

Guiding Reasoning in Small Language Models with LLM Assistance

Yujin Kim* Euiin Yi* Minu Kim Se-Young Yun† Taehyeon Kim†

KAIST AI

{yujin399, euiin_mercyii}@kaist.ac.kr, kimtaehyeon610@gmail.com

Abstract

The limited reasoning capabilities of small language models (SLMs) cast doubt on their suitability for tasks demanding deep, multi-step logical deduction. This paper introduces a framework called *Small Reasons, Large Hints (SMART)*, which selectively augments SLM reasoning with targeted guidance from large language models (LLMs). Inspired by the concept of cognitive scaffolding, *SMART* employs a score-based evaluation to identify uncertain reasoning steps and injects corrective LLM-generated reasoning only when necessary. By framing structured reasoning as an optimal policy search, our approach steers the reasoning trajectory toward correct solutions without exhaustive sampling. Our experiments on mathematical reasoning datasets demonstrate that targeted external scaffolding significantly improves performance, paving the way for collaborative use of both SLM and LLM to tackle complex reasoning tasks that are currently unsolvable by SLMs alone. Our code is available at <https://github.com/kimyujii/ScaffoldingReasoning>.

1 Introduction

The significant success of large language models (LLMs) in natural language processing (NLP) has driven researchers and engineers to explore their potential applications in robotics (Driess et al., 2023) and fundamental science (Madani et al., 2023; Thirunavukarasu et al., 2023). Their remarkable reasoning capabilities enable them to tackle complex problems requiring multi-step logical deductions, hypothesis generation, and structured decision-making. A key factor behind LLMs’ strong reasoning ability is their capacity for what aligns with System 2 reasoning—deliberate, step-by-step cognitive processes often associated with human analytical thinking (Kahneman, 2011; Guan

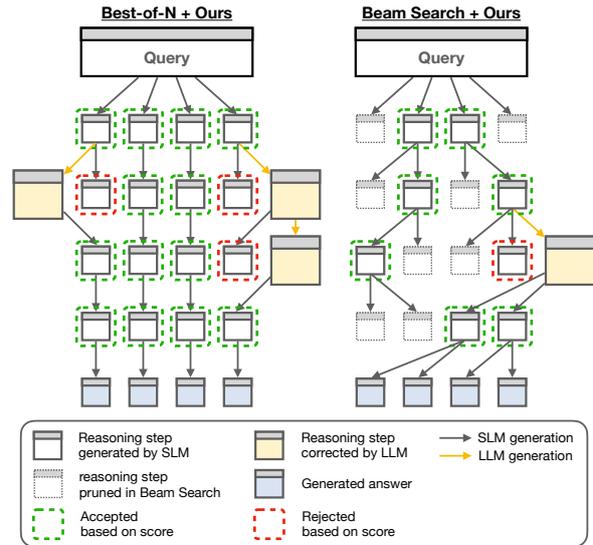


Figure 1: Test-time compute scaling method with our *SMART* framework. During inference, LLM selectively intervenes only at rejected steps of SLM’s reasoning.

et al., 2024). Unlike rapid, heuristic-based System 1 reasoning, which relies on instinct and pattern recognition, System 2-like reasoning—such as test-time compute—is crucial for tasks such as mathematical problem-solving, scientific reasoning, and strategic planning (Snell et al., 2024). Recent studies suggest that LLMs, when properly prompted or trained via reinforcement learning (RL), can exhibit emergent System 2-like behavior, making them highly effective in structured reasoning tasks (Shao et al., 2024; Guo et al., 2025).

However, small language models (SLMs), despite their efficiency, struggle with such structured reasoning due to their limited capacity and lack of emergent cognitive patterns (Mirzadeh et al., 2024). Their reasoning is often shallow, relying more on memorization and surface-level heuristics rather than deep logical deduction (Nikankin et al., 2024). This limitation raises a critical question:

*Equal contribution. †Corresponding author.

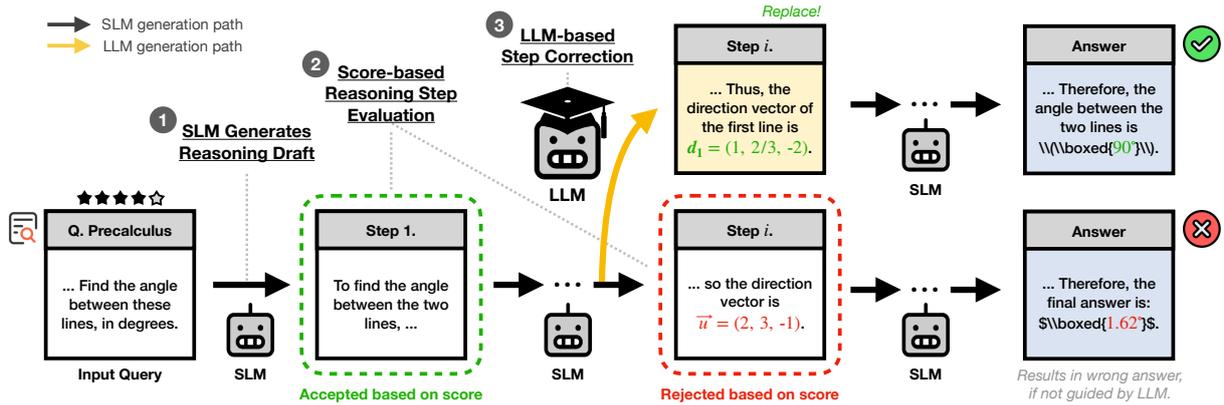


Figure 2: Overview of our *SMART* framework. First, the SLM generates an initial reasoning trajectory, producing step-by-step solutions. Second, after each step is generated, a score-based evaluation mechanism assesses its reliability, determining whether it meets a predefined threshold. Third, for the steps identified as uncertain, the LLM is queried to generate new step, replacing the original SLM step while preserving the preceding SLM-generated steps, and then the process continues until the EOS token or the final answer is reached.

Research question

If SLMs inherently lack reasoning capabilities, does this mean they cannot be utilized in settings that require complex reasoning?

To address this challenge, we introduce a reasoning framework inspired by *scaffolding*, a concept from cognitive science that describes how humans rely on external support—such as guidance from a teacher or structured tools—to complete tasks beyond their independent capability (Wellman and Gelman, 1992; Battaglia et al., 2013; Gerstenberg and Stephan, 2021). Analogously, in our framework, an SLM executes as a primary reasoner while selectively integrating LLM-generated reasoning steps only when it encounters unreliable states (Lake et al., 2017) (Figure 1). For adaptivity, we introduce a score-based evaluation mechanism that determines when external guidance is required. At each reasoning step, the SLM generates a candidate step, which is then assigned a score using a predefined metric—such as a Process Reward Model (PRM) that evaluates reasoning coherence or averaged token-level confidence over the sequence. If the score surpasses a predefined threshold, the SLM proceeds independently. However, if the score falls below this threshold, the framework queries an LLM to generate a replacement step (Figure 2). This approach ensures that external scaffolding occurs only at critical decision points, reducing redundant costs while significantly improving reasoning robustness.

A particularly striking observation in our study is that while increasing the number of sampled rea-

soning trajectories (e.g., Best-of-N sampling) naturally improves performance, this approach is computationally expensive and often lacks principled guidance—frequently relying on lucky guesses rather than a structured search for optimal reasoning (Snell et al., 2024). In contrast, our framework achieves more significant improvements while modifying only specific reasoning steps rather than re-evaluating the entire trajectory. This suggests that rather than globally increasing sampling and scaling compute, selectively correcting key decisions is sufficient to guide the reasoning process toward an optimal solution (Sharma et al., 2023).

From an RL perspective, structured reasoning can be viewed as an optimal policy search, where the model must navigate a sequence of states to reach a correct answer (Chen et al., 2024). Our findings indicate that even when an SLM struggles to determine the next step in a reasoning trajectory, providing guidance only at critical points is enough to keep the overall trajectory on an optimal path. Notably, this selective scaffolding is non-trivial—one might expect that RL-based search methods, such as diffusion models (Ren et al., 2024) or traditional policy optimization (Cetin and Celiktutan, 2022), could naturally handle this process. However, in practice, these methods require extensive exploration or dense reward signals, making them computationally expensive or difficult to scale effectively (Ding et al., 2024). Our results indicate that even without full trajectory optimization, targeted scaffolding can provide sufficient correction, suggesting a promising direction for future integration with RL-based structured reason-

ing frameworks. Our key contributions are:

- We introduce a novel framework called Small Reasons, Large Hints (*SMART*), where an SLM reasons but selectively incorporates LLM-generated reasoning steps (Section 2).
- We conduct experiments on mathematical reasoning benchmark, demonstrating that SLM reasoning, selectively incorporating response from LLM scaffolding, can solve complex reasoning problems that SLMs alone cannot (Section 3).
- We provide a detailed analysis of when LLM scaffolding is beneficial, offering insights into hybrid reasoning systems (Section 4).

