.73	0.69	0.73	0.70	0.73	0.69	0.70	0.70	0.73
Llama-3-8b-finetuned	**0.82**	**0.79**	0.76	**0.81**	**0.83**	**0.81**	**0.79**	**0.79**	**0.86**

leading to reduced mistake detection performance despite the dataset's simpler nature. Moreover, due to the complexity of JEEBENCH models exhibit a high rate of false positives, flagging reasoning chains as incorrect even when they are accurate. For example, GPT-4o frequently misidentifies correct reasoning chains as flawed in JEEBENCH. This tendency inflates error detection rates on JEEBENCH but reflects the model's limited ability to accurately discern reasoning quality rather than genuine improvement in mistake detection.

C SVAMP Variations

Figure 4 shows the 9 different types of carefully curated variations sampled from existing datasets, AsDiv-A (Miao et al., 2020) and MAWPS (Koncel-Kedziorski et al., 2016). Across each category, we evaluated the mistake detection performance of each model. Table 7 shows that Llama-3-8b-finetuned performed the best due to this preexisting knowledge on solving MWPs, achieving the highest average F1 score of 81% across the variations. We also evaluated the model's sensitivity to variations, table 8 shows the max performance change across different models. GPT-4o performance significantly dropped by 0.08, 0.06, and 0.02 when exposed to variations related to question sensitivity, reasoning ability, and structural invariance, respectively. Examples of variations are as follows:

- Reasoning ability - Change Information: This involves changing entities, e.g., replacing "Jack" with "Dorothy."

- Reasoning ability - Invert Operation: Here, operations or calculations are altered while keeping the rest of the structure the same.

- Structural Invariance - Change Order of Objects: This variation reverses the order of entities, such as changing "8 marbles and 3 stones" to "3 stones and 8 marbles.

Across all these simple variations, we observed a performance drop of 10% in GPT-4 and around 6% in GPT-4o, highlighting the sensitivity of these models. Interestingly, fine-tuned models like Llama-3-8B-Finetuned demonstrated greater robustness, with just a 2% performance drop.

D SLMs reasoning steps

SLMs were used to generate a chain of thought (COT) reasoning step and final answers for all dataset questions. Each model Llama-2-7b-chat, Mixtral-8x7B, Phi-3-mini where prompted using Listing 1 to curate the reasoning step without an answer. If the final answer was incorrect, we filtered out the reasoning steps as incorrect. Table 9 shows examples of SLM incorrect reasoning steps from GSM-8K dataset.

E Task T1 and T2

Task T1 evaluates the model's ability to detect mistakes, rectify them and derive the correct answer. Listing 2 was used in a few-shot setting for task T1.

Listing 2: Prompt for Task T1

```
You are a mathematics educator \
with a deep understanding of \
elementary and middle school \
mathematics. You are experienced\
 in teaching multi-step problem-\
solving techniques and have a \
knack for breaking down complex \
problems into manageable steps. \
Your expertise lies in basic \
arithmetic operations such as \
addition, subtraction, \
multiplication, and division. \
You can provide clear, step-by-\
step solutions to mathematical \
problems that require multi-step\
 reasoning.

You are provided with a \
mathematical question and a step\
-by-step solution along with it.\
 The solution might have some \
mistakes. Identify if the \
solution is correct or incorrect\
. If the solution is correct, \
output the final answer with the\
 help of the solution provided. \
If the solution is incorrect, \
correct the existing solution \
and determine the final answer \
with the help of the corrected \
solution.
Reasoning chain Correct (Yes/No)\
:
Corrected reasoning chain or NA:
Final answer (just the number):
```

Task T2 evaluates the model's ability to detect mistakes and solve MWP based on the provided reasoning step. Listing 3 was used in a few-shot setting for task T2. Here, we ensure that the final answer is generated with the help of the reasoning steps provided, which may or may not be correct.

Listing 3: Prompt for Task T2

```
You are a mathematics educator \
with a deep understanding of \
elementary and middle school \
mathematics. You are experienced\
 in teaching multi-step problem-\
solving techniques and have a \
knack for breaking down complex \
problems into manageable steps. \
Your expertise lies in basic \
arithmetic operations such as \
addition, subtraction, \
multiplication, and division. \
You can provide clear, step-by-\
step solutions to mathematical \
problems that require multi-step\
 reasoning.

You are provided with a \
mathematical question and a step\
-by-step solution along with it.\
 The solution might have some \
mistakes. Identify if the \
solution is correct or incorrect\
 and output the final answer \
based on the provided solution.
```

