

# Derailer-Rerailer: Adaptive Verification for Efficient and Reliable Language Model Reasoning

Guangya Wan<sup>1\*</sup>, Yuqi Wu<sup>2\*</sup>, Hao Wang<sup>3</sup>, Shengming Zhao<sup>2</sup>, Jie Chen<sup>2†</sup>, Sheng Li<sup>1†</sup>

<sup>1</sup>School of Data Science, University of Virginia

<sup>2</sup>Department of Electrical and Computer Engineering, University of Alberta

<sup>3</sup>Beaconfire Solution Inc.

{wxr9et, shengli}@virginia.edu, {yuqi14, jc65, shengmi1}@ualberta.ca, haowang229@gmail.com

## Abstract

Large Language Models (LLMs) have shown impressive reasoning capabilities, yet existing prompting methods face a critical trade-off: simple approaches often struggle with complex tasks and reasoning stability, while more sophisticated methods require multiple inferences and substantial computational resources, limiting their practical deployment. To address this challenge, we propose **Derailer-Rerailer**, a novel framework that adaptively balances reasoning accuracy and computational efficiency. At its core, our framework employs a lightweight **Derailer** mechanism to assess reasoning stability and selectively triggers an advanced **Rerailer** verification process only when necessary, thereby optimizing computational resource usage. Extensive evaluation across both open and closed-source models on more than 20 categories of mathematical, symbolic, and commonsense reasoning tasks demonstrates our framework’s effectiveness: **Derailer-Rerailer** achieves significant accuracy improvements (8-11% across various reasoning tasks) while maintaining 2-3 times better efficiency than existing verification methods, with particularly strong performance in mathematical and symbolic reasoning, offering a practical solution for enhancing LLM reasoning reliability while significantly reducing computational overhead<sup>1</sup>.

## 1 Introduction

Large language models (LLMs) have been augmented with various prompting techniques to induce human-like complex reasoning capabilities (Wei et al., 2022a). While early approaches like Chain-of-Thought aimed to emulate fast thinking for direct step-by-step reasoning (Wei et al., 2022b),

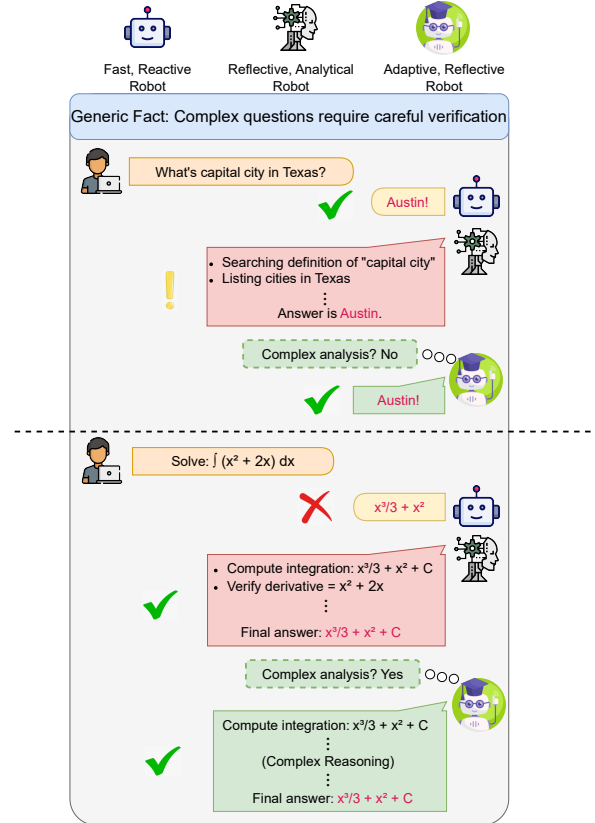


Figure 1: While simple questions can be answered quickly with minimal reasoning, complex tasks require deeper analysis and verification to ensure accuracy. Adaptive approaches balance efficiency and reasoning depth based on the problem’s complexity.

\*These authors contributed equally to this work.

†Corresponding author.

<sup>1</sup>Code available at [https://github.com/wan19990901/CoT\\_rerailer](https://github.com/wan19990901/CoT_rerailer)

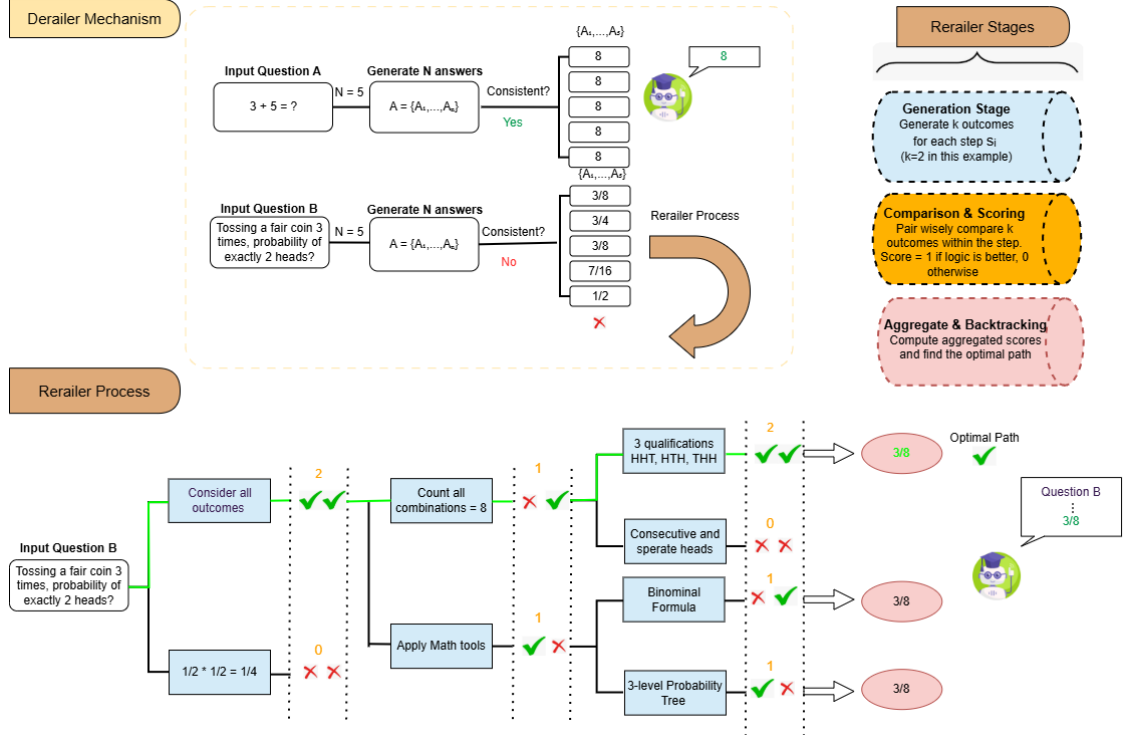


Figure 2: Overview of the Derailer-Rerailer framework: The Derailer filters questions based on reasoning stability, while the Rerailer optimizes reasoning paths for inconsistent cases. Arithmetic examples illustrate the workflow.

cognitive processes where we transition from quick intuition to careful analysis when facing complex challenges (Kahneman, 2011).

These complex prompting methods, however, face a critical trade-off between accuracy and computational efficiency. Such approaches dramatically increase computational overhead. For example, in Self-Consistency (Wang et al., 2023), a single problem requires 40 model calls and results in thousands of tokens (Li et al., 2024). While this computational intensity proves necessary for some complex situations, a key limitation lies in their uniform application—indiscriminately deploying these expensive procedures across all queries regardless of whether the problem actually requires such intensive operations. This computational burden becomes particularly problematic in real-world agentic applications (Huang et al., 2024) that often involve various complex prompting methods, where inference time is as crucial (Snell et al., 2024a) as accuracy (e.g. real-time clinical support agentic systems, which require both reliable reasoning and low latency (Umerenkov et al., 2023; Wu et al., 2025a)). These constraints highlight the need for an adaptive prompting method that can selectively apply computational resources where they are most needed.

To address these limitations, we propose a novel

two-stage framework, *Derailer-Rerailer* (Fig. 2), inspired by the relationship between LLMs’ stability on answers and problem solvability. Drawing an analogy from train operations—where derailment assessment precedes strategic rerailment—our framework combines two complementary mechanisms: (1) The **Derailer** performs full consistency checks through multiple independent answers, efficiently identifying which queries require intervention; and (2) the **Rerailer** applies targeted correction techniques only to cases where inconsistencies are detected. This selective approach ensures that expensive verification methods are deployed only when necessary, simultaneously improving accuracy and preserving computational efficiency.

We have conducted experiments across multiple LLMs on 7 reasoning benchmarks covering more than 20 categories to demonstrate the effectiveness of our framework. The result shows that our efficient framework outperforms various prompting methods up to 10% in accuracy and by more than 50% in efficiency metrics. The framework also shows robust performance across different reasoning types, with the most substantial improvements observed in mathematical and symbolic reasoning tasks. In summary, our contributions are as follows:

- **Derailer Mechanism:** We introduce an effi-

Table 1: Comparison of prompting strategies that leverage both fast, reactive and slow, reflective reasoning modes in LLMs. Immediate prompting provides low QA improvement with low token usage and inference time. Whereas Iterative prompting yields higher QA improvement, but at the cost of higher token usage and inference time. Our goal in this paper is to combine the best of both.

Category	Example Methods	QA Perf.	Token Usage	Inference Time
Immediate Prompting	<i>Chain of Thought, Least-to-Most</i>	Low	Low	Low
Iterative Prompting	<i>Self-Consistency, Chain-of-Verification, Tree of Thought, <b>Rerailer</b></i>	High	High	High
Adaptive Iterative Prompting (Ours)	<i>Derailer + {Self-Consistency, Chain-of-Verification, Tree of Thought, <b>Rerailer</b>}</i>	High	Medium	Medium

cient approach for detecting reasoning instability, enabling targeted application of complex prompting techniques.

- **Rerailer Verification:** We develop a novel prompting method that enhances reasoning stability while preserving accuracy and efficiency.
- **Practical Insights:** We provide comprehensive empirical evidence on the efficiency-accuracy trade-offs in various LLM prompting strategies, offering guidelines for deploying LLMs in resource-constrained environments.