2 SMART

2.1 Preliminaries and notations

We formalize the reasoning process as an iterative decision-making problem. Given a query Q , a reasoning trajectory $R = (r_1, r_2, \dots, r_m)$ consists of a sequence of intermediate step r_i , leading to a final answer A . The probability of generating R given Q is modeled as:

$$P(R | Q) = \prod_{i=1}^m P(r_i | Q, r_{<i}). \quad (1)$$

The answer A is then determined as:

$$A = \arg \max_a P(a | Q, R). \quad (2)$$

In this formulation, each r_i represents an intermediate reasoning step, and the correctness of the final answer depends on the entire trajectory R .

2.2 Motivation

SLMs are efficient but frequently fail to generate globally coherent reasoning trajectories due to their limited capacity. A straightforward way to improve reasoning performance is to increase test-time compute (Snell et al., 2024), such as generating multiple trajectories via Best-of-N sampling and selecting the most probable path: $R^* = \arg \max_{R^{(j)}} P(R^{(j)} | Q)$ where $R^{(j)} \sim P_{\text{SLM}}(R | Q)$. However, such approaches introduce significant computational overhead, requiring exponentially more sampling as reasoning complexity increases. Additionally, such selection does not inherently optimize for logical correctness; instead, it favors heuristic shortcuts that may align with fluency but not necessarily with accurate reasoning.

One motivation for our approach is the observation that not all reasoning steps within a query are equally complex (Xue et al., 2024; Liu et al., 2025; Yang et al., 2025). Many intermediate steps in a chain-of-thought process are simple enough for an SLM to handle correctly; however, critical steps—particularly those involving complex calculations or deeper logical inference—pose significant challenges and are more error-prone for smaller models. If we can precisely identify these critical points and selectively engage the stronger reasoning capabilities of an LLM to guide or replace uncertain steps, the remaining reasoning steps performed by the SLM can be corrected, ultimately leading to accurate final predictions.

To address this limitation, we propose selective scaffolding during the reasoning process rather than post-hoc correction of a completed trajectory. As the SLM generates reasoning steps sequentially, each step is assessed for reliability. If a step r_i is identified as unreliable, an LLM guides immediately, replacing it with a revised step r'_i : $r'_i \sim P_{\text{LLM}}(r'_i | Q, r_{<i})$. This assistance directly influences all subsequent steps, leading to a new reasoning trajectory R' that builds upon the corrected information.

2.3 Method

Our framework, termed as Scaffolding Teacher-Assisted Reasoning (*SMART*), refines the reasoning process of an SLM by selectively intervening at critical decision points. The method follows:

1. SLM-generated reasoning draft The SLM generates an initial reasoning trajectory $R = (r_1, r_2, \dots, r_m)$ autoregressively, conditioned on the query Q : $R \sim P_{\text{SLM}}(R | Q)$. Since SLMs lack robust reasoning capabilities, some steps in R may be incorrect or uncertain, leading to errors that propagate throughout the trajectory.

2. Score-based step evaluation To identify unreliable reasoning steps, we introduce a step-wise scoring function that assigns a score $s(r_i | r_{<i}, Q)$ to each step r_i . We consider two scoring methods:

- PRM score:** A learned reward model evaluates the correctness of each step based on prior context, assigning a score $s(r_i | r_{<i}, Q) \in [0, 1]$.
- Token-level confidence (TLC):** Instead of an external reward model, the token-level confidence of r_i is estimated with its averaged token

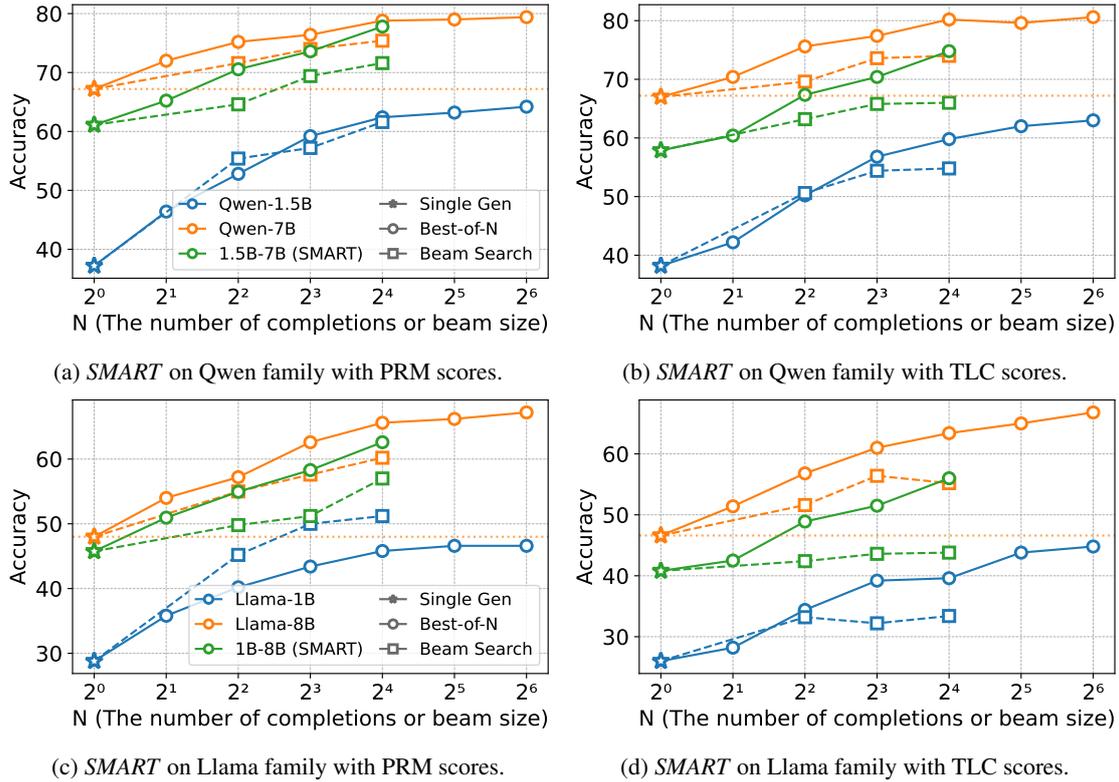


Figure 3: Overall performance results for two different models under various search strategies.

probability: $s(r_i | r_{<i}, Q) = \frac{1}{n} \sum_{j=1}^n P(x_j | x_1, \dots, x_{j-1}, r_{<i}, Q)$ for each token x_i in r_i .

During the autoregressive generation, each step is assessed in real-time for score evaluation. Steps with low scores indicate erroneous reasoning or uncertainty, and are candidates for correction.

3. LLM-based step correction If step r_i 's score falls below a threshold τ , we replace it with an LLM-generated alternative: $r'_i \sim P_{\text{LLM}}(r'_i | Q, r_{<i})$. This selective correction improves reasoning quality with only necessary interventions. See Appendix A for algorithmic details.

Scalability We briefly demonstrate how it can be integrated with test-time compute scaling methods.

- **Best-of-N:** Multiple reasoning trajectories are generated in parallel. SMART is applied to each trajectory independently, ensuring that erroneous steps are corrected within each sampled path.
- **Beam Search:** Given N candidate sequences and a beam width of M , at each reasoning step, the top M sequences are retained based on their scores. Each step r_i within a sequence is evaluated, and if any candidate node falls below the predefined confidence threshold τ , it is replaced with an LLM-generated alternative.

3 Experiments

3.1 Experimental setup

SMART is evaluated on the MATH500 dataset¹. (SLM, LLM) pairs for architectures are tested with (Qwen2.5-1.5B, Qwen2.5-7B) and (Llama3.2-1B, Llama3.2-8B). For both pairs, we employ a process reward model with RLHFlow/Llama3.1-8B-PRM-Deepseek-Data (Dong et al., 2024). Stochastic decoding is implemented with temperature equal to 0.8 following Snell et al. (2024). We report Weighted@N performance for accuracy. Other than mentioned, threshold values τ for PRM score and TLC are mainly set to 0.9 and 0.93. More details regarding the design choices are deferred to Appendix B.

3.2 Main results

To address our core question—whether scaffolding can genuinely improve the performance of an SLM—we firstly examine the overall improvements by SMART framework. Figure 3 shows the performance of SMART across different models

¹MATH500 contains a subset of 500 problems from the MATH benchmark (Lightman et al., 2023a) and is provided by Huggingface (<https://huggingface.co/datasets/HuggingFaceH4/MATH-500>).

Table 1: Performance of *SMART* method using different models, search strategies, and N according to the difficulty level of problem on the MATH500 dataset. The values in parentheses indicate the improvement over the SLM baseline, with colors denoting the direction of change. Performance values exceeding 90.00 are underlined, and the improvement larger than 10.00 are bold-faced. Average values of 16 samples are reported.