CATEGORY	VARIATION	EXAMPLES
Question Sensitivity	Same Object, Different Structure	**Original:** Allan brought two balloons and Jake brought four balloons to the park. How many balloons did Allan and Jake have in the park? **Variation:** Allan brought two balloons and Jake brought four balloons to the park. How many more balloons did Jake have than Allan in the park?
	Different Object, Same Structure	**Original:** In a school, there are 542 girls and 387 boys. 290 more boys joined the school. How many pupils are in the school? **Variation:** In a school, there are 542 girls and 387 boys. 290 more boys joined the school. How many boys are in the school?
	Different Object, Different Structure	**Original:** He then went to see the oranges being harvested. He found out that they harvest 83 sacks per day and that each sack contains 12 oranges. How many sacks of oranges will they have after 6 days of harvest? **Variation:** He then went to see the oranges being harvested. He found out that they harvest 83 sacks per day and that each sack contains 12 oranges. How many oranges do they harvest per day?
Reasoning Ability	Add relevant information	**Original:** Every day, Ryan spends 4 hours on learning English and 3 hours on learning Chinese. How many hours does he spend on learning English and Chinese in all? **Variation:** Every day, Ryan spends 4 hours on learning English and 3 hours on learning Chinese. If he learns for 3 days, how many hours does he spend on learning English and Chinese in all?
	Change Information	**Original:** Jack had 142 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Jack have now? **Variation:** Dorothy had 142 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Dorothy have now?
	Invert Operation	**Original:** He also made some juice from fresh oranges. If he used 2 oranges per glass of juice and he made 6 glasses of juice, how many oranges did he use? **Variation:** He also made some juice from fresh oranges. If he used 2 oranges per glass of juice and he used up 12 oranges, how many glasses of juice did he make?
Structural Invariance	Change order of objects	**Original:** John has 8 marbles and 3 stones. How many more marbles than stones does he have? **Variation:** John has 3 stones and 8 marbles. How many more marbles than stones does he have?
	Change order of phrases	**Original:** Matthew had 27 crackers. If Matthew gave equal numbers of crackers to his 9 friends, how many crackers did each person eat? **Variation:** Matthew gave equal numbers of crackers to his 9 friends. If Matthew had a total of 27 crackers initially, how many crackers did each person eat?
	Add irrelevant information	**Original:** Jack had 142 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Jack have now? **Variation:** Jack had 142 pencils. Dorothy had 50 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Jack have now?

Figure 4: Variations in SVAMP dataset on simple Math Problems (Patel et al., 2021)

Table 8: Max Performance change with introduction of variations on SVAMP dataset.)

Model	Question Sensitivity	Reasoning Ability	Structural invariance
GPT-4o	0.08	0.06	0.02
GPT-4	0.07	0.04	0.04
GPT-3.5Turbo	0.06	0.03	0.06
Llama-2-7b-chat	0.05	0.09	0.02
Phi-3-mini	0.09	0.06	0.01
Qwen2-7B-Instruct	0.20	0.13	0.1
Llama-3-8b	0.01	0.07	0.06
Llama-3-70b	0.04	0.04	0.03
Llama-3-8b-finetuned	0.06	0.02	0.07

```
Reasoning chain Correct (Yes/No)\
:
Final answer (just the number):
```

F T2 Results

Table 10 provides the results for Task T2 (Performance in deriving the correct final answer)

G Model Used

Below are brief details of the models we have used for benchmarking our `MWP-MISTAKE` dataset.

1. **GPT-4o:** GPT-4o is a multimodal model by OpenAI, and it has the same high intelligence as GPT-4 Turbo but is much more efficient—it generates text 2x faster and is 50% cheaper. Additionally, GPT-4o has the best vision and performance across non-English languages of

Table 9: Example of incorrect reasoning steps generated using SLM's (GSM-8K dataset)

Question	Gerald spends \$100 a month on baseball supplies. His season is 4 months long. He wants to use the months he's not playing baseball to save up by raking, shovelling, and mowing lawns. He charges \$10 for each. How many chores does he need to average a month to save up for his supplies?
Final Answer	5
Gold Reasoning step	He needs to save up \$400 because 4 x 100 = 400. He has 8 months to earn this money because 12 - 4 = 8 He needs to earn \$50 a month because 400 / 8 = 50 He needs to do 5 tasks a month because 50 / 10 = 5 #### 5
Llama-2-7b-chat Incorrect reasoning step	Gerald spends \$100 on baseball supplies per month. Season is 4 months long. To save up for baseball supplies, Gerald needs to earn more than the cost of supplies per month. Gerald charges \$10 for each chore he does. To calculate the number of chores Gerald needs to do per month, divide the cost of supplies by the amount he charges per chore.
Llama-2-7b-chat Predicted Answer	""
Mixtral-8x7B Incorrect reasoning step	Gerald spends \$100 a month for 4 months, so his total spending for the season is 100*4 = \$400 Now, we need to find out how many chores he needs to do to earn this amount He charges \$10 per chore, so we divide the total amount by the cost per chore: 400 / 10 = ####
Mixtral-8x7B Predicted Answer	40
Phi-3-mini Incorrect reasoning step	Gerald's total expenditure for baseball supplies = \$100 per month * 4 months = \$400 Gerald earns \$10 per chore. To find out how many chores he needs to do to save up \$400, we divide his total expenditure by the amount he earns per chore.= \$400 / \$10= ####
Phi-3-mini Predicted Answer	40