## 2 Motivations and Preliminary Findings

### 2.1 Emulating Human Cognitive Systems in Language Model Prompting

Human cognition, as described by Kahneman’s dual-process theory (Gronchi and Giovannelli, 2018), operates through two distinct systems: System 1, which provides quick, intuitive responses, and System 2, which enables slower, more deliberate reasoning. While System 1 excels at routine tasks through its efficiency, System 2’s complex approach proves invaluable for problems requiring careful analysis. This cognitive framework provides a valuable lens for understanding and improving language model reasoning, where different prompting techniques can be viewed as analogs to these two systems. In this paper, we will define the categories of prompting, aiming to balance the benefits of both fast and deliberate reasoning.

**Immediate Prompting:** This represents the simplest form of interaction with a language model - a single-pass generation process. While it can still generate reasoning steps, the key characteristic is that it produces the answer in one forward pass:

$$y = \mathcal{M}(x, p)$$

where  $p$  encapsulates the prompting strategy (such as Chain of Thought (Wei et al., 2022b), Least to Most (Zhou et al., 2023)). Even when the output contains intermediate reasoning steps  $s_1, \dots, s_n$ , they flow in a single, uninterrupted sequence:

$$s_1, \dots, s_n, y = \mathcal{M}(x, p)$$

**Iterative Prompting:** Another category is iterative prompting which embraces a more deliberate approach. It either generates *parallel* independent samples (like Self-Consistency (Wang et al., 2023) or Best of N sampling (OpenAI, 2022)) and use some sort of aggregation (Wan et al., 2024) or selection (Gu et al., 2025) mechanism to obtain the final answer, or *sequentially* evaluates and refines the reasoning path based on some evaluators (like Chain of Verification (Dhuliawala et al., 2023) or Self-Refine (Madaan et al., 2023)). Formally, this can be defined as:

$$y_i = \mathcal{M}(x, p_i) \quad \text{for } i = 1, \dots, k$$

$$y^* = V(y_1, \dots, y_k, x)$$

where  $V$  represents reward functions that leverage additional model calls to refine the initial solutions or explore other independent solution paths. With a better reasoning performance, the computational cost of iterative prompting increases significantly scaled by both the number of attempts  $k$  and the cost of verification  $V$ .

The fact that human cognition does not necessitate deliberate processing for all decisions suggests a parallel for language models: computationally intensive iterative prompting may not always be advantageous. This motivates the central question: **Under what conditions is the investment in additional computation for deliberate reasoning (Iterated Prompting) warranted?**

Our proposed answer lies in analyzing the stability of the model’s reasoning process. We argue

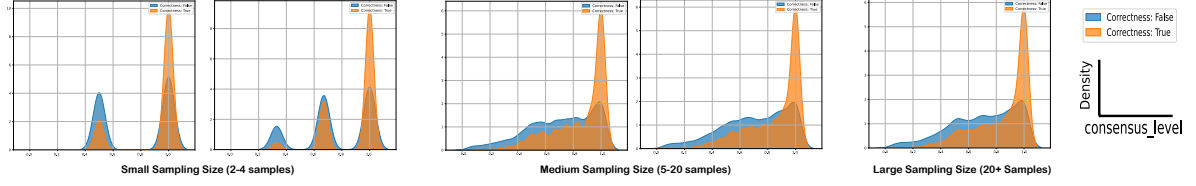


Figure 3: Impact of Sampling Size on Model Performance Consistency. The distributions show consensus levels for *Solvable* questions (orange) and *Insolvable* questions (blue). Distinct patterns on high consensus peaks for *Consistently Solvable* questions and a notable peak for *Consistently Insolvable* questions were observed, indicating that models maintain consistency both when questions are within and fundamentally beyond their capabilities. Besides, as sampling size increases (medium: 5-20, large: 20+), these patterns stabilize, suggesting that minimal sampling is sufficient to assess whether a question falls into our three-way classification.

that the variability in LLM performance—ranging from high confidence to complete uncertainty, and including instances of knowledge without consistent application—suggests three distinct categories: **Stable Success**: When the model demonstrates consistent success with parallel sampling, usually in some single question like "what's the result of  $1 + 2$ ", engaging in additional refinement thus becomes computationally inefficient. **Stable Failure**: When the model consistently fails regardless of the approach, indicating a fundamental knowledge or reasoning gap that additional computation cannot bridge. **Unstable Reasoning**: When the model shows potential but lacks consistency, producing a mix of successful and failed attempts. These cases represent ideal candidates for a system 2 style of thinking, where multiple attempts can help stabilize the reasoning process.

## 2.2 Empirical Validation

To empirically validate the relationship between answer consistency and reasoning stability, we conducted a controlled study across 3 LLMs (Claude-3.5-sonnet, Llama3.1-70B, GPT-4o-mini) using datasets from various reasoning benchmarks including tasks where models typically exhibit stable reasoning (GSM8K, CommonsenseQA, etc.) and those that naturally lead to unstable outputs (such as TruthfulQA, GPQA-Diamond, etc.).

As shown in Fig. 3, this analysis reveals two key patterns. First, regarding sampling efficiency, our experiments show that small samples already provide clear distinctions in consensus levels (the proportion of answers belonging to the majority class), with these patterns stabilizing at medium sampling sizes and larger samples offering diminishing returns. This aligns with recent findings (Wan et al., 2024) on the effectiveness of reduced sampling in self-consistency approaches, which suggests a

small set of samples determines the solvability of question. Second, we observe distinct peaks in both distributions: while questions with correct reasoning (orange) show high stability as expected, we also find a notable peak for cases of *incorrect reasoning* (blue) where questions consistently exceed the model’s capabilities. This suggests that sampling just a few examples can effectively identify a question’s stability category, thus motivating the design of an adaptive prompting method that adjusts computational investment based on reasoning stability, as suggested in Table 1.

## 3 Methodology

### 3.1 Derailer: A Mechanism to Filter Stabilized Reasoning

Building upon the above motivation and analysis, we propose Derailer, a selective gatekeeper that identifies and filters out cases with stable reasoning (whether consistently correct or incorrect) where expensive iterative prompting would provide little benefit. As shown in Fig. 1, Derailer implements this filtering through a lightweight consistency check with  $n$  samples. Only when samples yield inconsistent answers, indicating unstable reasoning patterns, does the question proceed to iterative prompting for deeper analysis. This filtering mechanism naturally aligns with our stability-based classification: cases with stable reasoning (both correct and incorrect) are filtered out to avoid unnecessary computation, while cases exhibiting reasoning instability receive additional verification and refinement.

The effectiveness of Derailer relies on two key conditions: First, the proportion of cases with stable reasoning should be substantial - a condition satisfied by modern LLMs which tend to be either consistently right or consistently wrong on many tasks as demonstrated in our preliminary studies



and other work (Wan et al., 2024). Second, the stability check should be computationally efficient by keeping the number of samples  $n$  low. By meeting these conditions, Derailer optimizes the trade-off between computational cost and answer quality, applying more intensive reasoning procedures only when the model’s unstable reasoning patterns suggest potential for improvement through iteration.

### 3.2 Rerailer: Stabilizing Inconsistent Reasoning

**Motivation:** While Derailer serves as a sampling-based diagnostic mechanism that can precede any iterative refinement algorithm, existing approaches aren’t specifically designed to handle cases of unstable reasoning. Through analysis of examples like Figure 12, we observe that models often make intermediate mistakes due to incorrect path selection and unstable execution of otherwise viable strategies. This observation motivates our Rerailer mechanism, which stabilizes inconsistent reasoning patterns through targeted pairwise comparisons—particularly beneficial for questions where LLMs demonstrate potential but lack consistency (as evidenced by the left-tailed distribution in Figure 3).

The core insight behind Rerailer is that unstable performance typically stems from error accumulation in multi-step reasoning, formalized as

$$P(y, S|x) = P(s_1|x) \prod_{i=2}^n P(s_i|s_{1:i-1}, x) P(y|S, x) \quad (1)$$

where errors at any intermediate step  $s_i$  can propagate through conditional dependencies  $P(s_i|s_{1:i-1}, x)$ , destabilizing the entire process even when the model possesses the fundamental capability. While approaches like Tree of Thoughts (ToT) similarly explore multiple decoding paths, Rerailer offers two key advantages: (1) the evaluation for each state employs pairwise comparisons between independent solutions rather than direct value assignment through few-shot learning, proving more reliable as models generally perform better at comparative judgments than absolute scoring; and (2) we employ a new search mechanism that dynamically explores branches only when comparisons yield ties, rather than maintaining a full tree exploration—enabling focused intervention precisely where reasoning instability occurs while preserving both accuracy and computational efficiency.

**Framework and Implementation:** As demonstrated on the bottom half of Fig 1, The Rerailer implements an efficient approach to stabilize multi-step reasoning through four key stages:

1. **Candidate Generation:** For each reasoning step  $i$ , generate two candidate solutions using an adaptive sampling strategy:

$$\{s_i^1, s_i^2\} \sim \mathcal{M}(\cdot|x, s_{1:i-1})$$

The language model  $\mathcal{M}$  is conditioned on the problem instance  $x$  and the previously generated reasoning steps  $s_{1:i-1}$ .

2. **Pairwise Comparison:** Leverage the LLM’s inherent reasoning capabilities to evaluate the semantic and logical differences between candidate thoughts. We employ a bidirectional voting mechanism where each pair is compared twice, with a scoring function  $f(s_i^1, s_i^2)$  defined as:

$$f(s_i^1, s_i^2) = \begin{cases} (2, 0) & \text{if both favor } s_i^1 \text{ over } s_i^2 \\ (1, 1) & \text{if yield a tie decision} \\ (0, 2) & \text{if both favor } s_i^2 \text{ over } s_i^1 \end{cases}$$

Each solution pair is evaluated independently in both directions to mitigate potential ordering bias and ensure more robust outcomes.

3. **Adaptive Strategy Selection:** Implement an adaptive decision mechanism based on the comparison outcomes:

$$\sigma(s_i) = \begin{cases} \text{greedy} & f(s_i^1, s_i^2) = (2, 0), (0, 2) \\ \text{explore} & f(s_i^1, s_i^2) = (1, 1) \end{cases}$$

A greedy strategy is employed when both comparisons prefer one candidate, while an exploratory strategy is triggered when the comparisons yield a split decision, indicating reasoning instability that warrants exploration of both paths.