Model	Search	N	Lv 1	Lv 2	Lv 3	Lv 4	Lv 5	
Qwen2.5 7B / Qwen2.5 1.5B	Single	1	79.79 (+12.94)	76.39 (+17.60)	72.49 (+24.33)	57.43 (+23.80)	36.84 (+21.34)	
		2	83.41 (+8.13)	80.89 (+13.74)	76.00 (+19.94)	62.16 (+21.34)	41.09 (+22.48)	
	Best-of-N	4	87.50 (+4.68)	85.01 (+9.31)	83.20 (+18.56)	68.08 (+18.86)	45.61 (+22.48)	
		8	89.50 (+3.45)	87.23 (+4.48)	87.40 (+15.25)	73.05 (+16.20)	47.98 (+22.03)	
		16	<u>91.85 (+3.45)</u>	<u>90.00 (+6.10)</u>	89.50 (+11.90)	76.95 (+19.55)	50.35 (+20.50)	
		32	<u>93.00 (+4.60)</u>	<u>91.10 (+5.50)</u>	88.60 (+7.60)	79.70 (+21.10)	51.50 (+20.90)	
		4	83.70 (-4.70)	80.00 (+3.30)	78.10 (+14.30)	60.90 (+5.40)	40.30 (+16.40)	
	Beam Search	8	86.00 (-2.40)	81.10 (+0.00)	78.10 (+5.70)	64.10 (+13.30)	44.80 (+19.40)	
		16	<u>93.00 (+2.30)</u>	85.60 (+2.30)	81.90 (+5.70)	64.80 (+9.30)	47.00 (+14.90)	
		32	<u>93.00 (+2.30)</u>	85.60 (+2.30)	81.90 (+5.70)	64.80 (+9.30)	47.00 (+14.90)	
	Llama3.1 8B / Llama3.2 1B	Single	1	79.58 (+20.72)	66.66 (+20.10)	55.40 (+21.15)	40.56 (+20.63)	20.22 (+11.52)
			2	84.59 (+10.87)	72.56 (+16.52)	61.84 (+20.09)	45.96 (+21.24)	24.21 (+13.45)
Best-of-N		4	85.74 (+13.41)	78.34 (+17.43)	67.86 (+20.23)	50.10 (+19.82)	28.00 (+18.10)	
		8	88.35 (+11.65)	82.78 (+19.00)	72.85 (+19.55)	54.68 (+19.98)	30.05 (+20.95)	
		16	89.55 (+12.25)	86.70 (+16.13)	76.65 (+20.75)	58.20 (+21.60)	34.70 (+20.40)	
		32	<u>90.70 (+15.40)</u>	85.60 (+15.10)	81.90 (+26.40)	59.40 (+25.50)	36.60 (+25.10)	
		4	83.00 (-4.00)	70.00 (+7.80)	61.90 (+11.40)	43.00 (+6.30)	22.40 (+6.00)	
Beam Search		8	81.40 (-7.00)	74.40 (+4.40)	59.00 (-3.90)	43.00 (+3.20)	26.90 (+10.50)	
		16	<u>83.70 (+0.00)</u>	81.10 (+10.00)	69.50 (+5.70)	50.00 (+9.00)	29.10 (+8.20)	
		32	<u>83.70 (+0.00)</u>	81.10 (+10.00)	69.50 (+5.70)	50.00 (+9.00)	29.10 (+8.20)	

Table 2: Comparison of *SMART* with the target LLM (used as the teacher in scaffolding) for the Qwen model family, using Best-of-N strategy, and PRM as score. Each cell shows the percentage of our performance to that of the target LLM (i.e., 100% means *SMART* brings the identical performance with that of LLM). Values in the parentheses indicate the relative performance of SLM against the target LLM. Performance values over 90% are underlined. We report average values of 16 samples.

N	Lv 1	Lv 2	Lv 3	Lv 4	Lv 5
1	89.93% (75.35%)	89.00% (68.49%)	89.21% (59.27%)	<u>90.98%</u> (53.28%)	<u>90.84%</u> (38.22%)
2	<u>91.68%</u> (82.75%)	<u>91.21%</u> (75.72%)	89.66% (66.13%)	<u>91.70%</u> (60.22%)	<u>95.55%</u> (43.28%)
4	<u>94.67%</u> (89.61%)	<u>92.90%</u> (82.72%)	<u>94.20%</u> (73.18%)	<u>93.43%</u> (67.54%)	<u>95.50%</u> (48.44%)
8	<u>100.0%</u> (95.56%)	<u>97.50%</u> (92.99%)	<u>96.99%</u> (88.43%)	<u>98.24%</u> (88.56%)	<u>98.26%</u> (75.69%)
16	<u>100.0%</u> (96.89%)	<u>98.65%</u> (93.53%)	<u>98.56%</u> (89.17%)	<u>98.61%</u> (85.22%)	<u>99.12%</u> (81.15%)
32	<u>100.0%</u> (99.32%)	<u>99.21%</u> (96.74%)	<u>99.19%</u> (94.55%)	<u>98.43%</u> (90.54%)	<u>99.00%</u> (85.10%)

and scoring methods as the number of completions N increases under Best-of-N and Beam Search.

As shown in Figure 3, *SMART* consistently outperforms the SLM baseline across all settings and rapidly approaches LLM-level performance even with a small number of completions. This trend is evident even in the single-generation scenario $N = 1$, where *SMART* provides a clear accuracy improvement over the solely SLM-generated reasoning process.

The benefit of *SMART* becomes even more pronounced when scaling up the test-time compute. As the number of completions increases under Best-of-N or Beam Search, *SMART* systematically approaches near-LLM accuracy, showcasing that ad-

ditional test-time compute further narrows the gap. This suggests that selective scaffolding is particularly effective in guiding the reasoning trajectory efficiently, reducing the need for excessive test-time compute and full trajectory evaluation. While larger N leads to increasingly better results, the marginal gains diminish at higher N , implying that *SMART* can already achieve strong improvements with moderate test-time compute, without relying on exhaustive search. Comparing the PRM-based and TLC-based scoring methods, both exhibit similar scaling patterns, while PRM shows slightly stronger performance at higher N .

Table 1 further quantifies *SMART*'s performance improvements across five levels of problem diffi-

Table 3: Comparison of corrected step and token ratios using Best-of-N and Beam Search strategies for $N = 16$. Average values of 16 samples are reported.

Model	Search	Correction	Lv 1	Lv 2	Lv 3	Lv 4	Lv 5	Total
Qwen2.5 7B / Qwen2.5 1.5B	Best-of-N (with $N = 16$)	Step ratio	0.1830	0.2948	0.4277	0.5499	0.7331	0.4959
		Token ratio	0.1338	0.2112	0.2902	0.3555	0.4235	0.2021
	Beam Search (with $N = 16$)	Step ratio	0.0212	0.0596	0.1001	0.1470	0.2309	0.1331
		Token ratio	0.0052	0.0148	0.0228	0.0295	0.0435	0.0271
Llama3.1 8B / Llama3.2 1B	Best-of-N (with $N = 16$)	Step ratio	0.2500	0.3876	0.5388	0.6390	0.7965	0.5815
		Token ratio	0.1645	0.2256	0.3026	0.3567	0.3927	0.3149
	Beam Search (with $N = 16$)	Step ratio	0.0429	0.0994	0.1554	0.2185	0.2868	0.1870
		Token ratio	0.0059	0.0156	0.0257	0.0305	0.0396	0.0272

culty. As expected, the baseline SLM struggles with higher-level problems, but *SMART*’s targeted scaffolding consistently elevates its accuracy. This effect is most pronounced at the more complex levels (Lv 4 and Lv 5), where reasoning errors occur more frequently. In all cases, Best-of-N sampling proves particularly effective for complex tasks, especially N is higher ($N = 16, 32$). Beam Search follows a similar pattern, though higher N sometimes yield diminishing returns at lower difficulty levels—likely because exhaustive search is less critical when problems are simpler. Interestingly, we observe *SMART* with Best-of-N consistently outperforming *SMART* paired with Beam Search across all settings. This is because the Best-of-N retains a global search space: Under Best-of-N, *SMART* ensures that even partially flawed beams are recoverable by replacing low-scoring nodes with improved alternatives. This not only preserves diversity in beam candidates but also mitigates catastrophic failure from early missteps. Whereas the Beam Search allows LLM intervention only on unpruned (candidate) nodes, restricting its search space for scaffolding. We defer further analysis to Section 3.3.

Table 2 compares *SMART*’s accuracy against the full LLM’s performance, showing that *SMART* often achieves near-LLM results while relying only on selective scaffolding. At lower difficulty levels (Lv 1 and 2), *SMART* already attains over 90% of the LLM’s accuracy, even for smaller N . As the tasks become more challenging, the performance gap between the SLM and full LLM widens, yet *SMART* remains highly effective, maintaining above 95% of the LLM’s accuracy at $N = 8, 16$, and 32 . While Beam Search also benefits from LLM interventions, its narrower search space sometimes limits exploration compared to Best-of-

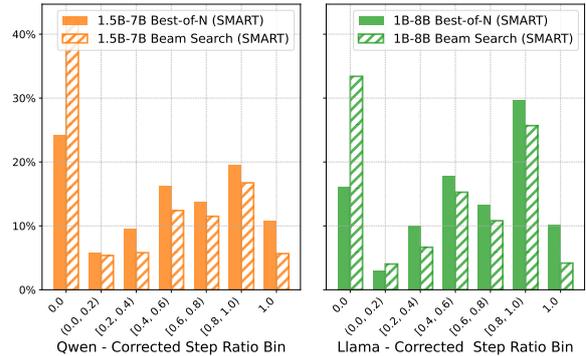


Figure 4: Step ratio distribution across Qwen (left) and Llama (right) models upon different search strategies. We report average values of 16 samples.

N , resulting in smaller gains at higher difficulty levels. See Section C.1 for more results.

