SIFT: Grounding LLM Reasoning in Contexts via Stickers

Zihao Zeng, Xuyao Huang[*]**, Boxiu Li**[*]**, Zhijie Deng**[†]
Shanghai Jiao Tong University
{zengzihao, huangxuyao, lbxhaixing154, zhijied}@sjtu.edu.cn

Abstract

This paper identifies the misinterpretation of the context can be a significant issue during the reasoning process of large language models, spanning from smaller models like Llama3.2-3B-Instruct to cutting-edge ones like DeepSeek-R1. For example, in the phrase "10 dollars per kilo," LLMs might not recognize that "per" means "for each," leading to calculation errors. We introduce a novel, post-training approach called **Stick to the Facts (SIFT)** to tackle this. SIFT leverages increasing inference-time compute to ground LLM reasoning in contexts. At the core of SIFT lies the *Sticker*, which is generated by the model itself to explicitly emphasize the key information within the context. Given the curated Sticker, SIFT generates two predictions—one from the original query and one from the query augmented with the Sticker. If they differ, the Sticker is sequentially refined via *forward* optimization (to better align the extracted facts with the query) and *inverse* generation (to conform with the model's inherent tendencies) for more faithful reasoning outcomes. Studies across diverse models (from 3B to 100B+) and benchmarks (e.g., GSM8K, MATH-500) reveal consistent performance improvements. Notably, SIFT improves the pass@1 accuracy of DeepSeek-R1 on AIME2024 from 78.33% to **85.67**% and that on AIME2025 from 69.8% to **77.33**%, establishing a new state-of-the-art in the open-source community. The code is available at `https://github.com/zhijie-group/SIFT`.

1 Introduction

Recent advancements in large language models (LLMs) (Dubey et al., 2024; Yang et al., 2024; Liu et al., 2024) have significantly advanced the field of natural language processing. Techniques including Chain-of-Thought (CoT) Prompting (Wei et al., 2022b; Kojima et al., 2022)

[*]Equal contribution.
[†]Corresponding author.

Query

Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?

Sticker

Conditions:
1. Josh buys a house for $80,000.
2. He spends $50,000 on repairs.
3. The value of the house increases by 150%.

Question:
What is the total profit Josh made from flipping the house?

Figure 1: An example of a query and its Sticker.

and Self-Consistency (Wang et al., 2023b), as well as reasoning-enhanced models, e.g., OpenAI-o1 (Jaech et al., 2024), DeepSeek-R1 (Guo et al., 2025), and KIMI-k1.5 (Team et al., 2025), have all contributed to improvements in multi-step reasoning for solving hard problems.

Recent discussions in the community suggest that advanced reasoning capabilities in LLMs mainly stem from two factors: (i) foundational knowledge acquisition through massive pretraining on diverse data (Dubey et al., 2024; Lin et al., 2025), and (ii) strategic refinement via post-training interventions like supervised fine-tuning (SFT) (Chung et al., 2022) or reinforcement learning (RL) (Guo et al., 2025), which optimize the model's ability to select contextually relevant reasoning pathways. However, our studies reveal a critical lacuna in this framework: LLMs of varying sizes systematically misinterpret, overlook, or hallucinate key information in the query context—an emergent vulnerability we term *factual drift*. For example, Llama3.2-3B-Instruct (Dubey et al., 2024) might incorrectly interpret "per" as "total" instead of "for each" in the phrase "10 dollars per kilo," leading to reasoning errors even with the logical steps being correct. As a result, while current research prioritizes *optimizing reasoning mechanisms* in LLMs (Zelikman et al., 2022, 2024; Wu