*Greedy Extension:* If  $\sigma(s_i) = \text{greedy}$ , select the preferred candidate based on the pairwise comparison scores:

$$s_i^* = \begin{cases} s_i^1 & \text{if } f(s_i^1, s_i^2) = 2 \\ s_i^2 & \text{if } f(s_i^1, s_i^2) = 0 \end{cases}$$

Append  $s_i^*$  to the current path for subsequent reasoning.

*Exploratory Extension:* If  $\sigma(s_i) = \text{exploratory}$ , the tied votes ( $f(s_i^1, s_i^2) = (1, 1)$ ) indicate reasoning instability. Extend the current path in

**both** directions, creating parallel paths to explore alternative reasoning routes. Both extensions are retained for subsequent reasoning.

The algorithm then returns to **(1) Candidate Generation** for the next reasoning step ( $i + 1$ ), using the extended path(s) as the new  $s_{1:i}$ . This iterative process continues until a complete solution is reached.

4. **Final Path (Answer) Selection:** After the iterative reasoning process concludes, evaluate all generated paths. Calculate a score for each path  $S = \{s_1, s_2, \dots, s_n\}$  by summing the scores of the individual steps:

$$\text{Score}(S) = \sum_{i=1}^n f(s_i, \cdot)$$

Select the path with the highest score as the final reasoning chain  $S^*$ :

$$S^* = \arg \max_S \text{Score}(S)$$

This selects the answer  $y$  which is expected to be the last step of  $S^*$ .

This structured process systematically improves reasoning stability for complex cases. By focusing on stabilizing intermediate steps, the Rerailer addresses the core challenge of *unstable reasoning* questions left unsolved by the Derailer.

## 4 Experiments

We evaluate our framework across three broad categories: Mathematical, Symbolic, and Commonsense Reasoning, using 27 data categories from a total of 7 standard benchmarks (e.g Big-BenchHard, MATH, StrategyQA, etc.) to ensure comprehensive coverage of real-world reasoning challenges. Experiments are conducted using three open and closed source LLMs with fixed temperature of 0.5, comparing against several established baselines that represent different paradigms in multi-step reasoning: Least to Most, Chain-of-Thought and its variants represent immediate prompting approaches that generate reasoning paths in a single forward pass; We also explore (1) Self-Consistency (SC) explores multiple solution paths independently; (2) Chain-of-Verification (CoVe) focuses on sequentially improving a single reasoning path through step-wise validation; and (3) Tree-of-Thought (ToT) combines both parallel

exploration and sequential refinement through a tree-structured search as three baselines in iterative prompting. These selected methods represent fundamental methods that imply applicability to many advanced techniques building upon these methods (Liu et al., 2024b; Yao et al., 2024b)

To evaluate the performance of different prompting methods, we evaluate (1) *effectiveness*: through accuracy across different reasoning types, and (2) *efficiency*: through total token consumption (sum of input and output tokens). To better show and evaluate the trade-off, we additionally introduce a novel metric, *Accuracy Gain per K Token (AGKT)*, which quantifies the efficiency of prompting methods by measuring accuracy improvement per 1K tokens relative to the zero-shot baseline. Unlike time and number of API calls, this metric provides a Model and hardware independent assessment of computational efficiency, offering more reliable comparisons than direct runtime measurements that can vary across different settings.

### 4.1 Main Results

We analyze the experimental results from Table 2 along several dimensions:

**Performance Gains over Various Prompting.** Our Derailer-Rerailer framework consistently outperforms all baseline methods across all evaluated models, demonstrating superior performance in both accuracy and computational efficiency. This synergistic combination leverages the strengths of both components: Derailer’s ability to reduce computational overhead and Rerailer’s effectiveness in error correction, resulting in a robust and efficient reasoning system.

**Excelling on Math and Symbolic Reasoning Tasks.** Analysis reveals distinct patterns of improvement across different reasoning types. The framework shows particularly strong performance in mathematical and symbolic reasoning tasks, while improvements in commonsense reasoning are more moderate. This disparity aligns with empirical expectations, as structured reasoning tasks typically benefit more from explicit verification processes. The smaller gains in commonsense reasoning can be attributed to the inherent ambiguity in these tasks, where determining the validity of intermediate steps requires more nuanced evaluation. **Efficiency Gain with Iterative Prompting.** While Iterative prompting methods like Self-Consistency demonstrate accuracy improvements over immediate prompting, they come at a substantial com-

Table 2: Comparison of reasoning methods across multiple models and datasets. All accuracy metrics are expressed in percentages. **Math Acc** measures performance on mathematical reasoning problems. **Symbolic Acc** measures accuracy on symbolic reasoning tasks. **Commonsense Acc** reflects performance on common-sense reasoning tasks. **Overall Acc** is a combined measure across these categories. **Tokens (K)** indicates the total number of tokens used (in thousands). **Acc/ K Token Gain (AGKT)** shows how much the accuracy improves per 1K tokens over that model’s zero-shot CoT baseline. Values in parentheses next to the Overall Acc indicate the relative improvement compared to the zero-shot CoT baseline for each model.

Model	Method	Math Acc (%)	Symbolic Acc (%)	Commonsense Acc (%)	Overall Acc (%)	Tokens (K)	Acc Gain per K Tokens (%)
Claude-3.5-Sonnet	Zero-shot CoT	68.3	77.6	72.2	72.7	0.108	-
	Least-to-Most CoT	70.4	79.3	74.1	74.6 (+1.9)	0.372	5.11
	Five-shot CoT	72.9	81.4	74.8	76.4 (+3.7)	0.518	7.14
	Self-Consistency(SC)	77.8	86.5	75.2	79.8 (+7.1)	1.732	4.10
	Derailer + SC	77.6	86.3	75.1	79.7 (+7.0)	0.762	9.19
	Chain of Ver (CoVe)	78.2	86.8	75.4	80.1 (+7.4)	1.778	4.16
	Derailer + CoVe	77.9	86.6	75.3	79.9 (+7.2)	0.793	9.08
	Tree of Thought (ToT)	78.1	86.7	75.4	80.1 (+7.4)	2.245	3.30
	Derailer + ToT	77.8	86.5	75.2	79.8 (+7.1)	0.845	8.40
	Rerailer	78.0	86.7	75.3	80.0 (+7.3)	1.458	5.01
	<b>Derailer + Rerailer</b>	<b>80.8</b>	<b>89.7</b>	<b>75.9</b>	<b>82.1 (+9.4)</b>	0.592	<b>15.88</b>
Llama-3.1 70B	Zero-shot CoT	38.2	71.3	70.4	60.0	0.132	-
	Least-to-Most CoT	39.6	72.4	72.3	61.4 (+1.4)	0.448	3.13
	Five-shot CoT	41.8	74.7	72.9	63.1 (+3.1)	0.628	4.94
	Self-Consistency(SC)	45.9	79.1	73.4	66.1 (+6.1)	1.932	3.16
	Derailer + SC	45.7	78.8	73.3	65.9 (+5.9)	0.892	6.61
	Chain of Ver (CoVe)	46.2	79.4	73.6	66.4 (+6.4)	1.972	3.25
	Derailer + CoVe	46.0	79.2	73.5	66.2 (+6.2)	0.923	6.72
	Tree of Thought (ToT)	46.1	79.3	73.5	66.3 (+6.3)	2.458	2.56
	Derailer + ToT	45.9	79.1	73.4	66.1 (+6.1)	0.968	6.30
	Rerailer	46.1	79.3	73.5	66.3 (+6.3)	1.652	3.81
	<b>Derailer + Rerailer</b>	<b>48.4</b>	<b>81.8</b>	<b>74.2</b>	<b>68.1 (+8.1)</b>	0.648	<b>12.50</b>
GPT-4o-mini	Zero-shot CoT	59.5	46.8	69.2	58.5	0.121	-
	Least-to-Most CoT	62.4	49.1	71.8	61.1 (+2.6)	0.389	6.68
	Five-shot CoT	63.9	50.8	72.4	62.4 (+3.9)	0.556	7.01
	Self-Consistency (SC)	68.6	55.2	72.9	65.6 (+7.1)	1.795	3.96
	Derailer + SC	68.3	54.9	72.8	65.3 (+6.8)	0.823	8.26
	Chain of Ver (CoVe)	69.3	55.5	73.2	66.0 (+7.5)	1.842	4.07
	Derailer + CoVe	68.9	55.3	73.1	65.8 (+7.3)	0.864	8.45
	Tree of Thought (ToT)	69.2	55.4	73.1	65.9 (+7.4)	2.325	3.18
	Derailer + ToT	68.8	55.2	73.0	65.7 (+7.2)	0.912	7.89
	Rerailer	69.0	55.4	73.1	65.8 (+7.3)	1.539	4.74
	<b>Derailer + Rerailer</b>	<b>72.0</b>	<b>58.5</b>	<b>73.8</b>	<b>68.1 (+9.6)</b>	0.639	<b>15.02</b>

putational cost. Our Derailer component, when integrated with these complex iterative prompting methods, maintains comparable accuracy while significantly reducing token consumption. This demonstrates that strategic, selective verification can be more resource-efficient than comprehensive checking approaches without compromising performance.

## 4.2 Abalation Study and Analysis

**Sample Size - Derailer Analysis:** We conduct a detailed analysis of how the number of samples  $n$  in the Derailer affects the overall framework performance. Fig. 4 shows the relationship between three key metrics: accuracy, token consumption, and accuracy gain per thousand tokens (AGTK). While accuracy continues to improve with more

samples, increasing from 64.3% at  $n = 2$  to 69% at  $n = 10$ , the token consumption grows linearly with  $n$ . The efficiency metric AGTK reveals a clear optimal point at  $n = 5$ , where the framework achieves the best balance between accuracy improvement and computational cost. Beyond this point, the marginal accuracy gains (less than 1% per additional sample) do not justify the increased token consumption, as evidenced by the sharp decline in AGTK after  $n = 5$ .