Taken together, our results reveal that *SMART* not only corrects erroneous intermediate reasoning steps generated by the SLM but also scales effectively with additional test-time compute. The greatest improvements occur at higher difficulty levels, underscoring the benefits of targeted corrections for complex mathematical reasoning tasks. By allowing the SLM to function autonomously on simpler problems while selectively scaffolding harder ones, *SMART* demonstrates that SLMs, which have been considered unsuitable for complex reasoning, can effectively contribute to solving such tasks when supported by minimal, yet strategic LLM guidance.

3.3 Analysis

We analyze the impact of LLM scaffolding by quantifying the extent of intervention in the reasoning process. In doing so, we aim at verifying whether *SMART* intervenes only when and where it is most needed. To this end, we introduce two key metrics—the corrected step ratio and the corrected token ratio—which measure the frequency and extent

of LLM modifications, respectively.

Let $\mathcal{S} = \{1, \dots, m\}$ be the set of all reasoning steps generated by the SLM. We define the corrected steps as a subset $\mathcal{S}_c \subseteq \mathcal{S}$, where each corrected step r'_i for $i \in \mathcal{S}_c$ is a modification of the original step r_i due to LLM scaffolding. The *corrected step ratio* is then defined as:

$$(\text{Corrected step ratio}) := \frac{|\mathcal{S}_c|}{|\mathcal{S}|}, \quad (3)$$

which reflects how frequently the LLM intervenes in the reasoning trajectory. For Beam Search, we only consider the final reasoning steps. This means that we do not calculate the pruned steps on *corrected step ratio*. To measure *extent* of these corrections, we also introduce the *corrected token ratio*. Given a tokenization function $\text{Token}(\cdot)$ that returns the token count of a reasoning step, we compute the corrected token ratio as:

$$(\text{Corrected token ratio}) := \frac{\sum_{l \in \mathcal{S}_c} \text{Token}(r'_l)}{\sum_{j \in \mathcal{S}} \text{Token}(r_j)}. \quad (4)$$

This metric captures how extensively the LLM’s interventions rewrite the original reasoning content.

Table 3 presents the corrected step and token ratios for varying levels of problem difficulty. At lower levels, the SLM requires only minimal intervention, as evidenced by low ratios of both corrected step and token ratios. However, as the task difficulty increases (Lv 4 and 5), LLM intervention becomes more frequent. This suggests that the LLM effectively detects and intervenes when the SLM struggles, selectively guiding reasoning only when necessary. The increasing frequency of intervention for higher levels aligns with our expectation that more complex reasoning require greater external support. LLM intervention—measured by both the corrected step and token ratios—is significantly lower under Beam Search compared to Best-of-N, even at higher reasoning levels. This discrepancy can be from the structural properties of Beam Search. Since Beam Search maintains a tree-like structure, a single correction to a parent node can automatically propagate changes to its multiple child paths. In contrast, Best-of-N treats each reasoning path independently, requiring the LLM to individually intervene in each path to achieve similar corrections. This characteristic holds consistently across different model families.

To gain further insights, Figure 4 illustrates the distribution of samples across corrected step ratios for each model family and test-time scaling

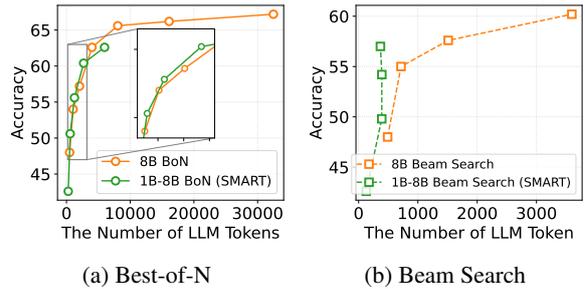


Figure 5: Accuracy according to the number of LLM tokens used during inference.

compute method. In this part, we consider Beam Search reasoning samples as independent samples. If there is two sample which is sharing same parents, we regard two reasoning paths as two independent path. For both model families, a large proportion of samples exhibit no LLM intervention, indicating that *SMART* predominantly allows the SLM to operate independently. Beam Search results are heavily concentrated at the 0.0 token correction ratio, suggesting that errors tend to propagate less aggressively within its constrained search space, consistent with its pruning of less-promising candidates early on and thus requiring fewer large-scale corrections. In contrast, Best-of-N shows a broader distribution across correction ratios, with a significant number of samples exhibiting high correction levels (e.g., 0.8–1.0 and 1.0 bins). This suggests that Best-of-N not only invites more frequent intervention but also allows for more aggressive correction when necessary.

These findings validate the effectiveness of *SMART*, where the LLM only corrects key reasoning steps rather than applying exhaustive modifications. Our method preserves the SLM’s independence on easier problems while ensuring adequate support for more complex reasoning tasks.

4 Discussion

4.1 Cost-effective and practical usage in *SMART*

In the previous sections, we demonstrated that *SMART* effectively bridges the performance gap between an SLM and an LLM by selectively integrating LLM-generated reasoning steps in a controlled manner. Our results indicate that SLMs, despite their limited reasoning capacity, can achieve near-LLM performance with minimal but well-targeted LLM scaffolding. This suggests a viable deployment strategy where an SLM operates as the pri-

mary model, with an LLM providing corrective reasoning only when necessary.

A key application of this framework is in scenarios where an SLM runs locally on a device, while an LLM is accessible remotely via an API. Recently, such collaborative frameworks have drawn increased attention due to their potential to simultaneously optimize both accuracy and resource efficiency (Narayan et al., 2025). However, due to constraints such as cost, latency, or privacy, frequent LLM queries may be impractical, making it crucial to minimize intervention while maintaining strong reasoning performance. *SMART* addresses this challenge by enabling an adaptive mechanism that dynamically determines when an LLM query is needed, thereby reducing unnecessary API calls while preserving accuracy.

As illustrated in Figure 5, *SMART* effectively reduces LLM token usage while maintaining comparable accuracy to an LLM alone. For the Best-of-N strategy, our approach shows efficiency similar to directly employing an LLM. With Beam Search, we consistently observe substantial reductions in LLM token usage—up to 90%—without sacrificing accuracy. This improvement occurs because the LLM intervenes selectively, only when the SLM’s top-ranking candidate reasoning paths fail to meet a specified confidence threshold, thus minimizing overall LLM intervention. Furthermore, as N increases, the number of LLM tokens utilized by *SMART* remains stable in low number, due to the fixed number (M) of top-ranking paths. These observations suggest that our framework is particularly favorable for tree-structured search strategies, highlighting its potential to significantly reduce token-based costs associated with LLM API usage. Additional results using the Qwen model family are provided in Section C.2.

4.2 Comparison between PRM and TLC scores

In Figure 3, we observe that PRM slightly outperforms TLC at higher values of N . This is because the PRM score explicitly evaluates the correctness of each reasoning step based on predefined criteria, making it a more structured and interpretable measure of logical validity. As a result, PRM is particularly useful for interpretability and control, aligning well with predefined reward structures. However, evaluating the PRM score becomes more challenging for general problems across various domains and tasks. In contrast, the TLC score demon-

strates reasoning consistency and robustness, which may better reflect generalization across diverse domains. Unlike PRM, which relies on predefined evaluation criteria, TLC focuses on the model’s internal certainty about its predictions. Nonetheless, token-level confidence scoring has its limitations. Specifically, it fails to account for error propagation in Chain-of-Thought (CoT) reasoning, where a strongly confident but incorrect step can mislead subsequent steps. Self-confidence-based scores, such as the TLC score, remain an active area of research (Wang and Zhou, 2024). As models increasingly exhibit self-awareness, intrinsic confidence can be used as a score (Ji et al., 2023). If self-awareness research continues to advance, this approach appears promising. Nevertheless, if computational resources are available, PRM can still be utilized as the primary scoring method.

5 Related works

Application of SLMs SLMs are increasingly employed in resource-constrained environments, such as mobile devices and embedded systems, where deploying LLMs is impractical due to computational and memory limitations (Blog, 2024). SLMs’ efficiency makes them well-suited for real-time applications, including chatbots, live captioning, and gaming interactions, by enabling low-latency processing (Xu et al., 2025). They are also utilized in cloud-integrated systems, where they enhance automation and personalized services while reducing infrastructure costs (Talluri et al., 2024). Despite these advantages, SLMs face significant limitations, particularly in structured reasoning tasks (Bi et al., 2024). Their constrained model size leads to performance saturation during training, making it difficult to map low-dimensional outputs to high-rank contextual probability distributions. As a result, they struggle with nuanced language comprehension and exhibit reduced accuracy on complex tasks compared to larger models (Godey et al., 2024; Yi et al., 2024).

Multi-LM collaboration Collaborative decoding enables multiple language models to work together to enhance text generation (Shen et al., 2024a). At the token level, different models contribute at various points in the sequence by employing a small model to draft tokens and a large model to verify them for efficient decoding (Leviathan et al., 2023), utilizing a mixture of expert models aligned with specific tasks to improve perfor-

mance (Liu et al., 2021), or integrating a verifier model to refine generation (Lightman et al., 2023b). To address complex reasoning tasks, collaborative decoding has evolved to the step level. For hallucination mitigation, a primary model generates reasoning steps while another model verifies their validity (Feng et al., 2024). In multi-agent setups, multiple smaller models assume different roles, contributing individual insights in a structured debate. Arguments are then synthesized to reach a final conclusion (Shen et al., 2024b). While multi-LLM collaboration has been widely explored, prior work has primarily focused on models with comparable capabilities. There is limited research on leveraging two models with large discrepancies in reasoning capacity, such as an SLM and an LLM.