	GSM8K		**MATH**		**MATHBENCH**		**JEEBENCH**		**AVERAGE**		
Model	D	SM	D	SM	D	SM	D	SM	D	SM	Overall
GPT-4o	0.92	0.61	0.66	0.52	0.77	0.39	0.22	0.29	0.64	0.45	0.55
GPT-4	0.99	0.63	0.69	0.46	0.82	0.26	0.38	0.30	0.72	0.42	0.57
GPT-3.5Turbo	0.80	0.23	0.60	0.29	0.66	0.16	0.44	0.18	0.62	0.22	0.42
Llama-2-7b-chat	0.81	NA	0.30	NA	0.44	NA	0.27	NA	0.46	NA	0.46
Mixtral-8x7b	0.87	NA	0.58	NA	0.65	NA	0.06	NA	0.54	NA	0.54
Phi-3-mini	0.88	NA	0.56	NA	0.65	NA	0.47	NA	0.64	NA	0.64

Table 10: Performance in deriving the correct final answer in Task T2 setting.

any OpenAI model. Last training data: October 2023.

2. **GPT-4:** GPT-4 is a large multimodal model by OpenAI that can solve difficult problems with greater accuracy than any of OpenAI's previous models, thanks to its broader general knowledge and advanced reasoning capabilities. Last training data: September 2021.

3. **GPT-3.5Turbo:** GPT-3.5Turbo is a large language model by OpenAI GPT-3.5 that can understand and generate natural language or code and has been optimized for chat using the Chat Completions API, but works well for non-chat tasks as well. Last training date: September 2021.

4. **Claude-3-Opus:** Claude-3-Opus is Anthropic's most capable and intelligent model yet, ideal for navigating complex tasks like in-depth analysis, research, and task automation. Last training data: August 2023.

5. **Llama-2-7b-chat:** Llama 2 is a collection of pretrained and fine-tuned generative text models ranging in scale from 7 billion to 70 billion parameters from Meta. This is the 7B fine-tuned model, optimized for dialogue use cases. Training date: September 2022.

6. **Mixtral-8x7B:** Mixtral is a Mixture of Experts (MoE) model with 8 experts per MLP, with a total of 45 billion parameters. Despite the model having 45 billion parameters, the compute required for a single forward pass is the same as that of a 14 billion parameter model. This is because even though each of the experts have to be loaded in RAM (70B like ram requirement), each token from the hidden states are dispatched twice (top 2 routing) and thus the compute (the operation required at each forward computation) is just 2 X sequence_length.

7. **Phi-3-mini:** The Phi-3-Mini-128K-Instruct is a 3.8 billion-parameter by microsoft, lightweight, state-of-the-art open model trained using the Phi-3 datasets. This dataset includes both synthetic data and filtered publicly available website data, with an emphasis on high-quality and reasoning-dense properties. Last training data: October 2023.

H Categories Wise results

Table 11 shows the F1 score analysis on different types of Rule-based reasoning mistakes on GSM-8K dataset. Furthermore Figure 5, 6, 7 and 8 shows the GPT-4o Mistake detection and Performance F1 score on different type of rule based and SLM based mistakes on GSM-8K, MATH, MATHBENCH andJEEBENCH respectively. Looking at category-wise results, we see that:

- Reasoning chains without actual mistakes are also constantly flagged as erroneous by the model.
- The model particularly struggles with identifying errors in shuffle reasoning steps and delete reasoning steps. We believe these are critical errors in the reasoning chain that significantly impact logical coherence and should not be overlooked.

I METEOR and BertScore results

BertScore computes a similarity score for each token in the candidate sentence with each token in the reference sentence using the BERT embeddings. Metric for Evaluation of Translation with Explicit Ordering (METEOR) score is a metric that measures the quality of generated text based on the alignment between the generated text and the reference text. The metric is based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision.