We did additional analysis in Appendix A.4.2 to reveal that the pass rates (percentage of consistency) of Derailer remain stable at around 50% across different  $n$ , suggesting that approximately half of the questions are identified as either *Consistently Solvable* or *Intrinsically Insolvable*. This stability aligns with our preliminary findings about

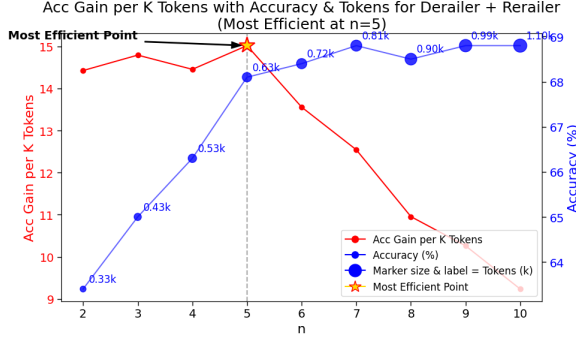


Figure 4: Analysis of sample size ( $n$ ) effects in the Derailer-Rerailer framework. The blue line (right y-axis) shows accuracy improvement with increasing samples, with marker size indicating token consumption. The red line (left y-axis) shows efficiency measured by accuracy gain per thousand tokens. The star marks the optimal efficiency point at  $n = 5$ . Results are averaged across all models on math tasks

Table 3: Comparison of Different Preference Ranking Methods and Number of Samples Generated at Each Step  $k$ . Accuracy in % and Tokens in K. Results are averaged across all models

Task	Binary ( $k = 2$ )		Ranking ( $k = 3$ )		Pairwise ( $k = 3$ )	
	Acc (%)	Token (K)	Acc (%)	Token (K)	Acc (%)	Token (K)
Math	67.2	1.59	65.8	2.12	67.3	2.88
Symbolic	75.2	1.52	73.5	2.08	75.1	2.72
Commonsense	71.2	1.40	70.8	1.85	71.2	2.25
Avg	70.8	1.47	70.2	2.02	70.9	2.61

the relationship between answer consistency and question solvability, providing additional evidence that a small number of samples is sufficient for reliable classification.

**Sample Size and Comparison Mechanism - Rerailer Analysis:** We investigated two key aspects of the Rerailer framework: expanding the number of samples generated at each step  $k$  and varying the comparison mechanism. First, we explored increasing the number of samples from  $k = 2$  to  $k = 3$ , while maintaining the pairwise comparison approach. For a sample size  $k$ , the pairwise comparison requires  $\binom{k}{2}$  comparisons, leading to  $O(k^2)$  complexity in terms of model queries. This quadratic growth in computational cost is reflected in our empirical results: increasing from  $k = 2$  to  $k = 3$  maintains similar accuracy levels but incurs a 77% increase in token consumption. We also examined an alternative ranking-based approach where the model directly orders  $k = 3$  samples simultaneously. While this approach achieves  $O(k)$  complexity, it leads to decreased performance across all reasoning tasks.

This superiority of pairwise comparisons, despite their higher computational cost, aligns with recent findings that language models excel at *comparative judgments rather than holistic ranking tasks*, which is often biased by positional (Wang et al., 2024b; Liu et al., 2024a). The results further demonstrate the design choice of the Rerailer by leveraging the model’s comparative reasoning strengths while keeping computational costs manageable.

**Case Study:** As suggested in Table 2, Our Framework particularly excels in correcting flawed chains of thought in mathematical reasoning, addressing the key challenges of unstabilized reasoning patterns. Fig.5 demonstrates the Rerailer’s effectiveness in identifying and rectifying errors in both basic and advanced math problems. For instance, in a counting problem, the Rerailer successfully stabilizes divergent reasoning paths by correcting computational mistakes that would have led to an incorrect final answer. Similarly, in a differential equation example, it identifies fundamental methodological errors and realigns the solution approach with correct mathematical principles. This correction mechanism directly addresses the common types of unstabilized reasoning: it prevents solution drift, resolves conflicting intermediate results, and remedies incomplete reasoning chains. The core capability of the Rerailer lies in stabilizing these patterns through systematic verification of intermediate steps, particularly benefiting tasks that can be *decomposed into distinct, verifiable components*.

## 5 Related Work

### 5.1 Prompting LLMs for Reasoning

In-context Learning (Dong et al., 2024), including few-shot Chain-of-Thought prompting (Wei et al., 2022b), marked a breakthrough in eliciting reasoning capabilities from language models by encouraging them to articulate intermediate thought steps. Building upon this insight, researchers have developed various sophisticated methods to enhance reasoning performance. These approaches include different thinking style decompositions (Zhou et al., 2023), exploring multiple parallel reasoning paths (Wang et al., 2023; Yao et al., 2024a), and sequential verification or refinement of generated rationales (Madaan et al., 2023; Dhuliawala et al., 2023), or combinations thereof (Yao et al., 2024a; Snell et al., 2024a). Other techniques, particularly those related to LLM agents (Wang et al., 2024a), rely



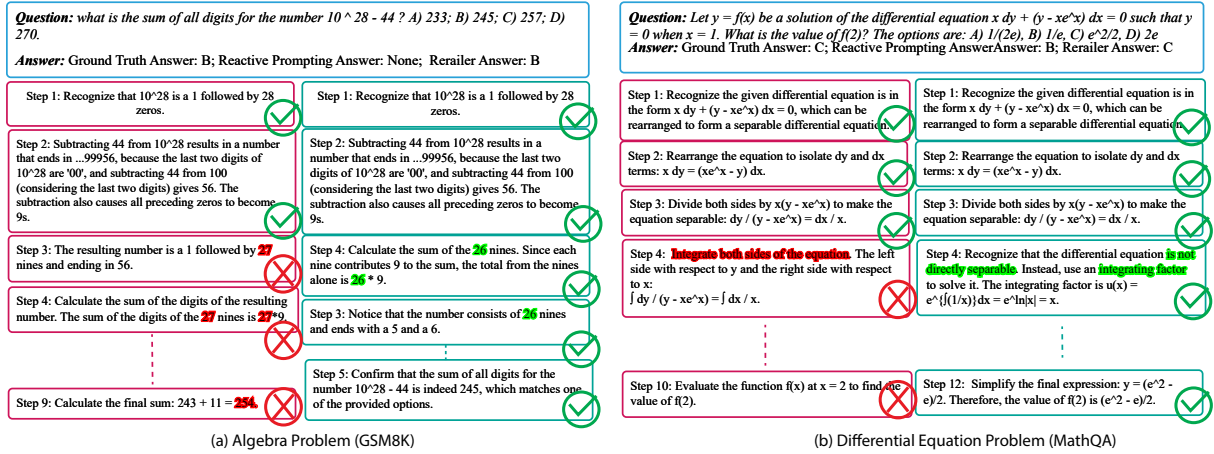


Figure 5: Derailer-Rerailer Solving Math Questions. The questions and answers, which were retrieved from the GSM8K and MathQA datasets, are exhibited in the blue box. The red boxes are steps generated via the baseline CoT method and the green boxes are the corrected RP from Rerailer. Mistakes are highlighted in red and corrections are highlighted in green. Additional examples across different problem types can be found in Appendix A.5.

on external knowledge or tool integration (Shinn et al., 2023; Gao et al., 2023; Yao et al., 2023), while additional training methods can also enhance reasoning capabilities (Trung et al., 2024). However, our paper focuses specifically on inference-time elicitation techniques for reasoning abilities already present in base models, which serve as the foundations for these more complex methods.

## 5.2 Efficiency in Large Language Model Inference

Despite significant advancements in reasoning capabilities through various prompting techniques, a critical challenge limiting the practical deployment of LLMs is their computational inefficiency (Wang et al., 2025a). These models frequently exhibit unproductive behaviors, including generating redundant verifications after reaching correct answers or pursuing "fake thinking" patterns that simulate progress without meaningful problem-solving (Wu et al., 2025b; Wang et al., 2025b), misallocating computational resources by applying uniform algorithms regardless of problem complexity or by inadequately distributing computation between simple and complex tasks (Wan et al., 2024; Parashar et al., 2025; Yang et al., 2025). While existing solutions typically rely on sophisticated predictive strategies for budget allocation and algorithm selection (Han et al., 2025; Snell et al., 2024b), we propose a simpler, more principled approach based on model’s consistency for the input queries with (1) a Derailer mechanism that enhances efficiency by terminating unproductive reasoning paths, and (2) a Rerailer component that recovers accuracy in cases where

the Derailer incorrectly intervenes.

## 6 Conclusion

In this work, we proposed **Derailer-Rerailer**, a novel two-stage framework that balances LLM reasoning accuracy and computational efficiency. Our framework demonstrates that selective verification can maintain the benefits of exhaustive reasoning while significantly reducing computational costs. Beyond the technical contribution, our findings reveal important practical implications: while increased computation generally improves accuracy, this relationship isn’t linear, and many methods reach diminishing returns despite substantial resource investment. As LLMs continue to grow in both size and capability, we advocate for a more holistic evaluation approach that considers both accuracy and efficiency metrics for future research. This shift in evaluation criteria is particularly crucial for advancing research on LLM deployment in resource-constrained environments.

## Limitations

Our framework demonstrates promising results, but several important limitations warrant discussion:

**Preliminary Stability Analysis:** Our investigation of the relationship between the model’s capability and answer consistency mainly motivates the design of our framework and remains high-level. A deeper theoretical understanding of how this relationship varies across reasoning tasks, model architectures, and knowledge domains could yield more nuanced strategies for applying iterative prompting adaptively.

**Sampling Parameter Selection:** Determining the optimal number of samples for the Derailer stage remains challenging, as this parameter significantly impacts both efficiency and effectiveness. Currently, this depends on model’s pre-training and requires empirical tuning, which may not be practical in all deployment scenarios.

**Single Model Constraints:** The scope of this paper is to elicit the reasoning of a single model without external sources, extending the discussions of efficiency on understanding the recently developed reasoning models (DeepSeek-AI et al., 2025) and exploring applications related to LLM’s agents (Li et al., 2023) would be essential future directions.

## Acknowledgments

The work is supported in part by the National Science Foundation under Grants IIS-2316306 and CNS-2330215. The authors acknowledge Research Computing at The University of Virginia for providing computational resources and technical support that have contributed to the results reported within this publication.