Test-time compute scaling Recent work has explored test-time compute scaling as an alternative to fine-tuning or distillation, enhancing SLM reasoning by generating multiple reasoning paths or refining outputs iteratively (Snell et al., 2024; Ehrlich et al., 2025; Muennighoff et al., 2025). While increasing test-time computation improves performance, SLMs inherently lack the capacity to solve complex reasoning problems, even with extensive compute scaling. This fundamental limitation stems from their restricted expressivity, making it impossible for SLMs to reach LLM-level reasoning solely through brute-force search or increased sampling. Additionally, test-time scaling incurs higher operational costs and computational overhead, limiting its practicality for real-time applications. Research has explored selective test-time scaling, applying computation only where necessary to balance cost and performance while acknowledging the constraints of SLM capacity.

Confidence-based score Recent studies have demonstrated that the intrinsic confidence of a model’s predictions can serve as a reliable indicator of reasoning quality. For instance, Wang and Zhou (2024) found that in large language models, the presence of a chain-of-thought (CoT) reasoning path correlates with higher confidence in the model’s decoded answer. Moreover, Huang et al. (2025) introduced a self-calibration method that dynamically adjusts the number of sampling responses based on the model’s confidence in its predictions.

6 Conclusion

This paper presents a novel *SMART* (Scaffolding Teacher-Assisted Reasoning) framework that enhances structured, multi-step reasoning by selectively incorporating LLM interventions into an SLM’s reasoning process. *SMART* dynamically assesses each reasoning step using a score-based mechanism and replaces unreliable steps with LLM-generated alternatives only at critical junctures. Experimental results indicate that SLMs, despite their limited reasoning capacity, can achieve near-LLM performance with minimal but well-targeted LLM intervention, enabling SLMs to tackle complex reasoning problems that are unsolvable when operating alone. This suggests a viable deployment strategy where an SLM functions as the primary model, with an LLM providing corrections only when necessary—a practical avenue for deploying SLMs where on-device operation is desired—thereby avoiding the overhead of exhaustive compute scaling.

Limitations

While our method demonstrates the feasibility and effectiveness of *SMART* framework, there is room for optimization to improve its practical deployment. Currently, the degree of LLM scaffolding is not explicitly controlled but is instead regulated through indirect thresholding via PRM and TLC scores. Although this provides a mechanism for adaptive intervention, it lacks fine-grained control, which could allow for more precise optimization based on task complexity or computational constraints. Additionally, our current criteria for triggering LLM scaffolding may not be optimal, and alternative strategies could further minimize computational overhead while preserving or even improving accuracy. Future work could explore more adaptive scaffolding mechanisms, such as RL-based policies or uncertainty-aware scaffolding, to enhance adaptability across different reasoning tasks.

Acknowledgement

This work was supported by Artificial intelligence industrial convergence cluster development project funded by the Ministry of Science and ICT(MSIT, Korea) & Gwangju Metropolitan City.

References

- Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. 2013. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332.
- Jing Bi, Yuting Wu, Weiwei Xing, and Zhenjie Wei. 2024. Enhancing the reasoning capabilities of small language models via solution guidance fine-tuning. *arXiv preprint arXiv:2412.09906*.
- IBM Think Blog. 2024. [Small language models](#). Accessed: 2025-02-16.
- Edoardo Cetin and Oya Celiktutan. 2022. Policy gradient with serial markov chain reasoning. *Advances in Neural Information Processing Systems*, 35:8824–8839.
- Guoxin Chen, Kexin Tang, Chao Yang, Fuying Ye, Yu Qiao, and Yiming Qian. 2024. Seer: Facilitating structured reasoning and explanation via reinforcement learning. *arXiv preprint arXiv:2401.13246*.
- Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. 2024. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *arXiv preprint arXiv:2405.16173*.
- Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. 2024. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*.
- Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.
- Ryan Ehrlich, Bradley Brown, Jordan Juravsky, Ronald Clark, Christopher Ré, and Azalia Mirhoseini. 2025. Codemonkeys: Scaling test-time compute for software engineering. *arXiv preprint arXiv:2501.14723*.
- Shangbin Feng, Weijia Shi, Yike Wang, Wenxuan Ding, Vidhisha Balachandran, and Yulia Tsvetkov. 2024. Don’t hallucinate, abstain: Identifying llm knowledge gaps via multi-llm collaboration. *arXiv preprint arXiv:2402.00367*.
- Tobias Gerstenberg and Simon Stephan. 2021. A counterfactual simulation model of causation by omission. *Cognition*, 216:104842.
- Nathan Godey, Éric de la Clergerie, and Benoît Sagot. 2024. Why do small language models underperform? studying language model saturation via the softmax bottleneck. *arXiv preprint arXiv:2404.07647*.
- Melody Y Guan, Manas Joglekar, Eric Wallace, Saachi Jain, Boaz Barak, Alec Heylar, Rachel Dias, Andrea Vallone, Hongyu Ren, Jason Wei, et al. 2024. Deliberative alignment: Reasoning enables safer language models. *arXiv preprint arXiv:2412.16339*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiabin Huang. 2025. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*.
- Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating hallucination in large language models via self-reflection. *arXiv preprint arXiv:2310.06271*.
- Daniel Kahneman. 2011. Thinking, fast and slow. *Farar, Straus and Giroux*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023a. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023b. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*.
- Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. *arXiv preprint arXiv:2105.03023*.
- Yuliang Liu, Junjie Lu, Zhaoling Chen, Chaofeng Qu, Jason Klein Liu, Chonghan Liu, Zefan Cai, Yunhui Xia, Li Zhao, Jiang Bian, et al. 2025. Adaptivestep: Automatically dividing reasoning step through model confidence. *arXiv preprint arXiv:2502.13943*.
- Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. 2023. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106.

- Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.
- Avanika Narayan, Dan Biderman, Sabri Eyuboglu, Avner May, Scott Linderman, James Zou, and Christopher Re. 2025. Minions: Cost-efficient collaboration between on-device and cloud language models. *arXiv preprint arXiv:2502.15964*.
- Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2024. Arithmetic without algorithms: Language models solve math with a bag of heuristics. *arXiv preprint arXiv:2410.21272*.
- Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. 2024. Diffusion policy optimization. *arXiv preprint arXiv:2409.00588*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Pratyusha Sharma, Jordan T Ash, and Dipendra Misra. 2023. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. *arXiv preprint arXiv:2312.13558*.
- Shannon Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. 2024a. Learning to decode collaboratively with multiple language models. *arXiv preprint arXiv:2403.03870*.
- Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. 2024b. Small llms are weak tool learners: A multi-llm agent. *arXiv preprint arXiv:2401.07324*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- Subhash Talluri, Carlos Fernandez Casares, Deepak Rupakula, Mohammad Zoualfaghari, and Parham Beheshti. 2024. [Aws for industries: Opportunities for telecoms with small language models](#). Accessed: 2025-02-16.
- Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.
- Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. *arXiv preprint arXiv:2402.10200*.
- Henry M Wellman and Susan A Gelman. 1992. Cognitive development: foundational theories of core domains. *Annual review of psychology*.
- Borui Xu, Yao Chen, Zeyi Wen, Weiguo Liu, and Bingsheng He. 2025. Evaluating small language models for news summarization: Implications and factors influencing performance. *arXiv preprint arXiv:2502.00641*.
- Shangzi Xue, Zhenya Huang, Jiayu Liu, Xin Lin, Yuting Ning, Binbin Jin, Xin Li, and Qi Liu. 2024. Decompose, analyze and rethink: Solving intricate problems with human-like reasoning cycle. *Advances in Neural Information Processing Systems*, 37:357–385.
- Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*.
- Euiin Yi, Taehyeon Kim, Hongseok Jeung, Du-Seong Chang, and Se-Young Yun. 2024. Towards fast multilingual llm inference: Speculative decoding and specialized drafters. *arXiv preprint arXiv:2406.16758*.

A Implementation details of SMART

Algorithm 1 SMART

Require: Query Q , max steps M , scoring function $s(\cdot)$, threshold τ , Token count function $\text{Token}(\cdot)$, max token length L_{\max}

Ensure: Reasoning sequence $R = (r_1, \dots, r_m)$ and final answer $A = \phi(Q, R)$

```

1:  $R \leftarrow []$ 
2: for  $i = 1$  to  $M$  do
3:    $r_i^{(SLM)} \leftarrow P_{\text{SLM}}(r_i \mid Q, r_1, \dots, r_{i-1})$ 
4:    $s \leftarrow s(r_i^{(SLM)} \mid Q, R)$ 
5:    $r_i \leftarrow \begin{cases} P_{\text{LLM}}(r_i \mid Q, r_1, \dots, r_{i-1}) & \text{if } s < \tau, \\ r_i^{(SLM)} & \text{otherwise.} \end{cases}$ 
6:   Append  $r_i$  to  $R$ 
7:   if  $r_i = \text{EOS}$  or  $\sum_{j=1}^i \text{Token}(r_j) \geq L_{\max}$  then
8:     break
9:   end if
10: end for
11: return  $R$ 

```

We present the implementation details of our **Scaffolded Teacher-Assisted Reasoning (SMART)** framework, which selectively integrates LLM intervention to enhance the structured reasoning capabilities of SLMs. The reasoning process is formulated as an iterative decision-making problem, where the SLM generates reasoning steps autonomously, and an LLM intervenes selectively based on score-based evaluation.