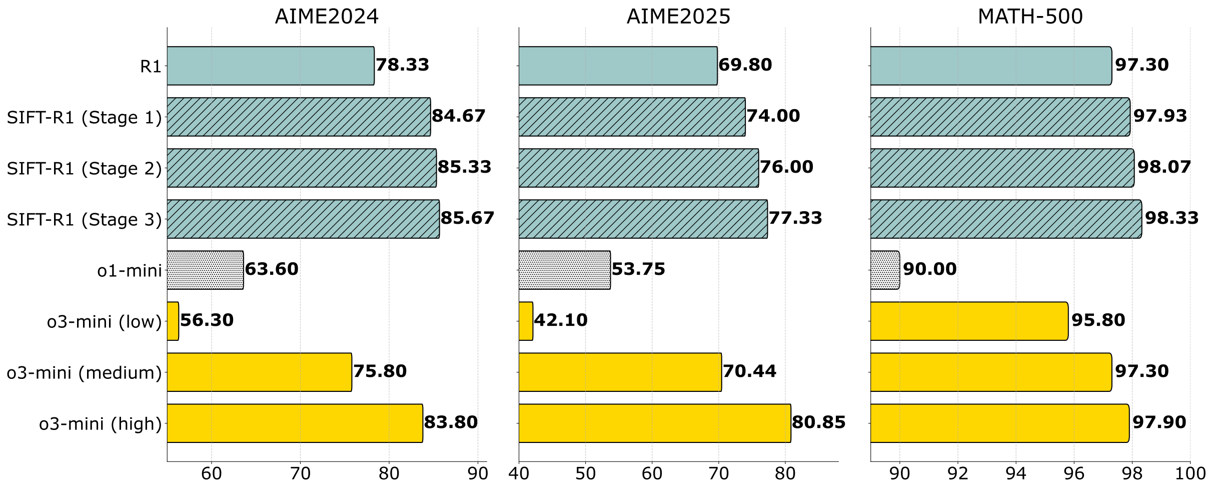


Figure 2: Applying SIFT to DeepSeek-R1 demonstrates highly competitive reasoning performance on AIME2024, AIME2025, and MATH-500 (pass@1 accuracy). The results for o1-mini and o3-mini on AIME are referenced from Ye et al. (2025).

et al., 2024; Zhang et al., 2024b), we argue equal attention should also be placed on *whether LLMs are reasoning about the correct problem.*

We note that advanced reasoning models, such as DeepSeek-R1 (Guo et al., 2025), can partially mitigate factual drift during its reasoning process via *self-verification*. For example, the model dynamically paraphrases critical constraints (e.g., converting "at least 3 days" to "minimum duration $\geq$72 hours") to implicitly perform error-checking. This helps correct prior misunderstandings of the context and leads to better-aligned reasoning results. However, such self-verification operates as a stochastic safeguard rather than a systematic protocol—it is not guaranteed to be triggered in various reasoning scenarios. Namely, the risk of *factual drift* remains and it can be significant considering the results in Figure 2.

Inspired by that humans usually use sticky notes to externalize critical elements when handling complex tasks, we propose the **Stick to the Facts (SIFT)** method to explicitly ground LLM reasoning in contexts using Stickers generated by the model itself. SIFT is a post-training approach, leveraging inference-time compute to improve generation quality yet without reliance on reward models as in Best-of-N (BoN) (Brown et al., 2024; Snell et al., 2024) and Monte-Carlo tree search (MCTS) (Qi et al., 2024; Zhang et al., 2025). Concretely, SIFT lets the target LLM summarize key facts within the input query, including essential *conditions* and the core *question*, into a structured *Sticker* (see Figure 1), and make two predictions based on the Sticker alone and the query augmented with the Sticker, respectively. If they differ, the Sticker is refined through bidirectional optimization—a *forward* one to better align the Sticker with the query and an *inverse* one to conform to the model's reasoning preference—for more faithful reasoning.

Experiments demonstrate that SIFT can consistently improve the reasoning performance across various LLMs and benchmarks. Notably, for DeepSeek-R1 (Guo et al., 2025), SIFT achieves a 1.03% accuracy improvement over the vanilla CoT (97.3%) on MATH-500 (Lightman et al., 2023). Additionally, on AIME2024 (of America, 2024) and AIME2025 challenges, it brings a significant accuracy improvement of 7.34% and 7.54% respectively (see Figure 2), establishing a new state-of-the-art in the open-source community. We also witness a striking performance improvement for small-to-medium-sized models including Llama3.2-3B-Instruct (Dubey et al., 2024), Llama3.1-8B-Instruct (Dubey et al., 2024), and Qwen2.5-7B-Instruct (Yang et al., 2024).

2 Related Work

Reasoning has long been a significant challenge for LLMs. Several approaches aim to improve the reasoning capabilities of LLMs. These methods can be broadly categorized into techniques that align reasoning through training, enhance reasoning through search and planning, or augment reasoning during inference.