## References

- Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. [Mathqa: Towards interpretable math word problem solving with operation-based formalisms](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.
- Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.
- 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. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanbiao Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yudian Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. [Chain-of-verification reduces hallucination in large language models](#). *Preprint*, arXiv:2309.11495.
- Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. [A survey on in-context learning](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. [Pal: Program-aided language models](#). *Preprint*, arXiv:2211.10435.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. [Did aristotle](#)

- use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.
- G Gronchi and F Giovannelli. 2018. Dual process theory of thought and default mode network: A possible neural foundation of fast thinking. *Frontiers in Psychology*, 9:1237.
- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A survey on llm-as-a-judge. *Preprint*, arXiv:2411.15594.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.
- Dong Huang, Jie M. Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. 2024. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. *Preprint*, arXiv:2312.13010.
- Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York.
- Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative agents for “mind” exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations*.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Yanming Liu, Xinyue Peng, Tianyu Du, Jianwei Yin, Weihao Liu, and Xuhong Zhang. 2024b. ERA-CoT: Improving chain-of-thought through entity relationship analysis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8780–8794, Bangkok, Thailand. Association for Computational Linguistics.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Ning Miao, Yee Whye Teh, and Tom Rainforth. 2024. Selfcheck: Using LLMs to zero-shot check their own step-by-step reasoning. In *The Twelfth International Conference on Learning Representations*.
- OpenAI. 2022. Measuring goodhart’s law. <https://openai.com/index/measuring-goodharts-law/>.
- OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.
- Shubham Parashar, Blake Olson, Sambhav Khurana, Eric Li, Hongyi Ling, James Caverlee, and Shuiwang Ji. 2025. Inference-time computations for llm reasoning and planning: A benchmark and insights. *arXiv preprint arXiv:2502.12521*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024a. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024b. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- Aarohi Srivastava et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615. *Preprint*, arXiv:2206.04615.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller,



- Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *Preprint*, arXiv:2307.09288.
- Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. [ReFT: Reasoning with reinforced fine-tuning](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614, Bangkok, Thailand. Association for Computational Linguistics.
- D. Umerenkov, G. Zubkova, and A. Nesterov. 2023. [Deciphering diagnoses: How large language models explanations influence clinical decision making](#). *Preprint*, arXiv:2310.01708.
- Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2024. Dynamic self-consistency: Leveraging reasoning paths for efficient llm sampling. *arXiv preprint arXiv:2408.17017*.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024a. [A survey on large language model based autonomous agents](#). *Frontiers of Computer Science*, 18(6).
- Peifeng Wang, Austin Xu, Yilun Zhou, Caiming Xiong, and Shafiq Joty. 2024b. [Direct judgement preference optimization](#). *Preprint*, arXiv:2409.14664.
- Rui Wang, Hongru Wang, Boyang Xue, Jianhui Pang, Shudong Liu, Yi Chen, Jiahao Qiu, Derek Fai Wong, Heng Ji, and Kam-Fai Wong. 2025a. [Harnessing the reasoning economy: A survey of efficient reasoning for large language models](#). *Preprint*, arXiv:2503.24377.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). *Preprint*, arXiv:2203.11171.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, et al. 2024c. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.
- Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. 2025b. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. [Emergent abilities of large language models](#). *Transactions on Machine Learning Research*. Survey Certification.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Yuqi Wu, Guangya Wan, Jingjing Li, Shengming Zhao, Lingfeng Ma, Tianyi Ye, Ion Pop, Yanbo Zhang, and Jie Chen. 2025a. [Wisemind: Recontextualizing ai with a knowledge-guided, theory-informed multi-agent framework for instrumental and humanistic benefits](#). *Preprint*, arXiv:2502.20689.
- Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025b. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.
- Wenkai Yang, Kangwook Lee, Robert D Nowak, and Dimitris Papailiopoulos. 2025. Towards thinking-optimal scaling of test-time compute for llm reasoning. *arXiv preprint arXiv:2502.18080*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. [ReAct: Synergizing reasoning and acting in language models](#). In *International Conference on Learning Representations (ICLR)*.
- Yao Yao, Zuchao Li, and Hai Zhao. 2024b. [GoT: Effective graph-of-thought reasoning in language models](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2901–2921, Mexico City, Mexico. Association for Computational Linguistics.
- Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. [Siren’s song in the ai ocean: A survey on hallucination in large language models](#). *Preprint*, arXiv:2309.01219.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans,



Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). In *The Eleventh International Conference on Learning Representations*.

## A Appendix

### A.1 Package Used and Code

For generating LLMs responses and parsing out answers, we utilize packages "langchain", "langchain\_openai", "langchain\_anthropic", "langchain\_community", and "langchain\_core" offered by Langchain<sup>2</sup>. In addition, we use "pandas" for data processing, "matplotlib" and "seaborn" for visualization, and "numpy" for basic mathematical manipulation.

Here is the code link for our GitHub: [https://github.com/wan19990901/CoT\\_rerailer](https://github.com/wan19990901/CoT_rerailer)

### A.2 Experiments Details

#### A.2.1 Data and Models

We source our evaluation data from over 20 categories across standard benchmarks:

- **Mathematical Reasoning:** MathQA (Amini et al., 2019), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021)
- **Symbolic Reasoning:** CoinFlip (Wei et al., 2022b), selected tasks from BigBench (Srivastava et al., 2023)
- **Commonsense Reasoning:** CommonsenseQA (Talmor et al., 2019), StrategyQA (Geva et al., 2021), MMLU (Hendrycks et al., 2021), MMLU-Professional (Wang et al., 2024c)

**Models:** We utilize three state-of-the-art LLMs:

- Claude-3.5-Sonnet (Anthropic, 2024)
- Llama-3.1 70B (Touvron et al., 2023)
- GPT-4o-mini (OpenAI, 2024)

For all experiments, we use a temperature of 0.5 to balance exploration and consistency. **Baselines:** We compare against several standard prompting methods:

- Zero-shot and few-shot Chain-of-Thought (CoT)

<sup>2</sup><https://www.langchain.com/>

- Least-to-Most
- Self-Consistency (SC) with  $n = 20$  samples
- Chain-of-Verification (CoVe)

**Data Sources** Our experiments draw from multiple standardized benchmarks to ensure comprehensive evaluation across different reasoning types:

- **BigBench:** We use multiple tasks including Date Understanding, Logical Deduction, and Penguins in a Table, which test different aspects of reasoning capabilities; we also take subset of data from **BIG-Bench Hard (BBH)**: to focus on Boolean Expressions, Object Counting, Date Understanding, Sports Understanding, and Temporal Sequences tasks for evaluation as alternative datasets for symbolic reasoning
- **Various Mathematical Reasoning Benchmark:** We incorporate GSM8K, MathQA, and MATH datasets for comprehensive mathematical reasoning assessment.
- **Various Benchmark Commonsense Reasoning:** We use CommonsenseQA and StrategyQA for evaluating practical reasoning and strategic thinking capabilities.
- **MMLU:** We utilize both general and professional subjects from MMLU, including medicine, law, engineering, and various scientific disciplines.
- **Additional Benchmarks:** We also dataset including CoinFlip for symbolic reasoning evaluation.

**Task Categories** Our evaluation framework spans three broad categories of reasoning:

- **Mathematical Reasoning** (1500 questions):
  - Core Mathematics (MATH, MathQA)
  - Applied Mathematics (GSM8K)
  - Professional Mathematics (MMLU-Mathematics)
  - Statistical Reasoning
  - Abstract Algebra
  - Quantitative Problem-Solving
- **Symbolic Reasoning** (1000 questions):
  - Formal Logic

- Boolean Expressions (BBH)
- Temporal Reasoning
- Programming Concepts
- Physical Systems
- Circuit Analysis

• **Commonsense Reasoning** (1500 questions):

- General Common Sense (CommonsenseQA)
- Strategic Thinking (StrategyQA)
- Temporal Understanding
- Ethical Reasoning
- Professional Judgment
- Scientific Reasoning

**Evaluation Protocol** We conduct experiments using a standardized protocol across all models and tasks:

- Each experiment is repeated 10 times to ensure statistical reliability
- Standard deviation is calculated across all metrics
- Token usage is carefully tracked for efficiency analysis
- Performance is measured both within categories and across the entire test set

The categorization is designed to reflect the diversity and scope of the subjects our models are evaluated against, ensuring a comprehensive assessment across a wide array of knowledge domains. To obtain LLM RP for the experiment, we spend roughly \$2000 USD and 250 hrs in total for LLM API usage, combining both Claude and OPENAI.

### A.3 Prompts

In this section, we included our prompts for each component and their corresponding results.

#### A.3.1 Zero-Shot CoT Generator

**System Message:** *You are an expert in {subject}. Break down this problem step-by-step, showing all calculations and intermediate reasoning. Let's solve this systematically.*

**Human:** {question}

#### A.3.2 Least-to-Most CoT Generator

**System Message:** *You are an expert problem solver. Break down complex problems into simpler sub-problems, solving them in order of increasing difficulty.*

**Human:** *In a school fundraiser, tickets cost \$5 each. If they sold 145 tickets and the school keeps 70% of the money, how much does the school keep?*

**Assistant:** Let me break this down: 1. Sub-questions (from simplest to most complex):  
- How much money did they collect in total?  
- What does 70% mean in decimal form? - How much money does the school keep?

2. Let's solve each: Q1: What's the total money collected from ticket sales? A1: 145 tickets  $\times$  \$5 = \$725 total

Q2: What's 70% as a decimal? A2: 70% = 70/100 = 0.70

Q3: How much does the school keep? A3: \$725  $\times$  0.70 = \$507.50

Therefore, the school keeps \$507.50.

#### A.3.3 Three-Shot CoT Generator

**System Message:** *You are an expert problem solver. For each problem, break down your reasoning step-by-step, showing all calculations and intermediate thoughts.*

**Human:** *What is 15% of 80?*

**Assistant:** Let me solve this step-by-step:  
1. Convert 15% to decimal: 15% = 0.15  
2. Multiply: 80  $\times$  0.15 = 12  
Therefore, 15% of 80 is 12.

**Human:** *If John has 3 apples and Mary has twice as many, how many apples do they have together?*

**Assistant:** Let me solve this step-by-step:  
1. John's apples = 3  
2. Mary's apples = 2  $\times$  3 = 6  
3. Total apples = John's + Mary's = 3 + 6 = 9  
Therefore, they have 9 apples together.