A.1 Reasoning Process

Given an input query Q , the objective is to construct a reasoning trajectory $R = (r_1, \dots, r_m)$ that leads to a final answer A . The SMART framework operates as follows:

1. **SLM Step Generation:** The SLM generates an intermediate reasoning step $r_i^{(SLM)}$ autoregressively:

$$r_i^{(SLM)} \sim P_{\text{SLM}}(r_i \mid Q, r_1, \dots, r_{i-1}) \quad (5)$$

where P_{SLM} represents the probability distribution modeled by the SLM.

2. **Step Scoring:** Each generated step is evaluated with a scoring function $s(r_i^{(S)} \mid Q, r_{<i})$, which assesses its reliability based on predefined heuristics or reward models.

3. **Selective LLM Intervention:** If the score s is lower than the threshold τ , the reasoning step is corrected by querying the LLM:

$$r'_i \sim P_{\text{LLM}}(r_i \mid Q, r_1, \dots, r_{i-1}) \quad (6)$$

$$r_i \leftarrow r'_i \quad (7)$$

Otherwise, the SLM-generated step is retained.

After that, append reasoning step r_i to reasoning trajectory R

4. **Termination Criteria:** The iterative reasoning process continues until one of the following conditions is met:

- The model generates an **end-of-the-sequence (EOS)** token.
- The cumulative token count surpasses L_{\max} , ensuring computational efficiency.

5. **Output Reasoning Trajectory:** The final reasoning sequence R is returned as the output.

B Experimental details

B.1 Detailed experiment setup

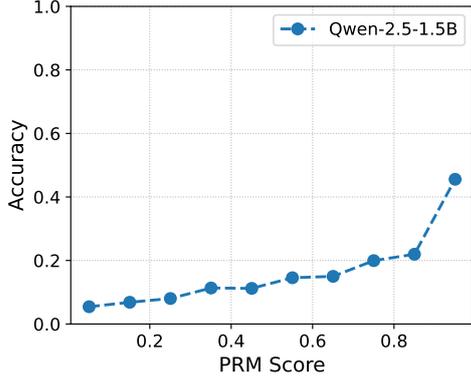
We use Qwen family (Qwen2.5-1.5B, Qwen2.5-7B) and Llama family (Llama3.2-1B, Llama3.2-8B) in our experiments. For experiments, we use four A5000 GPUs using inference package VLLM(Kwon et al., 2023).

B.2 Dataset details

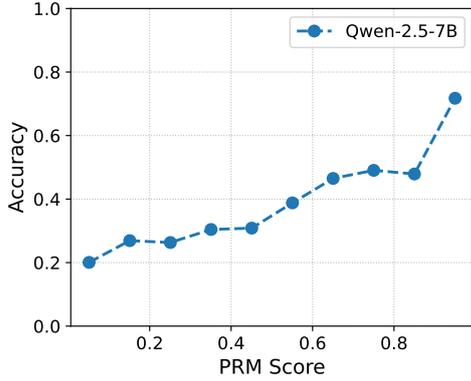
The MATH500 dataset is a curated subset of the MATH dataset, comprising 500 problems selected from MATH dataset(Lightman et al., 2023a), to evaluate mathematical reasoning in models. These problems are categorized by subject and difficulty level, facilitating a comprehensive assessment of model performance across various mathematical domains. Each problem is accompanied by a detailed step-by-step solution, enabling the evaluation of models' problem-solving processes. In our experiments, we utilize the MATH500 dataset to benchmark the performance of our models, ensuring a rigorous evaluation of their mathematical reasoning abilities.

B.3 Threshold selection for SMART

In Figure 6, the relationship between the PRM score and prediction accuracy differs notably between the two model sizes. For the 7B model



(a) Accuracy of Qwen-2.5-1.5B according to PRM score.



(b) Accuracy of Qwen-2.5-7B according to PRM score.

Figure 6: Accuracy according to PRM score. Qwen-2.5-1.5B and Qwen-2.5-7B are used for predicting reasoning steps.

(Qwen-2.5-7B) in Figure 6 (2), the PRM score strongly correlates with accuracy, indicating that higher PRM values reliably predict correct outcomes. In contrast, for the 1.5B model (Qwen-2.5-1.5B), accuracy increases only marginally across most PRM score ranges and remains low until approximately 0.9. Based on this observation, we set a PRM threshold of 0.9 to trigger LLM intervention, as reasoning steps with a score of 0.9 or lower exhibit a high likelihood of being incorrect, necessitating the intervention of the LLM.

B.4 Evaluation metrics

The Weighted@N metric selects the answer with the highest total reward by aggregating scores across identical responses. It prioritizes high-quality outputs by reinforcing frequently occurring, high-reward solutions. Formally, the selected answer A is given by:

$$A_{\text{weighted}} = \arg \max_a \sum_{i=1}^N \mathbb{I}(A_i = a) \cdot s(R_i | Q)$$

where $s(R_i | Q)$ is a scoring function which evaluates the correctness of reasoning trajectory R_i given query Q . The trajectory R_i consists of a sequence of intermediate steps leading to the final answer A_i . We report Weighted@N as it shows consistently high performance among other metrics such as majority voting.

B.5 Zero-shot prompt for evaluation

We use zero-shot prompting method to evaluate following . This prompt instructs to follow a structured step-by-step format, where each step is separated by double line breaks ($\backslash n \backslash n$). This separation facilitates evaluation and correction. The response always concludes with a boxed final answer for clarity.

```

<|im$_$start|>system
Solve the following math problem
efficiently and
clearly:

- For simple problems (2 steps or fewer)
:
Provide a concise solution with minimal
explanation.

- For complex problems (3 steps or more)
:
Use this step-by-step format:

## Step 1: [Concise description]
[Brief explanation and calculations]

## Step 2: [Concise description]
[Brief explanation and calculations]

...

Regardless of the approach, always
conclude
with:

Therefore, the final answer is: $\boxed{
answer}$ .
I hope it is correct.

Where [answer] is just the final number
or
expression that solves the problem. <|im$_$end|>

<|im$_$start|>user
{Problem}<|im$_$end|>

<|im$_$start|>assistant

```

C Additional results

C.1 Comparison of *SMART* with the target LLM

We further evaluate *SMART*'s performance using Beam Search on Qwen family (Table 4) and extend the analysis to the Llama model family (Table 5). The overall trend remains consistent—*SMART* achieves near-LLM accuracy at lower difficulty levels, exceeding 90% even for moderate N. As task difficulty increases, performance remains competitive. While Beam Search provides structured decoding, Best-of-N tends to yield higher gains, particularly at higher difficulty levels. Notably, for Llama, *SMART* occasionally surpasses LLM performance, highlighting its robustness across models and search strategies.

C.2 Accuracy according to LLM token usage

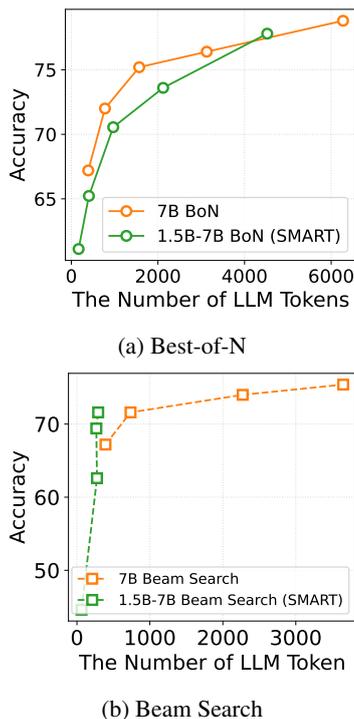


Figure 7: Accuracy according to LLM token usage on Qwen family.

We further provide result on Qwen family in Figure 7. As previously mentioned, for beam search, our method shows efficiency to achieve similar accuracy level in terms of LLM token usage compared to only employing LLM. However, the result of Best-of-N is quite different, where for Qwen, our method uses comparatively more LLM tokens compared to Llama.

D Further Discussions

D.1 The relationship between threshold values of PRM and TLC score and accuracy and token ratio

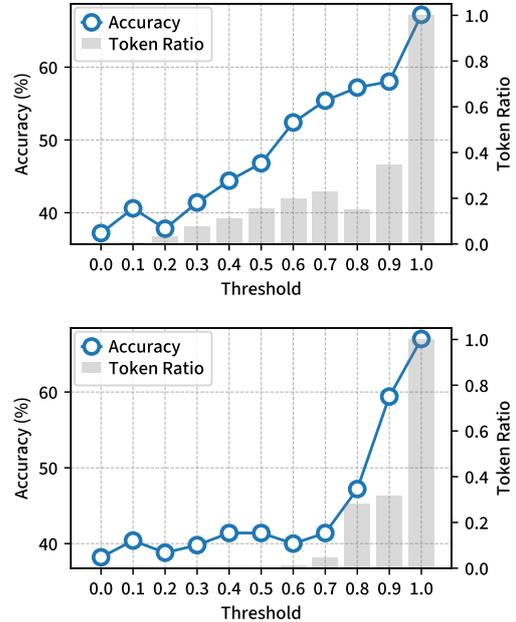


Figure 8: The relationship between accuracy and token usage ratio according to threshold values. The upper plot uses the PRM score, while the lower plot uses the TLC score.