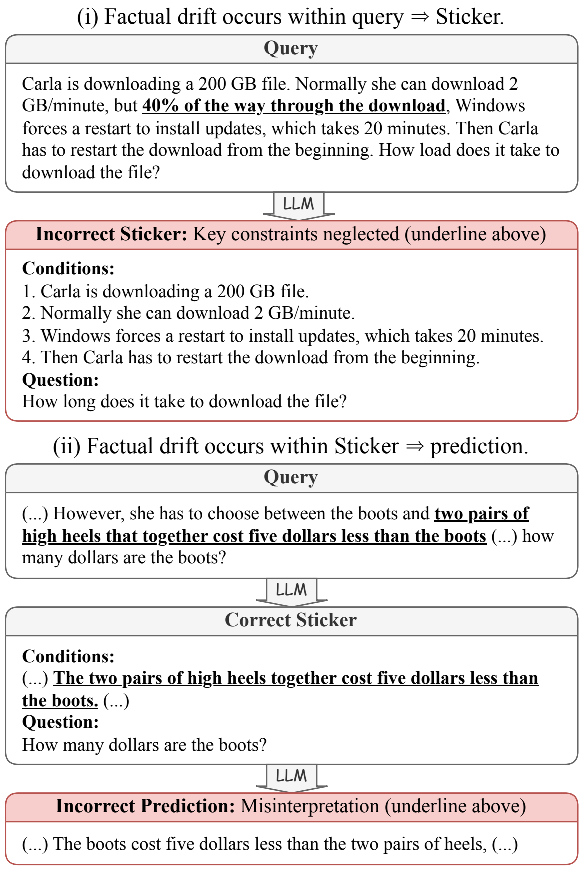
Some approaches focus on aligning the reasoning path of LLMs through Supervised Fine-Tuning

(SFT) or Reinforcement Learning (RL). STaR (Zelikman et al., 2022) enables the model to use reject sampling and learn from its mistakes by rationalizing its outputs, progressively enhancing its reasoning capabilities. Quiet-STaR (Zelikman et al., 2024) generates multiple rationales in parallel before each output token, thereby improving the model's ability to predict subsequent tokens. V-STaR (Hosseini et al., 2024) employs a dual-system framework where the generator creates preference pairs to train the verifier, which then scores the candidate solutions.

Additionally, a significant body of work aims to enhance model reasoning abilities through search and planning. Q* (Wang et al., 2024) formalizes multi-step reasoning as a Markov Decision Process (MDP) and uses the A* algorithm to guide the model in selecting the optimal next step. rStar (Qi et al., 2024) employs Monte Carlo Tree Search (MCTS) to enhance the model's reasoning exploration and uses Mutual Verification to evaluate the reasoning paths. SR-MCTS (Zhang et al., 2024a) combines Self-Refinement and MCTS to iteratively improve and optimize newly discovered reasoning paths. MCTS-DPO (Xie et al., 2024) leverages MCTS to collect step-level preference data and uses Decision-Policy Optimization (DPO) to refine the model's policy through multiple iterations. ReST-MCTS* (Zhang et al., 2025) takes a broader approach in evaluating reasoning paths, considering not only the correctness of the results but also the quality of the reasoning process, such as the shortest path and error-free intermediate steps. CoRe (Zhu et al., 2022) constructs a dual-system approach with System 1 for generation and System 2 for verification, training, and reasoning simultaneously to simulate human-like reasoning processes. AlphaMath (Chen et al., 2024) treats the output of the LLM as an action and integrates a value model and a policy model, iteratively training the model to enhance its reasoning capabilities.

There are also methods that focus on enhancing reasoning abilities during inference. Innovations in prompt engineering have contributed to advancements in reasoning capabilities. Chain-of-Thought (CoT) prompting (Wei et al., 2022a; Kojima et al., 2022) guides models in stepwise reasoning, such as by manually annotating natural language rationales or appending "Let's think step by step" after questions. Auto-CoT (Zhang et al., 2022) clusters questions and uses zero-shot Chain-of-Thought to generate reasoning chains, which are then used as prompts to guide the model's answers. ToT (Yao et al., 2023) removes the constraints of chain structures by incorporating tree structures and search algorithms, allowing models to explore widely during reasoning. The seminal Self-Consistency method (Wang et al., 2023a) aggregates answers through majority voting over multiple reasoning paths, while Madaan et al. (2024) introduces iterative self-correction via feedback loops.

However, these methods primarily focus on refining *how* models reason rather than ensuring that they address the *correct problem*. Our approach differs by prioritizing factual comprehension before answer generation, ensuring proper problem understanding.


(i) Factual drift occurs within query ⇒ Sticker.

Query

Carla is downloading a 200 GB file. Normally she can download 2 GB/minute, but **40% of the way through the download**, Windows forces a restart to install updates, which takes 20 minutes. Then Carla has to restart the download from the beginning. How load does it take to download the file?