**Human:** *If 6 cookies cost \$2, how much do 15 cookies cost?*

**Assistant:** Let me solve this step-by-step:  
1. Find cost per cookie: \$2  $\div$  6 = \$0.33...  
2. Multiply by 15: \$0.33...  $\times$  15 = \$5  
Therefore, 15 cookies cost \$5.

**Human:** {question}

### A.3.4 Chain-of-Verification

#### CoVe Framework

##### 1. Baseline Response

- **Query:** {question}
- **System:** Generate initial response
- **Output:** [Initial model response]

##### 2. Plan Verifications

- **Context:** [Initial response]
- **System:** Generate verification questions for each claim
- **Output:** [List of verification questions]

##### 3. Execute Verifications

- **System:** Answer each question independently
- **Format:**
  - Q1: [Verification question]
  - A1: [Independent answer]
  - Qx: ...
  - Ax: ...

##### 4. Cross-Check Analysis

- **Original:** [Claim from initial response]
- **Verification:** [Q&A result]
- **Status:** [CONSISTENT/INCONSISTENT/PARTIAL]
- **Evidence:** [Explanation]

##### 5. Final Verified Response

- **Query:** [Original question]
- **System:** Generate revised response using verified claims
- **Output:** [Final verified response]

### A.3.5 Rerailer Prompt Implementation

**System Message:** *You are an expert problem solver implementing a multi-step reasoning verification system. For each step, generate and evaluate multiple solution paths to ensure stability.*

#### Stage 1: Multiple Solution Generation

For each reasoning step  $i$ :

- **Input:** Question  $x$  and previous steps  $s_{1:i-1}$
- **Prompt:** *Generate  $k$  diverse solutions for this step. Ensure solutions are meaningfully different.*
- **Output:**  $k$  candidate solutions  $\{s_i^1, \dots, s_i^k\}$

**Stage 2: Pairwise Evaluation** For each pair of solutions  $(s_i^p, s_i^q)$ :

- **Prompt:** *Compare these two solutions:*
  - Solution 1: [First solution]
  - Solution 2: [Second solution]
  - *Which solution is more logical and why?*
- **Output:** Binary comparison score  $f(s_i^p, s_i^q)$

**Stage 3: Score Aggregation** For each solution  $s_i^j$ :

- **Input:** All pairwise comparison results
- **Computation:**  $\sum_{l=1, l \neq j}^k f(s_i^j, s_i^l)$
- **Output:** Aggregate score for each solution

#### Stage 4: Optimal Path Selection

- **Input:** Aggregated scores for all solutions
- **Selection:** Select  $s_i^*$  with highest score
- **Chain Construction:** Build  $S^* = \{s_1^*, \dots, s_n^*\}$

**Example Implementation:**

**Question:** *If a rectangle has width 4cm and length twice its width, what is its area?*

**Stage 1 - Generate Solutions:**

- $s_1^1$ : "Width = 4cm, length =  $2 \times$  width"
- $s_1^2$ : "Width = 4cm, length = 8cm (double the width)"
- $s_1^3$ : "Given  $w = 4\text{cm}$ ,  $l = 2w = 8\text{cm}$ "

**Stage 2 - Compare:**

- $s_1^2$  vs  $s_1^1$ :  $s_1^2$  better (more explicit)
- $s_1^3$  vs  $s_1^1$ :  $s_1^3$  better (more systematic)
- $s_1^3$  vs  $s_1^2$ :  $s_1^3$  better (shows work)

**Stage 3 - Scores:**

- $s_1^1$ : 0 wins
- $s_1^2$ : 1 win
- $s_1^3$ : 2 wins

**Stage 4 - Select:**  $s_1^3$  chosen

**Final Answer:** Area = length  $\times$  width =  $8\text{cm} \times 4\text{cm} = 32\text{cm}^2$

**A.4 Additional Preliminary Findings**

Besides the sampling size findings, the preliminary study also reveals interesting implications for model size and task type as shown in Fig. 6

**A.4.1 Model Size and Consistency**

Our results indicate that the ratio of consistently correct or consistently incorrect cases to inconsistent cases differs across models. Larger models (e.g., gpt-4o) tend to have higher confidence and produce more consistent answers, whether correct or incorrect. Smaller models, such as a llama3-7b variant, exhibit greater variability in answers. Interestingly, while large models may have a high proportion of *Solvable* outcomes, smaller models show a broader spread across consistent and inconsistent categories.

These findings support the hypothesis that consistency level is correlated with model size and training sophistication. Larger models are generally more knowledgeable and stable, reducing the necessity for complex prompting in easily solvable questions but also being less susceptible to

improvement on questions they *consistently fail* without external knowledge augmentation.

**A.4.2 Task Type: Commonsense vs. Mathematical Reasoning**

When examining commonsense (CM) reasoning and mathematical reasoning tasks, we observe distinct patterns. For CM tasks, the LLM responses are often either consistently correct or consistently incorrect, suggesting that the model either “knows” the fact or does not. The variability in these responses is low, indicating minimal benefit from advanced prompting techniques: if the model’s knowledge is lacking, complex prompts alone are unlikely to help. In contrast, mathematical reasoning tasks show a more substantial spread, with a notable portion falling into the *inconsistently solvable* category. This suggests that the model’s reasoning processes for mathematics are more delicate and prone to occasional errors or hallucinations, even when partial capability exists. Here, advanced prompting techniques, such as step-by-step reasoning or self-checking mechanisms, can meaningfully increase consistency and correctness. This reinforces the rationale for using specialized derailing strategies—introducing additional reasoning steps, verification routines, or multi-stage solution prompts.

**Pass Rate Analysis**

We provide a detailed analysis of Derailer’s pass rates (proportion of questions flagged for iterative prompting) across different sample sizes and reasoning categories. As shown in Fig. 7, the overall pass rates remain remarkably stable across different values of  $N$ . When breaking down by reasoning types, we observe some variation in absolute pass rates between commonsense (Fig. 8), mathematical (Fig. 9), and symbolic reasoning (Fig. 10), reflecting the different difficulty levels of these domains. However, the stability pattern holds across all categories - increasing the number of samples beyond 5 does not significantly change which questions are flagged for additional reasoning. This consistency supports our main paper’s findings about the efficiency of small sample sizes for detecting reasoning stability.

**A.5 Error Analysis and Case Study****A.5.1 Error Correction Analysis**

We provided a detailed analysis on how the instablized reasoning gets fixed after Derailer. The Figure



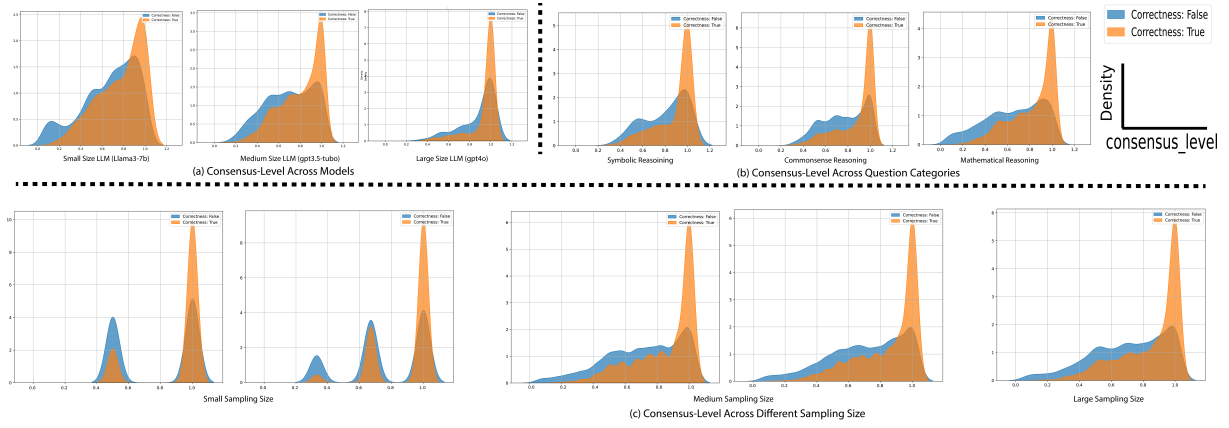


Figure 6: Consensus level and question type across different models and categories.

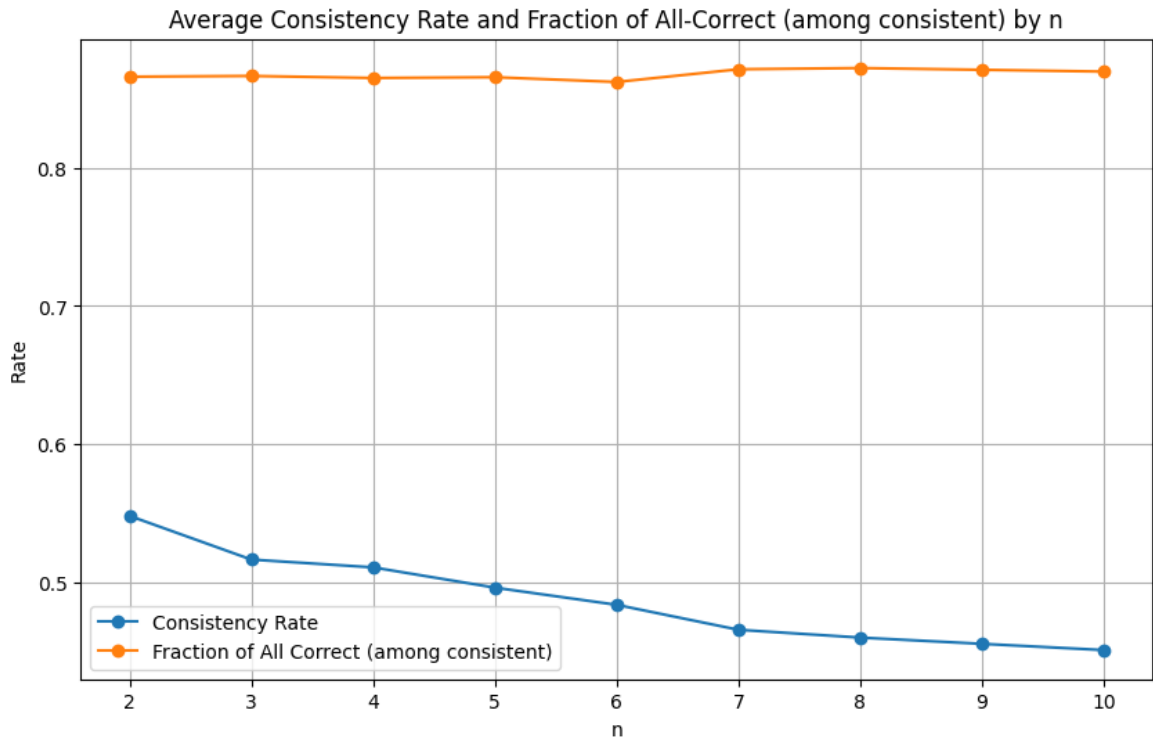


Figure 7: pass rate of all of data of varying N

11 demonstrates Rerailer’s balanced approach to handling unstable reasoning patterns. While maintaining comparable stability for correct answers (Correct→Correct 34%), Rerailer achieves the highest rate of error correction (Incorrect→Correct at 9.5%) while showing the lowest rate of destabilizing correct answers (Correct→Incorrect at 1.8%). This indicates our proposed Rerailer’s ability to effectively identify and fix genuine errors without over-correcting stable reasoning paths, outperforming both CoVe and Self-Consistency while preserving good efficiency.