Table 12 and Table 13 present the BertScore and Meteor Score, respectively, for all the datasets across all models. We observed that these two metric evaluations where not fully able to capture the nuance capabilities of LLMs in rectifying the mistakes within reasoning steps. This can be seen in the results. GPT-4o has a consistently high performance across all the datasets, but when you compare the BERTScore between the corrected reasoning step and ground truth reasoning step, you see the rest of the models clearly performing better than GPT-4o. GPT-4 has performed better than GPT-3.5Turbo in most datasets.

J Average Reasoning Step Length

We noticed that the average word length of rectified reasoning for correct and incorrect for GPT-4o was higher than other models. Table 14 presents the average word length of the rectified reasoning step for all datasets across the models.

K Data contamination and Memorization

Two different types of prompts were used to test out memorization of individual instances of the dataset. The reasoning step was split into two halves, and models were prompted in two different ways (general and guided) to complete the second half of the reasoning step. To detect contamination, we evaluated the overlap score between generated and reference instances using the ROUGE-L Score. Listing 4 and Listing 5 show the prompt used for guided and generation instructions ,respectively.

Listing 4: Guided instruction for dataset GSM8K

```
You are provided with a full \
question and the first answer \
piece from the test split of the\
 GSM8K dataset. Finish the \
second answer piece of the \
instance as exactly appeared in \
the dataset. Only rely on the \
original form of the instance in\
 the dataset to finish the \
second answer piece. Output only\
 the second answer piece.
```

Listing 5: General instruction for dataset GSM8K

```
Based on the provided question, \
finish the second answer piece \
based on the first answer piece,\
 such that these two pieces \
become a single instance answer.\
 Output only the second answer \
piece.
```

Here GSM-8K and test are the extra information provided for the model to uniquely identify instances from the source dataset and complete the reasoning step.

Figure 9 presents the variation in ROUGE-L scores between guided and general instructions across different datasets and models. The observed discrepancies provide strong initial evidence of data contamination, particularly in larger models. Table 15 presents the complete result for the average ROUGE-L score of guided and general for all datasets across all models. Our evaluation results suggest that the performance gains of GPT-4o are indeed influenced by data contamination. For instance, on datasets like GSM-8K and MATH, the

Table 11: F1 Score Analysis on Different Types of Rule-Based Reasoning Mistakes on GSM8k Dataset.

Model	Correct Reasoning	Shuffle Reasoning	Delete Reasoning	Shuffle Numerical	Replace Numerical	Shuffle Operations	Random Reasoning	SLM Combined
GPT-4o	0.69	0.84	0.87	0.92	0.96	0.93	0.67	0.73
GPT-4	0.95	0.38	0.54	0.85	0.89	0.72	0.33	0.52
GPT-3.5Turbo	0.83	0.65	0.71	0.82	0.87	0.78	0.76	0.52
Llama-2-7b-chat	1.00	0.00	0.01	0.00	0.02	0.00	0.09	NA
Mixtral-8x7b	0.83	0.59	0.60	0.77	0.75	0.63	0.63	NA
Phi-3-mini	0.85	0.72	0.42	0.60	0.52	0.58	0.82	NA
Claude-3-Opus	0.94	0.51	0.71	0.84	0.94	0.79	0.54	0.76
Qwen2-7B-Instruct	0.95	0.45	0.36	0.53	0.45	0.34	0.62	0.13
Llama-2-70b	0.85	0.55	0.49	0.45	0.44	0.41	0.78	0.55
Llama-3-8b	0.77	0.76	0.62	0.68	0.82	0.63	0.98	0.68
Llama-3-70b	0.75	0.52	0.60	0.83	0.93	0.87	0.84	0.54
Llama-3-8b-finetuned	0.77	0.91	0.80	0.77	0.75	0.65	0.99	0.68

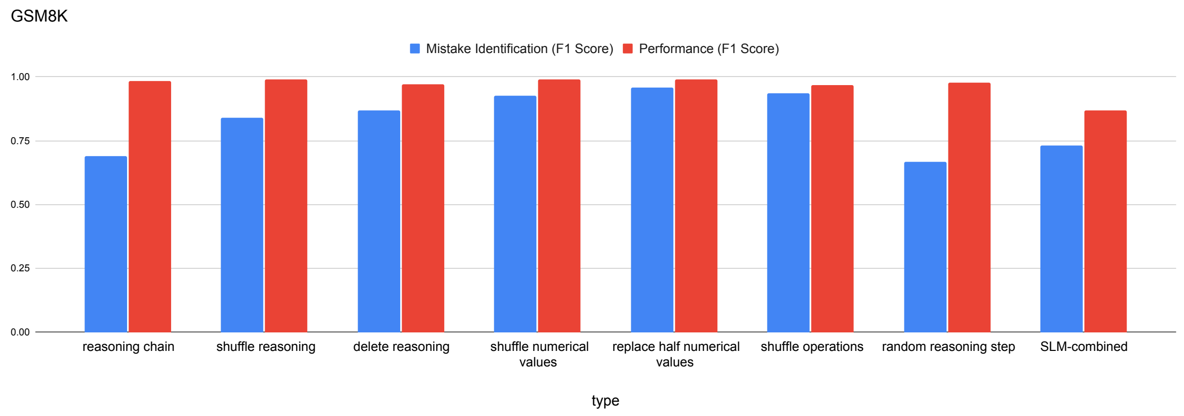


Figure 5: Category Wise mistake detection and performance results on GSM-8K dataset.