LLM

Incorrect Sticker: Key constraints neglected (underline above)

Conditions:
1. Carla is downloading a 200 GB file.
2. Normally she can download 2 GB/minute.
3. Windows forces a restart to install updates, which takes 20 minutes.
4. Then Carla has to restart the download from the beginning.

Question:
How long does it take to download the file?

(ii) Factual drift occurs within Sticker ⇒ prediction.

Query

(...) However, she has to choose between the boots and **two pairs of high heels that together cost five dollars less than the boots** (...) how many dollars are the boots?

LLM

Correct Sticker

Conditions:
(...) **The two pairs of high heels together cost five dollars less than the boots.** (...)

Question:
How many dollars are the boots?

LLM

Incorrect Prediction: Misinterpretation (underline above)

(...) The boots cost five dollars less than the two pairs of heels, (...)

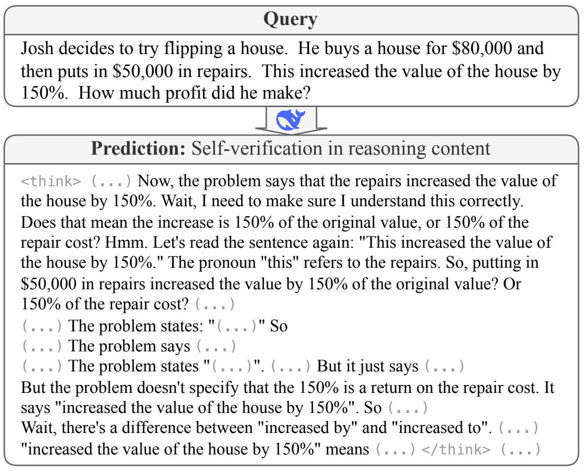
Figure 3: Factual drift occurs during (i) Sticker generation and (ii) prediction generation from Sticker.

3 Method

This section first presents the factual drift issue during LLM reasoning and then elaborates on the proposed Stick to the Facts (SIFT) approach.

3.1 Factual Drift in LLM Reasoning

We define *factual drift* as the phenomenon where the LLM reasoning fails due to misaligned com-



Query

Josh decides to try flipping a house. He buys a house for \$80,000 and then puts in \$50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?

Prediction: Self-verification in reasoning content

`<think>` `(...)` Now, the problem says that the repairs increased the value of the house by 150%. Wait, I need to make sure I understand this correctly. Does that mean the increase is 150% of the original value, or 150% of the repair cost? Hmm. Let's read the sentence again: "This increased the value of the house by 150%." The pronoun "this" refers to the repairs. So, putting in \$50,000 in repairs increased the value by 150% of the original value? Or 150% of the repair cost? `(...)`
`(...)` The problem states: "`(...)`" So
`(...)` The problem says `(...)`
`(...)` The problem states "`(...)`". `(...)` But it just says `(...)`
But the problem doesn't specify that the 150% is a return on the repair cost. It says "increased the value of the house by 150%". So `(...)`
Wait, there's a difference between "increased by" and "increased to". `(...)`
"increased the value of the house by 150%" means `(...)` `</think>` `(...)`

Figure 4: Self-verification occurs during DeepSeek-R1's reasoning, where the model revisiting the query, focusing on key information, and paraphrasing it.

Algorithm 1: LLM reasoning with SIFT

```
Input  : Query $Q$
Output : Final result of $Q$
$S_1 \leftarrow \mathrm{SG}(Q)$ ;                    // Sticker generation
$P_1 \leftarrow \mathrm{CP}(Q, S_1)$;
if $P_1 \neq \rightsquigarrow$ then
    return $P_1$ ;                                   // Exit if consensus
else
    // Forward
    $S_2 \leftarrow \mathrm{FO}(Q, S_1)$, $P_2 \leftarrow \mathrm{CP}(Q, S_2)$;
    if $P_2 \neq \rightsquigarrow$ then
        return $P_2$
    else
        // Inverse
        $S_3 \leftarrow \mathrm{FO}(Q, \mathrm{IG}(P_{Q,S_2}))$;
        $P_3 \leftarrow \mathrm{CP}(Q, S_3)$;
        return $P_3$ if $P_3 \neq \rightsquigarrow$ else $\mathrm{LLM}(Q)$
    end
end
```

Algorithm 2: Consensus Prediction (CP)

```
Input  : Query $Q$, Sticker $S$
Output : Prediction from $Q$ & $S$, or $\rightsquigarrow$ (unequal)
$P_S \leftarrow \mathrm{LLM}(S)$ ;                       // Sticker-only
$P_{Q,S} \leftarrow \mathrm{LLM}(Q, S)$ ;                // Query+Sticker
if $\mathrm{EQUIVALENT}(P_S, P_{Q,S})$ then
    // Consensus validation
    return $P_{Q,S}$
else
    return $\rightsquigarrow$
end
```

prehension of the query context rather than flawed reasoning logic. This occurs when LLMs neglect key constraints, misinterpret semantic relationships, or hallucinate non-existent conditions during reasoning procedures.