errors occurred with potentially downgrading our performance including wrong ground-truth, lack of background information, and ambiguous questions. The detailed example were show in Fig. 13

In our experiment, mainly three types of typical

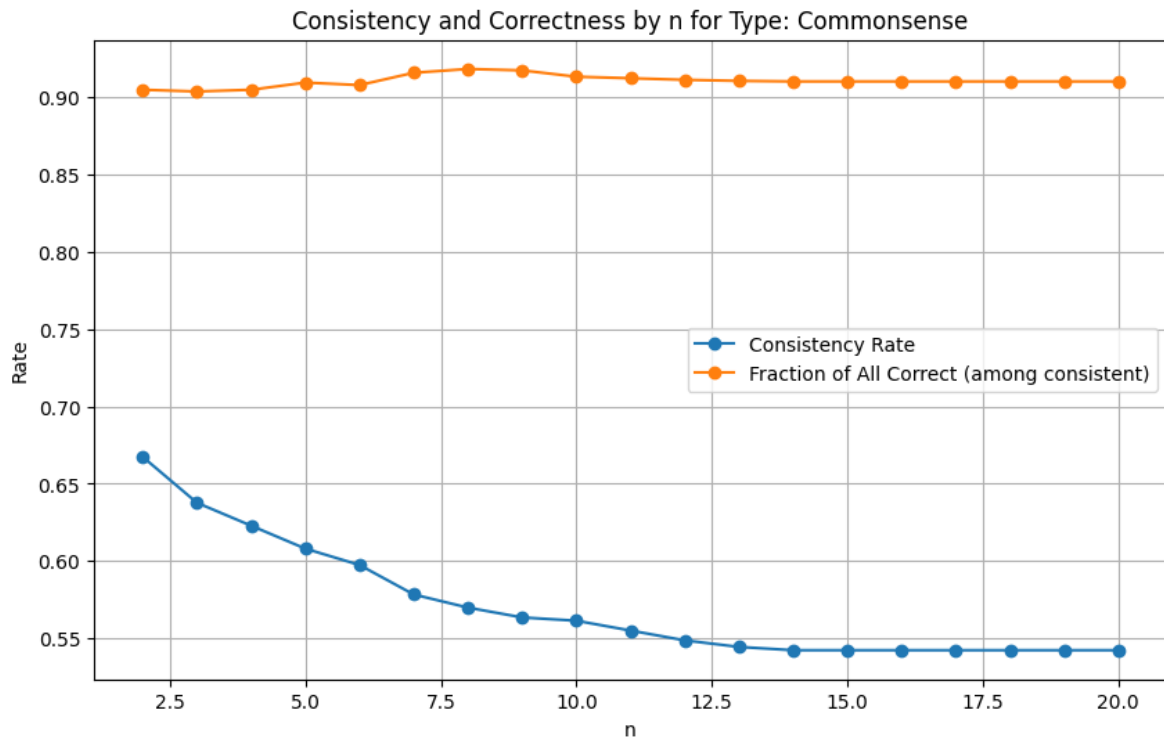


Figure 8: pass rate of Commonsense reasoning of varying N

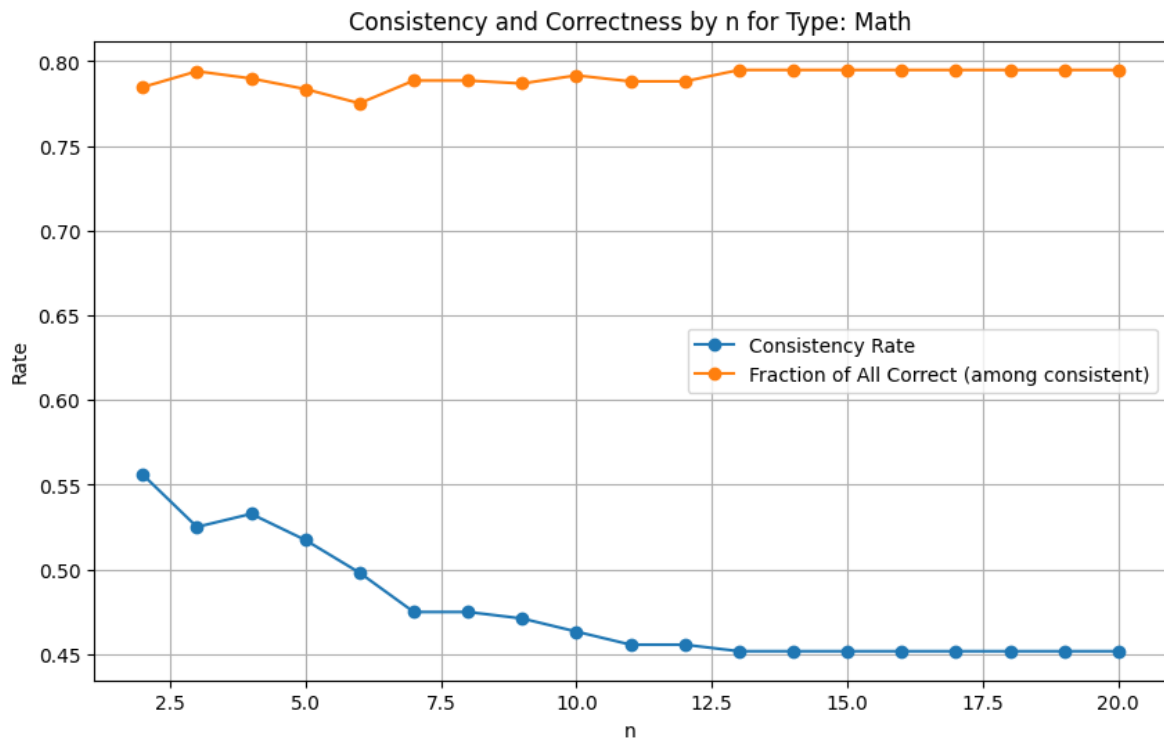


Figure 9: pass rate of all Math data of varying N

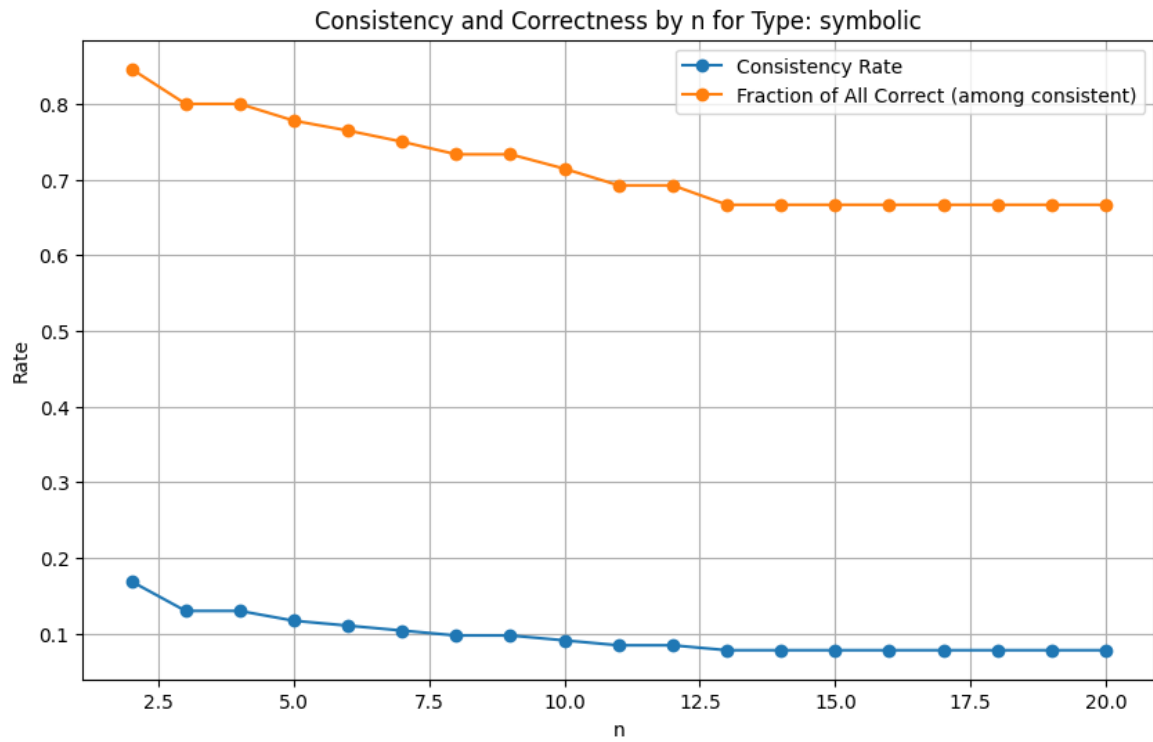


Figure 10: pass rate of Symbolic Reasoning data of varying N

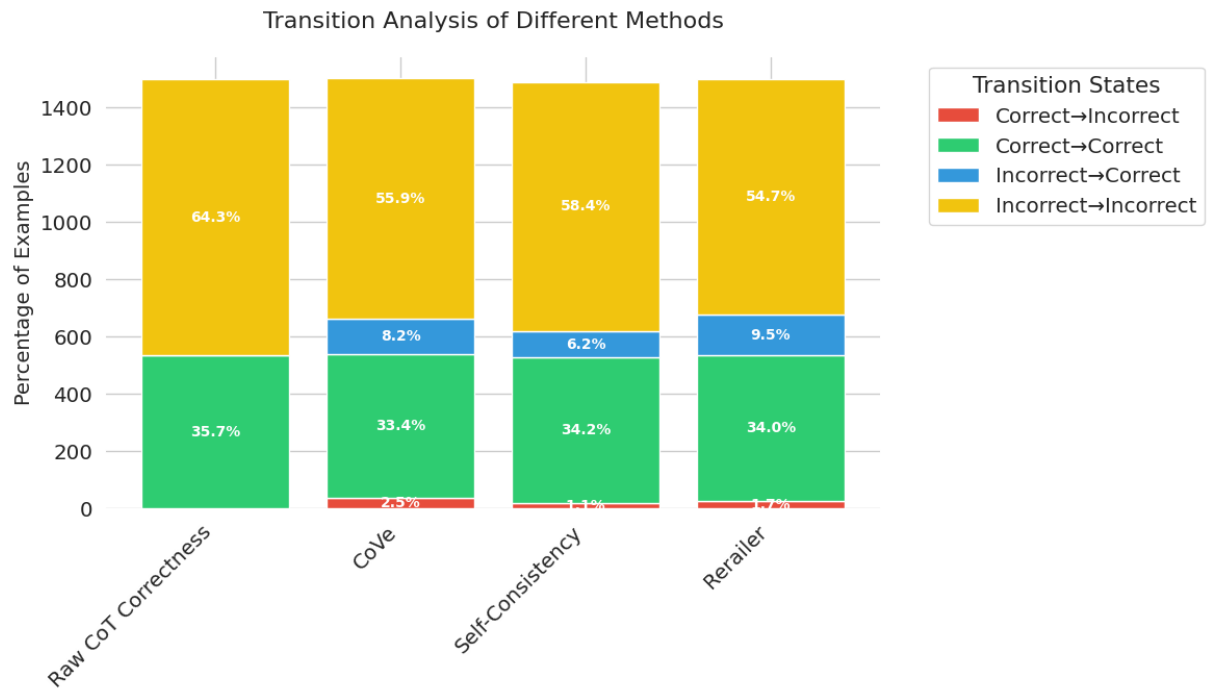


Figure 11: Correction ratio of various iterative prompting method with respect to CoT data after Derailer

<p><b>Question:</b> White light is normally incident on a puddle of water (index of refraction 1.33). A thin (500 nm) layer of oil (index of refraction 1.5) floats on the surface of the puddle. Of the following, the most strongly reflected wavelength is?</p> <p>A) 500nm. B) 550nm. C) 600nm. D) 650nm.</p> <p><b>Answer:</b></p> <p>- Ground Truth Answer: C - Plain CoT Answer: D - Rerailer Answer: C</p>	<p>Step 1: Understand that the phenomenon described is thin-film interference, which occurs when light reflects off the two surfaces of a thin film, causing interference. The most strongly reflected wavelength will be the one that undergoes constructive interference.</p>	<p>Step 1: Understand that the phenomenon described is thin-film interference, which occurs when light reflects off the two surfaces of a thin film, causing interference. The most strongly reflected wavelength will be the one that undergoes constructive interference.</p>
	<p>Step 2: Recall that constructive interference occurs when the path difference between the two reflected waves is an integer multiple of the wavelength in the medium, or <math>(2 * \text{thickness} * n) = m * \lambda</math>, where <math>m</math> is an integer (for the first order, <math>m=1</math>), thickness is the thickness of the oil layer, <math>n</math> is the index of refraction of the oil, and <math>\lambda</math> is the wavelength of light in the oil.</p>	<p>Step 2: Recall that constructive interference occurs when the path difference between the two reflected waves is an integer multiple of the wavelength in the medium, or <math>(2 * \text{thickness} * n) = m * \lambda</math>, where <math>m</math> is an integer (for the first order, <math>m=1</math>), thickness is the thickness of the oil layer, <math>n</math> is the index of refraction of the oil, and <math>\lambda</math> is the wavelength of light in the oil.</p>
	<p>Step 5: Solve for <math>\lambda</math> in oil: <math>\lambda</math> in oil = <math>1500 \text{ nm} / 1.5 = 1000 \text{ nm}</math>.</p>	<p>Step 5: Apply the formula for higher orders of interference (<math>m &gt; 0</math>) to find wavelengths within the visible spectrum. For <math>m=1</math>, the formula becomes <math>(2 * \text{thickness} * n) = (1 + 1/2) * \lambda</math>, which simplifies to <math>(2 * 500 \text{ nm} * 1.5) = 1.5 * \lambda</math>. This gives <math>1500 \text{ nm} = 1.5 * \lambda</math>, so <math>\lambda = 1000 \text{ nm}</math>. <b>This wavelength is still outside the visible spectrum.</b></p>
	<p>Step 6: Since the index of refraction of oil is 1.5, the wavelength in air would be <math>\lambda</math> in air = <math>\lambda</math> in oil / <math>n = 1000 \text{ nm} / 1.5 = 666.67 \text{ nm}</math>.</p>	<p>Step 6: Continue with <math>m=2</math>, the formula becomes <math>(2 * \text{thickness} * n) = (2 + 1/2) * \lambda</math>, which simplifies to <math>(2 * 500 \text{ nm} * 1.5) = 2.5 * \lambda</math>. This gives <math>1500 \text{ nm} = 2.5 * \lambda</math>, so <math>\lambda = 600 \text{ nm}</math>. <b>This wavelength is within the visible spectrum and is a candidate for the most strongly reflected wavelength.</b></p>