Although our work is, to the best of our knowledge, the first to focus on boosting an SLM's reasoning ability through limited LLM assistance, further optimization of LLM scaffolding strategies remains an open challenge. Currently, the scaffolding rate is controlled by a predefined threshold, which dictates when an SLM-generated step should be replaced. However, an optimal policy for LLM calls may require a more adaptive mechanism that learns when scaffolding is most beneficial. As illustrated in Figure 8, there exists an inherent trade-off between overall accuracy and the proportion of reasoning tokens contributed by the LLM. A promising direction for future research is developing RL or meta-learning strategies to dynamically adjust scaffolding thresholds based on problem complexity and past success rates.

D.2 Correction Timing of LLM

Beyond simply adjusting how often scaffolding occurs, determining at which step in the reasoning process to intervene may also be crucial. In our implementation, the scaffolding is triggered by a

Table 4: Comparison of *SMART* with the target LLM (used as the teacher in scaffolding) for the Qwen model family, using Beam Search. Each cell shows the percentage of our performance relative to the target LLM (i.e., 100% means *SMART* brings the identical performance with that of LLM). Values in parentheses indicate the relative performance of SLM against the target LLM. Performance values over 90% are underlined.

Approach	N	Lv 1	Lv 2	Lv 3	Lv 4	Lv 5
Beam Search	4	<u>92.28%</u> (97.46%)	88.89% (85.22%)	<u>93.20%</u> (76.13%)	83.77% (76.34%)	84.31% (50.00%)
	8	<u>92.47%</u> (95.05%)	87.96% (87.96%)	<u>91.13%</u> (84.48%)	<u>95.39%</u> (75.60%)	79.01% (44.80%)
	16	<u>100.00%</u> (97.53%)	<u>93.96%</u> (91.44%)	<u>92.44%</u> (86.01%)	87.33% (74.80%)	<u>92.70%</u> (63.31%)

Table 5: Comparison of *SMART* with the target LLM (used as the teacher in scaffolding) for the Llama model family, using different search methods. Each cell shows the percentage of our performance relative to the target LLM (i.e., 100% means *SMART* brings the identical performance with that of LLM). Values in parentheses indicate the relative performance of SLM against the target LLM. Performance values over 90% are underlined.

Approach	N	Lv 1	Lv 2	Lv 3	Lv 4	Lv 5
Best of N	1	<u>97.43%</u> (72.06%)	<u>96.52%</u> (67.42%)	<u>97.66%</u> (60.38%)	<u>103.06%</u> (50.64%)	<u>104.91%</u> (45.14%)
	2	<u>97.82%</u> (79.16%)	<u>95.14%</u> (72.49%)	<u>97.26%</u> (65.72%)	<u>102.28%</u> (55.00%)	<u>104.90%</u> (47.54%)
	4	<u>96.39%</u> (81.35%)	<u>97.59%</u> (77.16%)	<u>94.07%</u> (66.19%)	<u>97.14%</u> (58.70%)	<u>102.75%</u> (52.11%)
	8	<u>99.33%</u> (88.87%)	<u>100.67%</u> (78.38%)	<u>102.56%</u> (66.62%)	<u>104.21%</u> (67.09%)	<u>92.53%</u> (47.73%)
	16	<u>102.60%</u> (89.48%)	<u>101.42%</u> (88.15%)	<u>98.86%</u> (71.14%)	<u>97.12%</u> (70.45%)	<u>94.94%</u> (45.01%)
Beam Search	4	<u>94.68%</u> (100.00%)	88.72% (78.83%)	<u>98.41%</u> (80.29%)	87.40% (74.59%)	81.16% (59.42%)
	8	<u>94.65%</u> (102.79%)	<u>91.74%</u> (86.31%)	<u>91.05%</u> (97.07%)	75.44% (69.83%)	87.91% (53.60%)
	16	<u>92.28%</u> (92.28%)	<u>97.36%</u> (85.35%)	88.99% (81.69%)	<u>90.09%</u> (73.15%)	<u>114.57%</u> (82.28%)

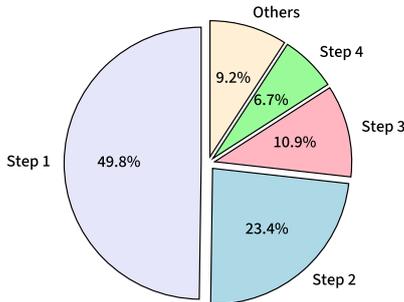


Figure 9: The ratio of SLM’s reasoning steps first corrected by the LLM during reasoning process. We used 16 random single-generation samples using Qwen-2.5-7B/1.5B model family.

score function that does not account for time-step dependencies, but real-world scenarios may well exhibit such dependencies. Indeed, our analysis in Figure 9 shows that the initial steps in the reasoning sequence receives the greatest amount of LLM guidance, suggesting that the LLM effectively identifies and rectifies errors in the early stages of reasoning. This also well aligns with the intuition that early assistance can be particularly beneficial, as unresolved errors may propagate through subsequent steps to degrade reasoning quality.

D.3 Comparison of step-wise score

Table 6: Comparison of PRM and TLC scores using Best-of-N and Beam Search approaches for Qwen2.5 and Llama3.1 models.

Model	Search	PRM Score	TLC Score
Qwen2.5 7B / Qwen2.5 1.5B	Best-of-N	0.8134	0.9039
	Beam Search	0.8637	0.9212
Llama3.1 8B / Llama3.2 1B	Best-of-N	0.8130	0.9451
	Beam Search	0.8837	0.9666

In this section, we compare the average PRM and TLC scores for different models and search strategies. As shown in Table 6, Beam Search consistently achieves higher scores than Best-of-N across both Qwen and Llama model families. This aligns with our main analysis, where Beam Search systematically prunes less-promising candidates early in the reasoning process, retaining only those with relatively higher scores. Consequently, the remaining candidates tend to have higher PRM and TLC scores. This trend is consistent across all models and scoring methods.

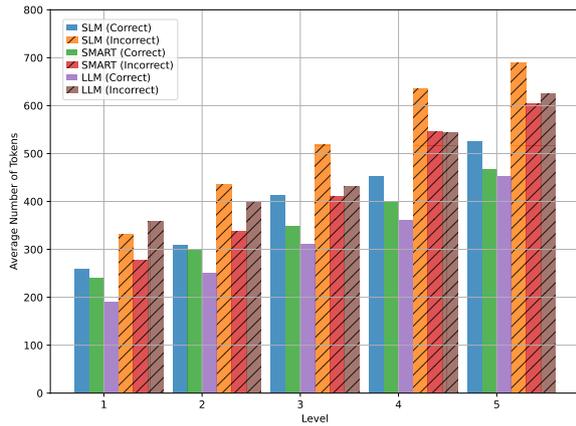


Figure 10: Average number of tokens by model, level, and correctness using the Best-of-N approach with Qwen2.5 models. $N = 32$ for *SMART*, $N = 64$ for *SLM* and *LLM*. PRM scores.

However, the concepts and structure are formulated by the authors, while AI models are employed primarily for refining code and enhancing the clarity and coherence of the writing.

D.4 Average Number of Tokens by Model, Level, and Correctness

Figure 10 illustrates the average number of tokens generated by three different models (*SLM*, *SMART*, and *LLM*) across multiple levels, distinguishing between correct and incorrect responses. The x-axis represents different levels of difficulty, while the y-axis indicates the average number of tokens used. Overall, we observe a trend where incorrect responses (represented with hatched bars) tend to have a higher token count across all models and levels. This suggests that models generally generate longer responses when their predictions are incorrect, possibly due to increased uncertainty or excessive generation in ambiguous scenarios.

D.5 Qualitative analysis of *SMART*

Table 7 illustrates where and how the *LLM* corrects the reasoning steps of the *SLM* when applying our *SMART* method. The table presents randomly selected samples to demonstrate this correction process.

As previously mentioned, the *LLM* consistently provides corrections within the initial reasoning steps (i.e., ## Step 1 or 2). Specifically, it identifies logically flawed components in the *SLM*'s reasoning, generates an alternative formulation, and subsequently adjusts the trajectory to ensure that the following reasoning steps align correctly.

E AI usage

We utilize GPT-4o, O3-mini-high, Copilot, and Perplexity to assist with coding and writing tasks.

Table 7: Example of generation using *SMART* on **Level 1** difficulty problem.

Question: Suppose $\sin D = 0.7$ in the diagram below. What is DE ? [asy]\nnpair D,E,F;\nF = (0,0);\nD = (sqrt(51),7);\nE = (0,7);\ndraw(D--E--F--D);\ndraw(rightanglemark(D,E,F,15));\nlabel("\$D\$",D,NE);\nlabel("\$E\$",E,NW);\nlabel("\$F\$",F,SW);\nlabel("\$7\$",(E+F)/2,W);\n[/asy]

STAR Output: We have two right-angled triangles in the diagram. Let's denote the length of DE as x and the length of DF as h .