We show that factual drift can be a systematic failure mode of general LLM problem-solving processes beyond reasoning. Taking the task of applying Stickers to Llama3.2-3B-Instruct (Dubey et al., 2024) on GSM8K test set (Cobbe et al., 2021) as an example, we curate Stickers with the model, based on which predictions are made. We observe extensive factual drift errors, with typical examples displayed in Figure 3. As shown, when mapping the query to Stickers, LLMs may neglect the original constraints. Moreover, even when the Sticker is correct, LLMs may still misunderstand it, especially when the question is complex or uses less familiar phrasing. The above observations also highlight that *more optimization mechanisms regarding the Sticker are required to make it (i) more aligned with the query and (ii) able to be easily understood and leveraged by the target LLM*.

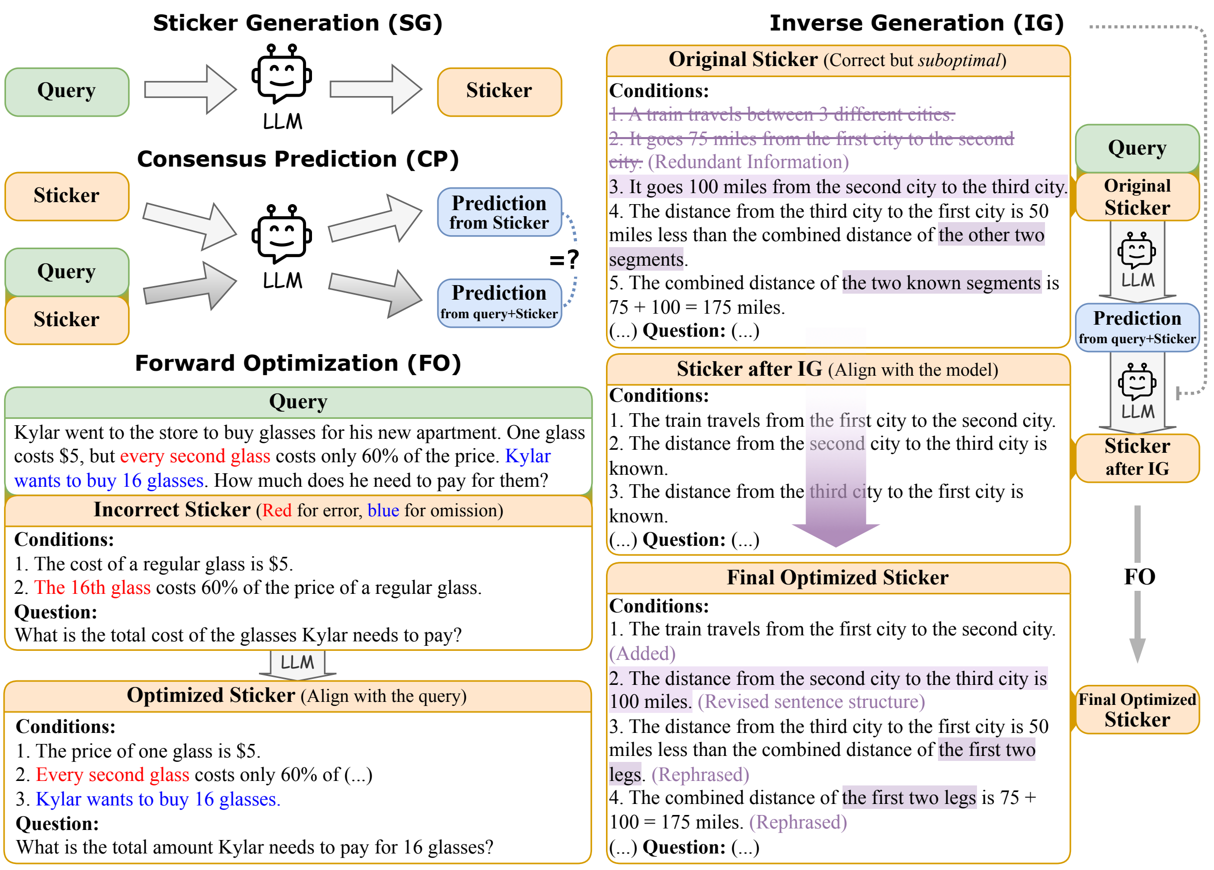
Self-verification of Advanced Reasoning Models. We note that, for advanced models like DeepSeek-R1 (Guo et al., 2025), the reasoning process sometimes involves *self-verification*—revisiting the original problem, focusing on key information, and paraphrasing it. As illustrated in Figure 4, DeepSeek-R1 often states, "Let's read the sentence again: ..." or "Wait, the problem states: ..." as part of its thought process, helping to deepen its understanding of the context or self-correct.