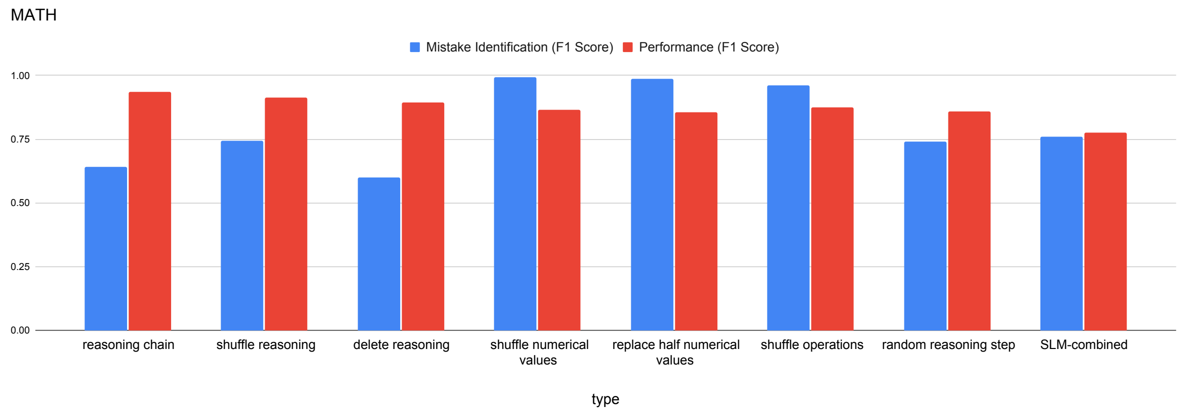


Figure 6: Category Wise mistake detection and performance results on MATH dataset.

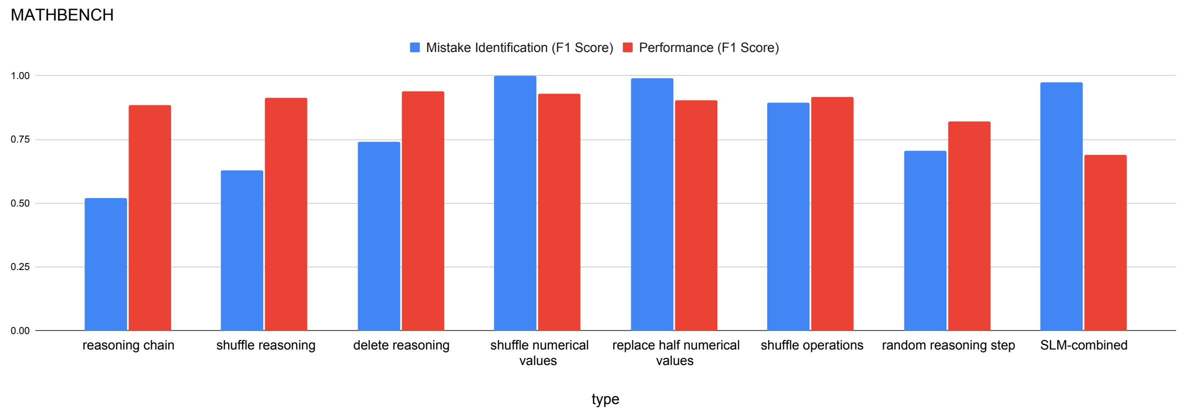


Figure 7: Category Wise mistake detection and performance results on MATHBENCH dataset.