(SLM step) From the given $\sin D = 0.7$, we have:\n\\[\frac{h}{7} = 0.7 \]\n\\[h = 7 \times 0.7 = 4.9 \]

(LLM scaffolding step) ## Step 1: Use the right triangle DFE \nIn $\triangle DFE$, we have $\sin D = \frac{\text{opposite}}{\text{hypotenuse}} = \frac{7}{h} = 0.7$. So, $h = \frac{7}{0.7} = 10$.

Step 2: Use the Pythagorean Theorem in triangle DFE \nWe also know that in $\triangle DFE$, $DE^2 + DF^2 = DF^2$. Substituting the known values, we get $x^2 + 7^2 = 10^2$. Simplifying, we get $x^2 + 49 = 100$, so $x^2 = 51$. \n\nTherefore, $x = \sqrt{51}$. \n\nTherefore, the final answer is: $\boxed{\sqrt{51}}$.

Table 8: Example of generation using *SMART* on **Level 2** difficulty problem.

Question: The Greek army contained two types of soldiers: the upper class and the lower class soldiers. If there were a total of 5 upper class soldiers, and 10 lower class soldiers in a certain part of Athens, and the battle of Thermopylae demands a force of 4 upper class soldiers and 8 lower class soldiers, how many different battalions can be sent?

STAR Output: To solve this problem, let's break it down step by step:\n\n

1. First, we need to determine the total number of soldiers in the given part of Athens. We know that there are 5 upper class soldiers and 10 lower class soldiers. So the total number of soldiers is:\n
$$5 + 10 = 15$$
\n\n

(SLM step) 2. Next, to find out how many different battalions can be sent, we need to divide the total number of soldiers by the number of soldiers required per battalion. In this case, each battalion needs 4 upper class soldiers and 8 lower class soldiers. So we divide the total number of soldiers by the sum of these requirements:\n
$$15 \div (4 + 8) = 15 \div 12 = 1.25$$
\n

(LLM correction step) 2. Next, we need to determine how many different battalions can be formed from the upper class soldiers. We need to choose 4 upper class soldiers out of 5. This is a combination problem, and the number of ways to choose 4 soldiers out of 5 is given by the combination formula $\binom{5}{4} = \frac{5!}{4!(5-4)!} = \frac{5!}{4!1!} = \frac{5 \cdot 4!}{4! \cdot 1} = 5$ \n

3. Similarly, we need to determine how many different battalions can be formed from the lower class soldiers. We need to choose 8 lower class soldiers out of 10. This is also a combination problem, and the number of ways to choose 8 soldiers out of 10 is given by the combination formula $\binom{10}{8} = \frac{10!}{8!(10-8)!} = \frac{10!}{8!2!} = \frac{10 \cdot 9 \cdot 8!}{8! \cdot 2 \cdot 1} = \frac{10 \cdot 9}{2} = 45$ \n\n4. Since the battalions are formed separately (one from upper class soldiers and one from lower class soldiers), the total number of different battalions that can be sent is the product of the number of ways to form the battalions from each group:\n
$$5 \cdot 45 = 225$$
\n\nTherefore, the final answer is: $\boxed{225}$ \n\n

Table 9: Example of generation using *SMART* on **Level 3** difficulty problem.

Question: Evaluate $(1+2i)^6-3i$.

STAR Output:

(SLM step) ## Step 1:*\nFirst, we need to distribute the 6 in the expression $(1+2i)^6-3i$. This means multiplying 6 by each term inside the parentheses.

(LLM correction step) ## Step 1: Distribute and simplify the expression\nFirst, distribute 6 across the terms in the parentheses and then subtract $3i$.

Step 2: Perform the multiplication and subtraction\n $(1 + 2i)^6 - 3i = 6 + 12i - 3i = 6 + 9i$.\n\nTherefore, the final answer is: $\boxed{6 + 9i}$.

Table 10: Example of generation using *SMART* on **Level 4** difficulty problem.

Question: What is the probability that a randomly selected integer in the set $\{1, 2, 3, \dots, 100\}$ is divisible by 2 and not divisible by 3? Express your answer as a common fraction.

STAR Output: To find the probability, we need to consider the following steps:

(SLM step) 1. Find the total number of integers in the set.
2. Find the number of integers divisible by 2 and not divisible by 3.
3. Find the number of integers divisible by 3 and not by 2.
4. Subtract the result from step 2 and the result from step 3 from the total to get the number of integers divisible by 2 and not divisible by 3.
5. Divide the result from step 4 by the total number of integers in the set to get the probability.

(LLM correction step) ## Step 1: Determine the total number of integers in the set.
The set is $\{1, 2, 3, \dots, 100\}$, so the total number of integers is 100 .

(SLM step) ## Step 2: Count the number of integers in the set that are divisible by 2.
Integers divisible by 2 in the set are $\{2, 4, 6, \dots, 100\}$. This is an arithmetic sequence where the first term $(a_1 = 2)$ and the common difference $(d = 2)$. The n th term of an arithmetic sequence can be found using the formula $(a_n = a_1 + (n-1)d)$. Setting $(a_n = 100)$ gives the equation $(100 = 2 + (n-1)2)$. Solving for (n) gives $(n = 50)$. Therefore, there are 50 integers in the set that are divisible by 2.

(LLM correction step) ## Step 2: Count the numbers divisible by 2.
A number is divisible by 2 if it is even. The even numbers in the set are $\{2, 4, 6, \dots, 100\}$. This is an arithmetic sequence where the first term is 2 and the last term is 100, with a common difference of 2. The number of terms in this sequence is $(\frac{100}{2} = 50)$.

Step 3: Count the numbers divisible by 2 but not by 3.
We need to exclude the numbers that are divisible by both 2 and 3 (i.e., divisible by 6) from the count of numbers divisible by 2, because they are already counted in the previous step. The numbers divisible by 6 in the set are $\{6, 12, 18, \dots, 96\}$. This is an arithmetic sequence where the first term is 6 and the last term is 96, with a common difference of 6. The number of terms in this sequence is $(\frac{96}{6} = 16)$. Therefore, the number of integers in the set that are divisible by 2 but not by 3 is $(50 - 16 = 34)$.

Step 4: Calculate the probability.
The probability that a randomly selected integer from the set is divisible by 2 but not by 3 is the number of such integers divided by the total number of integers in the set. This gives us $(\frac{34}{100} = \frac{17}{50})$. Therefore, the final answer is: $\boxed{\frac{17}{50}}$.

Table 11: Example of generation using *SMART* on **Level 5** difficulty problem.

Question: Bill walks $\frac{1}{2}$ mile south, then $\frac{3}{4}$ mile east, and finally $\frac{1}{2}$ mile south. How many miles is he, in a direct line, from his starting point? Express your answer as a decimal to the nearest hundredth.

STAR Output: To solve this problem, let's break down Bill's journey into a series of steps and then calculate the final distance from his starting point.

(SLM step) 1. **First Leg**: Bill walks $\frac{1}{2}$ mile south. His position is now $\frac{1}{2}$ mile South from his starting point.
Second Leg: Bill then walks $\frac{3}{4}$ mile east. After this leg, he is now $\frac{3}{4}$ mile North and $\frac{1}{2}$ mile South of his starting point. His net position change in the North-South direction is $-\frac{1}{2} + \frac{3}{4} = \frac{1}{4}$ mile North.
Third Leg: Finally, Bill walks $\frac{1}{2}$ mile south. After this leg, he is now $\frac{1}{4}$ mile North and $\frac{1}{2}$ mile South of his starting point. His net position change in the South direction is $\frac{1}{2} - \frac{1}{4} = \frac{1}{4}$ mile.

(LLM correction step) ## Step 1: Determine the total southward displacement
 Bill walks $\frac{1}{2}$ mile south, then another $\frac{1}{2}$ mile south. So, the total southward displacement is:
 $\frac{1}{2} + \frac{1}{2} = 1$ mile

Step 2: Determine the eastward displacement
 Bill walks $\frac{3}{4}$ mile east. This represents the eastward displacement.

(SLM step) ## Step 3: Calculate the resultant displacement
 To find the resultant displacement, we can represent Bill's movements as vectors. We have:
 - A southward displacement of $\frac{1}{2}$ mile
 - A southward displacement of $\frac{1}{2}$ mile (next south)
 - An eastward displacement of $\frac{3}{4}$ mile

(LLM correction step) ## Step 3: Use the Pythagorean theorem to find the direct distance
 Now we can use the Pythagorean theorem to find the direct distance from the starting point. The total southward displacement and the eastward displacement form the legs of a right triangle, with the direct distance being the hypotenuse. The Pythagorean theorem states:
 $c = \sqrt{a^2 + b^2}$
 where $a = 1$ mile (southward displacement) and $b = \frac{3}{4}$ mile (eastward displacement). Plugging in these values, we get:
 $c = \sqrt{1^2 + \left(\frac{3}{4}\right)^2} = \sqrt{1 + \frac{9}{16}} = \sqrt{\frac{16}{16} + \frac{9}{16}} = \sqrt{\frac{25}{16}} = \frac{5}{4} = 1.25$

Therefore, the final answer is: $\boxed{1.25}$ miles.