The excellent performance of such advanced reasoning models underscores the efficacy of mitigating factual drift to make the model better respect the context. Nevertheless, this self-verification functions more as a stochastic safeguard than a systematic protocol—it may not always be activated across different reasoning scenarios. Consequently, the risk of factual drift persists. We consequently develop the novel SIFT framework to address this.

3.2 Stick to the Facts (SIFT)

SIFT includes four core operations (see Figure 5): (i) Sticker Generation (SG), which extracts the Sticker from the original query; (ii) Consensus Prediction (CP), which validates the alignment between predictions from the Sticker and the query augmented with the Sticker; (iii) Forward Optimization (FO), which refines the Sticker to improve its alignment with the facts in the query; (iv) Inverse Generation (IG), which generates the Sticker based on the prediction inversely.

The full procedure of SIFT is shown in Algorithm 1 with the details of Consensus Prediction in Algorithm 2. All prompts used can be found in Appendix B. We explain some rationales below.



Query

Kylar went to the store to buy glasses for his new apartment. One glass costs $5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?

Incorrect Sticker (Red for error, blue for omission)

Conditions:
1. The cost of a regular glass is $5.
2. The 16th glass costs 60% of the price of a regular glass.

Question:
What is the total cost of the glasses Kylar needs to pay?

Optimized Sticker (Align with the query)

Conditions:
1. The price of one glass is $5.
2. Every second glass costs only 60% of (...)
3. Kylar wants to buy 16 glasses.

Question:
What is the total amount Kylar needs to pay for 16 glasses?

Original Sticker (Correct but *suboptimal*)

Conditions:
~~1. A train travels between 3 different cities.~~
~~2. It goes 75 miles from the first city to the second city.~~ (Redundant Information)
3. It goes 100 miles from the second city to the third city.
4. The distance from the third city to the first city is 50 miles less than the combined distance of the other two segments.
5. The combined distance of the two known segments is 75 + 100 = 175 miles.
(...) **Question:** (...)

Sticker after IG (Align with the model)

Conditions:
1. The train travels from the first city to the second city.
2. The distance from the second city to the third city is known.
3. The distance from the third city to the first city is known.
(...) **Question:** (...)

Final Optimized Sticker

Conditions:
1. The train travels from the first city to the second city. (Added)
2. The distance from the second city to the third city is 100 miles. (Revised sentence structure)
3. The distance from the third city to the first city is 50 miles less than the combined distance of the first two legs. (Rephrased)
4. The combined distance of the first two legs is 75 + 100 = 175 miles. (Rephrased)
(...) **Question:** (...)

Figure 5: Four core operations in SIFT: (i) Sticker Generation (SG), (ii) Consensus Prediction (CP), (iii) Forward Optimization (FO), (iv) Inverse Generation (IG).

Consensus Prediction: Beyond Answer Aggregation. Traditional self-consistency methods sample diverse reasoning paths to aggregate answers (Wang et al., 2023a), focusing on *how* models reason. In contrast, our Consensus Prediction verifies *whether models reason about the same problem* with dual representations: (i) the *Sticker-Only* one, which forces the model to solve the problem using only the key conditions and the core question, and (ii) the *Query+Sticker* one, which provides richer contexts. This way, the model explores *semantic invariance* rather than sampling diversity when reasoning about the answers.

CP does not require sampling and operates with greedy decoding by default. However, it remains compatible with stochastic sampling, as demonstrated in Table 1. Besides, the CP operates only based on the current Sticker, preventing contamination from historical reasoning traces. As illustrated in Algorithm 2, consensus between representations acts as a factual invariant—a necessary (though not sufficient) condition for correctness. This design intentionally avoids conflating factual grounding with reasoning quality assessment.