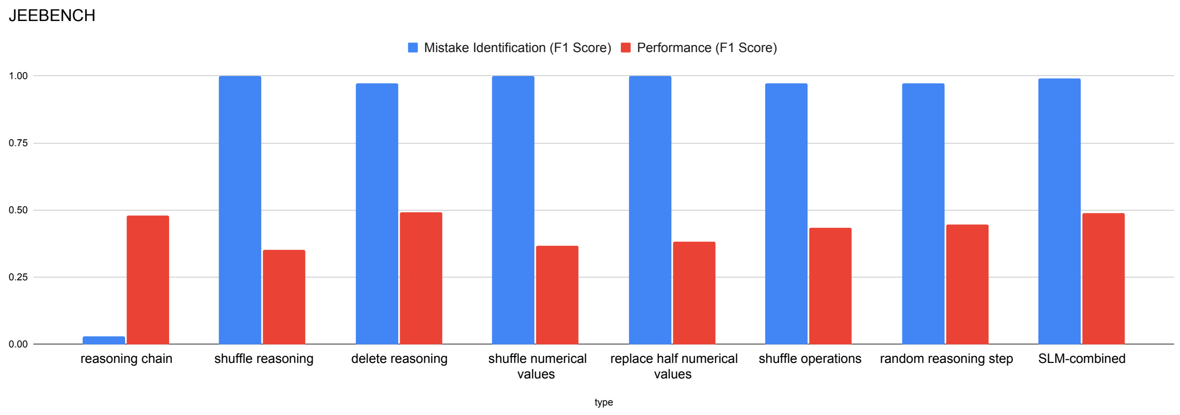


Figure 8: Category Wise mistake detection and performance results on JEEBENCH dataset.

Table 12: BERTscores for correct and incorrect final answers derived after mistake rectification across all models and datasets.

Datasets	Models	GPT-4o		GPT-4		GPT-3.5Turbo		Llama-2-7b-chat		Mixtral-8x7B		Phi-3-mini	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
GSM-8K	D	0.95	0.91	0.98	0.93	0.97	0.95	0.96	0.98	0.97	0.94	0.94	0.91
	SM	0.83	0.82	0.84	0.82	0.84	0.82	NA	NA	NA	NA	NA	NA
MATH	D	0.88	0.90	0.96	0.93	0.95	0.93	0.96	0.88	0.95	0.92	0.90	0.87
	SM	0.84	0.80	0.83	0.81	0.84	0.81	NA	NA	NA	NA	NA	NA
MATHBENCH	D	0.88	0.83	0.97	0.95	0.97	0.94	0.90	0.89	0.96	0.95	0.93	0.90
	SM	0.82	0.82	0.85	0.82	0.84	0.83	NA	NA	NA	NA	NA	NA
JEEBENCH	D	0.89	0.89	0.88	0.87	0.94	0.95	0.86	0.82	0.85	0.87	0.70	0.85
	SM	0.86	0.87	0.85	0.86	0.78	0.86	NA	NA	NA	NA	NA	NA

Table 13: Meteor Score for correct and incorrect final answers derived after mistake rectification across all models and datasets.

Datasets	Models	GPT-4o		GPT-4		GPT-3.5Turbo		Llama-2-7b-chat		Mixtral-8x7B		Phi-3-mini	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
GSM-8K	D	0.81	0.54	0.92	0.62	0.88	0.77	0.87	0.83	0.85	0.74	0.77	0.66
	SM	0.33	0.27	0.37	0.31	0.37	0.32	NA	NA	NA	NA	NA	NA
MATH	D	0.48	0.54	0.76	0.70	0.76	0.67	0.78	0.59	0.73	0.66	0.55	0.48
	SM	0.32	0.28	0.30	0.26	0.33	0.28	NA	NA	NA	NA	NA	NA
MATHBENCH	D	0.55	0.35	0.82	0.63	0.82	0.68	0.49	0.57	0.81	0.68	0.67	0.53
	SM	0.33	0.30	0.32	0.25	0.32	0.29	NA	NA	NA	NA	NA	NA
JEEBENCH	D	0.37	0.31	0.30	0.22	0.49	0.54	0.15	0.13	0.53	0.46	0.20	0.25
	SM	0.28	0.26	0.21	0.21	0.08	0.25	NA	NA	NA	NA	NA	NA

Table 14: Average length of rectified reasoning steps on `MWP-MISTAKE` dataset

	GSM-8K		MATH		MATHBENCH		JEEBENCH		Average		
Model	D	SM	D	SM	D	SM	D	SM	D	SM	Overall
GPT-4o	100.14	131.47	147.50	182.69	312.11	323.45	647.66	619.09	301.85	314.18	308.01
GPT-4	66.59	122.24	79.32	121.59	146.54	140.43	356.71	322.53	162.29	176.69	169.49
GPT-3.5Turbo	66.58	126.30	94.17	124.56	140.50	177.36	670.34	338.53	242.90	191.69	217.29
Llama-2-7b-chat	44.73	NA	113.35	NA	177.67	NA	137.05	NA	118.20	NA	118.20
Mixtral-8x7B	63.04	NA	88.26	NA	140.57	NA	402.79	NA	173.67	NA	173.67
Phi-3-mini	84.92	NA	115.10	NA	172.57	NA	293.90	NA	166.62	NA	166.62
Claude-3-Opus	62.18	138.91	70.60	134.05	144.85	192.84	561.88	438.44	209.88	226.06	217.97

ROUGE-L scores are notably high, corresponding to strong performance in mistake detection and rectification. However, on newer datasets such as JEEBENCH and MATHBENCH, where contamination scores are much lower, GPT-4o's performance in detecting and rectifying mistakes drops significantly. This highlights the impact of contamination on the model's performance.