	<p>Step 7: Among the given options, the closest to 666.67 nm is <b>650 nm, which is option D.</b></p>	<p>Step 9: Given that no higher orders of <math>m</math> within the visible spectrum and the options provided yield a wavelength that is more strongly reflected than 600 nm, we conclude that <b>600 nm is the most strongly reflected wavelength within the options given.</b></p>

Figure 12: Derailer-Rerailer Solving a Physics Question. The questions and answers, which were retrieved from the MMLU dataset, are exhibited in the blue box. The red boxes are steps generated via the baseline CoT method and the green boxes are the corrected RP from Rerailer. Mistakes are highlighted in red and corrections are highlighted in green.





Figure 13: Error analysis-Lacking Background Information Global Facts Problem. The questions and answers, which were retrieved from the MMLU dataset, are exhibited in the blue box. The red boxes are steps generated via the baseline CoT method and the green boxes are the corrected CoT from Rerailer.

<p><b>Question:</b> At 30, Anika is <math>\frac{4}{3}</math> the age of Maddie. What would be their average age in 15 years?</p> <p><b>Answer:</b></p> <ul style="list-style-type: none"> <li>- Ground Truth Answer: 50</li> <li>- Raw CoT Answer: 41.25</li> <li>- Rerailer Answer: 41.25</li> </ul>	<p>Step 1: Determine Anika's current age. Since Anika is currently <math>\frac{4}{3}</math> times Maddie's age, we can set up the equation: Anika's age = <math>(\frac{4}{3}) * \text{Maddie's age}</math> Let's denote Maddie's age as M. So, Anika's current age = <math>(\frac{4}{3}) * M</math></p>	<p>Step 1: Determine Anika's current age. Since Anika is currently <math>\frac{4}{3}</math> times Maddie's age, we can set up the equation: Anika's age = <math>(\frac{4}{3}) * \text{Maddie's age}</math> Let's denote Maddie's age as M. So, Anika's current age = <math>(\frac{4}{3}) * M</math></p>
	<p>Step 2: Determine Maddie's current age. Given that Anika is currently 30 years old, we can use the relationship established in step 1 to find Maddie's age. Anika's current age = <math>(\frac{4}{3}) * \text{Maddie's current age}</math> <math>30 = (\frac{4}{3}) * M</math>. To find M, we can multiply both sides by <math>\frac{3}{4}</math>: <math>30 * (\frac{3}{4}) = M</math> <math>22.5 = M</math> So, Maddie's current age (M) is 22.5 years.</p>	<p>Step 2: Determine Maddie's current age. Given that Anika is currently 30 years old, we can use the relationship established in step 1 to find Maddie's age. Anika's current age = <math>(\frac{4}{3}) * \text{Maddie's current age}</math> <math>30 = (\frac{4}{3}) * M</math>. To find M, we can multiply both sides by <math>\frac{3}{4}</math>: <math>30 * (\frac{3}{4}) = M</math> <math>22.5 = M</math> So, Maddie's current age (M) is 22.5 years.</p>
	<p>Step 3: Calculate their average age in 15 years. In 15 years, Anika will be <math>30 + 15 = 45</math> years old. And Maddie will be <math>22.5 + 15 = 37.5</math> years old.</p>	<p>Step 3: Calculate their average age in 15 years. In 15 years, Anika will be <math>30 + 15 = 45</math> years old. And Maddie will be <math>22.5 + 15 = 37.5</math> years old.</p>
	<p>Step 4: Find the average of their ages in 15 years. Average age = <math>(\text{Anika's age in 15 years} + \text{Maddie's age in 15 years}) / 2</math> <math>= (45 + 37.5) / 2</math> <math>= 82.5 / 2</math> <math>= 41.25</math> years</p>	<p>Step 4: Find the average of their ages in 15 years. Average age = <math>(\text{Anika's age in 15 years} + \text{Maddie's age in 15 years}) / 2</math> <math>= (45 + 37.5) / 2</math> <math>= 82.5 / 2</math> <math>= 41.25</math> years</p>

Figure 14: Error analysis-Wrong Ground Truth Math Problem. The questions and answers, which were retrieved from the GSM8K dataset, are exhibited in the blue box. The red boxes are steps generated via the baseline CoT method and the green boxes are the corrected RP from Rerailer.