Forward Optimization: Anchoring Stickers to Source Semantics. As discussed in Section 3.1, the SG process can also inevitably suffer from factual drift, where the original constraints are misrepresented or misunderstood. To address this, we combine the generated Sticker with the query to produce a refined Sticker. For example, it can correct misinterpretations, such as changing "the 16th glass" to "every second glass" in Figure 5.

Inverse Generation: Aligning Stickers to Model Reasoning Preference. It is frequently observed that for LLM reasoning, contexts with the same semantics but different presentations can yield distinct results. This implies that, after doing FO, it can be beneficial to further refine the Sticker based on the LLM's reasoning process. Given this insight, we use the LLM to inversely infer a new Sticker given the model prediction. We further invoke FO once again to the new Sticker to avoid factual drift. This step makes the Sticker respect the internal reasoning preferences of the model for representing facts, arranging conditions, or structuring questions (see Figure 5). This also helps the model recognize the difference between the Sticker from IG and the original question, to enable the model to capture overlooked information and generate a more comprehensive Sticker.

4 Experiments

In this section, we first validate the effectiveness and generalization of SIFT (Section 4.1). Next, we explore several variants (Section 4.2 & 4.3). Finally, we include ablation studies to gain further insights into our approach (Section 4.4).

4.1 Enhancing LLM Reasoning with SIFT

Models & Datasets. We test SIFT on a diverse set of state-of-the-art LLMs, including Llama3.2-3B-Instruct (Dubey et al., 2024), Llama3.1-8B-Instruct (Dubey et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2024), and DeepSeek-R1 (Guo et al., 2025). These models cover a range of sizes, architectures (Mixture-of-Experts (MoE) vs. dense), and reasoning capabilities. We select well-established reasoning benchmarks, including GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2023), GPQA-Diamond (Rein et al., 2023), and AIME2024 (of America, 2024).

Test Protocol. To isolate the effect of SIFT from the influence of sampling, all tests are conducted using greedy decoding, except for DeepSeek-R1. Because the default settings of the used Volcengine API (temperature=1.0, top-p=0.7) cannot be modified, the SIFT on DeepSeek-R1 is based on sampling. Specifically, for DeepSeek-R1 on MATH-500, we perform 3 sampling runs and report average results. For AIME2024, due to its small size, we perform 10 sampling runs and report the average. Additionally, we divide the entire SIFT process into three stages: (i) Stage 1: Only SG and CP are used. (ii) Stage 2: Building upon Stage 1, FO is used to optimize the Sticker. (iii) Stage 3: The complete process outlined in Algorithm 1. The accuracy after each stage is measured: If the CP results are not aligned ($\rightsquigarrow$), the model's direct answer to the query is used instead. All evaluations are performed on OpenCompass (Contributors, 2023).

Main Results. The results are shown in Figures 2, 6 and 11. As observed, SIFT consistently delivers robust and significant performance improvements compared to traditional Zero-shot CoT across all settings. From a methodological perspective, as the stages increase—i.e., with the forward and inverse optimization of Sticker—the average number of tokens used per sample rises, and accuracy shows an upward trend as well. From a model standpoint, SIFT demonstrates notable effectiveness across various scales (ranging from several billion to hundreds of billions of parameters), architectures (both dense and MoE), and paradigms (traditional and reasoning models). Particularly noteworthy is its significant impact on DeepSeek-R1. For instance, on MATH-500, it achieves a 1.03% absolute accuracy improvement over an already exceptionally high baseline of 97.3%. On AIME2024, it also brings a substantial absolute accuracy increase of 7.34%. These results indicate that even for advanced reasoning models like DeepSeek-R1, sticking to the facts remains crucial for optimal performance.

4.2 Iterative Optimization

In this section, we explore whether the Sticker can be continually optimized in SIFT.

Setup. We test with Llama3.2-3B-Instruct (Dubey et al., 2024) on the GSM8K dataset (Cobbe et al., 2021). Specifically, we conduct multiple optimization repeats for Stage 2 and Stage 3. The other settings are the same as in Section 4.1.

Results. The experimental results are shown in Figure 7. We observe that SIFT shows a test-time scaling, with the performance improving as the average number of tokens per sample increases. For Stage 2, the saturation is rapid, but adding Stage 3 can result in an additional, noticeable performance boost. Nevertheless, the most significant gains are observed at the first repeat. One possible explanation is that extracting the optimal Sticker for GSM8K is relatively easy. In more complex conditions, however, extracting a good Sticker may be harder, requiring more repeats to achieve optima. Additionally, since we use a training-free approach for SIFT, a model trained to exclusively optimize Sticker could lead to better iterative results.