Qualitative Analysis of Data Contamination

Here we provide qualitative examples to compare these LLMs in the context of data contamination experiments.

The input to the model is the question along with

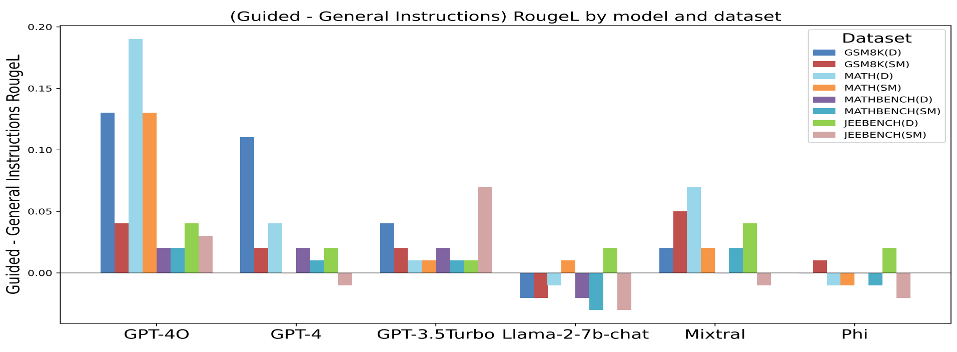


Figure 9: Difference between guided and general instructions rouge-L score across all datasets. A high positive difference indicates high contamination, and a low positive or negative difference indicates little to no contamination.

the first half of the reasoning chain. The model is tasked to reproduce the second half of the reasoning chain based on both general and guided instructions. By computing the ROUGE-L scores for these two instruction types, we measure the overlap to evaluate contamination.

We show examples from GPT4o and Llama-3-8b-finetuned, and we can clearly see in the GPT4o case that the model is able to replicate the second part of the reasoning chain exactly as the ground truth, indicating the contamination.

These qualitative comparisons highlight the extent of contamination in different models, reinforcing the quantitative trends in the paper. We will include these examples in the revised version for clarity.

Example: GPT4o on gsm8k

Input to the model:

- Question: 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?
- First half of reasoning chain (provided to the model): He sprints 3*3=«3*3=9»9 times So he r

Ground truth: Second half of the reasoning chain: uns 9*60=«9*60=540»540 meters ### 540

Guided reasoning chain generated: uns 60*9=«60*9=540»540 meters in a week.

General reasoning chain generated: uns 60 meters each sprint, so he runs a total of 9*60=540 meters each week. Therefore, James runs 540 meters a week by doing 3 sprints 3 times a week at 60 meters each sprint.

Rouge L score:

(i) Guided R_L = Rouge L between Guided and GT (0.79)

(ii) General R_L = Rouge L between General and GT (0.30)

(iii) Guided R_L – General R_L = 0.49

Finetuned llama8B Evaluation

Guided Output: uns 60 meters each time, which means he runs a total of 9 * 60 = «9*60=540»540 meters a week.

General Output: James runs a total of 3*60=«3*60=180»180 meters per sprint. Since he sprints 9 times a week, he runs a total of 9*180=«9*180=1620»1620 meters a week.

Rouge L score:

(i) Guided R_L: 0.53

(ii) General R_L: 0.22

(iii) Guided R_L – General_R_L = 0.30

L Model performance on self-generated reasoning steps

We evaluated the model's detection, rectification and final answer performance on their self generated error reasoning steps. For instance, using GPT-4 on the MATH dataset with 100 incorrect self-generated reasoning steps, we observed the following results for self-generated incorrect reasoning compared to SLM-generated reasoning:

- Mistake Identification: 0.914 (self-generated) vs. 0.90 (SLM-generated)
- Final Answer Accuracy: 0.471 (self-generated) vs. 0.65 (SLM-generated)
- Rectification Performance: 0.533 (self-generated) vs. 0.70 (SLM-generated)

Similarly, used GPT-4O on the MATH dataset with 100 incorrect self-generated reasoning steps, and we observed the following results:

Table 15: Rouge L score between guided and general instructions on MWP-MISTAKE dataset

Datasets	Models	GPT-4o		GPT-4		GPT-3.5Turbo		Llama-2-7b-chat		Mixtral-8x7B		Phi-3-mini		Llama-3-8b-finetuned	
		Guided	General	Guided	General	Guided	General	Guided	General	Guided	General	Guided	General	Guided	General
GSM-8K	D	0.57	0.44	**0.67**	0.56	**0.53**	0.49	0.26	0.28	0.46	0.44	0.32	0.32	0.47	0.49
	SM	0.55	0.51	0.57	0.55	0.49	0.47	0.30	0.32	0.55	0.50	0.42	0.41	0.51	0.46
MATH	D	0.44	0.25	0.52	0.48	0.39	0.38	0.25	0.26	0.39	0.32	0.26	0.27	0.39	0.35
	SM	0.51	0.38	0.54	0.54	0.45	0.44	0.30	0.29	0.48	0.46	0.38	0.39	0.48	0.46
MATHBENCH	D	0.43	0.41	0.48	0.46	0.38	0.36	0.26	0.28	0.36	0.36	0.30	0.30	0.38	0.37
	SM	0.40	0.38	0.43	0.42	0.39	0.38	0.30	0.33	0.40	0.38	0.29	0.30	0.43	0.40
JEEBENCH	D	0.43	0.39	0.42	0.40	0.34	0.33	0.27	0.25	0.38	0.34	0.33	0.31	0.37	0.34
	SM	0.32	0.29	0.34	0.35	0.31	0.24	0.22	0.25	0.26	0.27	0.20	0.22	0.34	0.30

Model	Validity Mistake-detected Reasoning Step	Validity Rectified Reasoning Step	Differene in Validity (Rectified - Detected)
GPT-4o	0.57	0.74	0.18
GPT-4	0.55	0.67	0.13
GPT-3.5Turbo	0.57	0.66	0.09
Llama-2-7b-chat	0.44	0.48	0.04
Mixtral-8x7b	0.42	0.57	0.15
Phi-3-mini	0.44	0.60	0.16
Claude-3-Opus	0.56	0.74	0.17
Qwen2-7B-Instruct	0.58	0.69	0.11
Llama-2-70b	0.58	0.63	0.05
Llama-3-8b	0.57	0.63	0.06
Llama-3-70b	0.56	0.69	0.13
Llama-3-8b-finetuned	0.57	0.61	0.04

Table 16: GSM-8K results on ReasonEval's validity metric

- Mistake Identification: 0.758 (self-generated) vs. 0.94 (SLM-generated)
- Final Answer Accuracy: 0.265 (self-generated) vs. 0.79 (SLM-generated)
- Rectification Performance: 0.255 (self-generated) vs. 0.84 (SLM-generated)

These findings suggest that while models can effectively identify mistakes in their own reasoning, the challenge lies in rectifying these errors and producing accurate final answers. This discrepancy underscores the limitations of LLMs in self-evaluation, particularly in more complex scenarios.

M Evaluating Rectified Reasoning Steps

We used the ReasonEval (Xia et al., 2024) framework to evaluate the rectified reasoning steps generated by our models. In Table 16, we provide results for the GSM-8K dataset, comparing the validity metric scores for:

- The mistake detected reasoning steps (reasoning steps where the model detected there was an error).
- The rectified reasoning steps (where the model rectified the above error reasoning steps).
- The difference between the mistake detected and rectified reasoning chains.

N Running Experiment Multiple Times

While running experiments on all models (LLMs and SLMs) we used the default hyperparameters to generate tokens. We ran a subset of the dataset on different prompt variations and saw comparable performance for various prompts. Due to the limitation of the API key, we were only able to run GPT-4o model on the GSM-8K dataset. On rerun we got very similar results, with an error rate of <= 0.01.

O Insights for math tutoring applications

- **False Positives in Reasoning Evaluation:** The model often flags reasoning chains that are actually correct as erroneous. This means that while using LLMs as tutors, educators should be cautious about relying solely on the model's error detection. It's essential to incorporate human oversight or secondary verification methods to avoid mistakenly discouraging students with incorrect feedback.
- **Effective Identification of Calculator Errors:** The model demonstrates strong performance in pinpointing calculator-related mistakes, such as arithmetic errors or incorrect operations. This capability can be valuable in automated tutoring systems where quick identification and correction of basic computational mistakes are needed.
- **Challenges with Detecting Incomplete Solutions:** The model struggles to identify issues in cases where reasoning steps are missing (e.g., shuffled or deleted steps in the chain of thought). Educators should be aware that while the model may excel at catching overt computational errors, it might not reliably flag gaps in the logical progression of a solution. This limitation suggests that LLMs should be supplemented with methods that ensure comprehensive, step-by-step reasoning is maintained.