4.3 Sample Augmentation

In this section, we explore the use of Self-Consistency (SC) (Wang et al., 2023a) to enhance SIFT, demonstrating how SIFT and SC can be effectively coupled together.

Specifically, SIFT and SC can be integrated in three ways: (i) Sticker-Consistency: Multiple Sticker samples are drawn, and consistency is applied to the predictions generated by each Sticker or by the query combined with each Sticker. (ii)

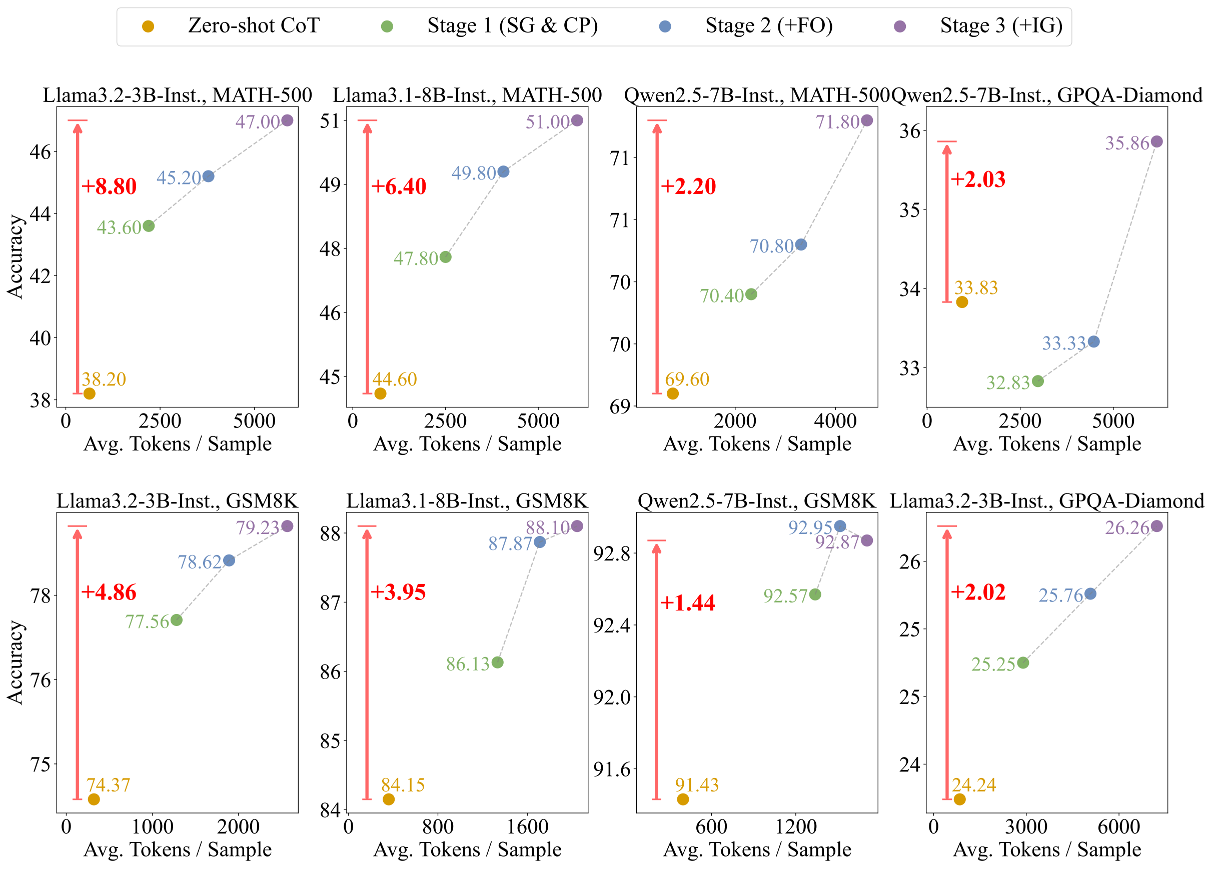


Figure 6: Comparison of SIFT and traditional Zero-shot CoT across multiple models and datasets. We divide SIFT into three stages: Stage 1 only uses SG & CP, while Stage 2 and Stage 3 optimize the Sticker through forward (+FO) and inverse (+IG) direction, respectively. The bidirectional arrows in the figure highlight the performance gap between Zero-shot CoT and the complete SIFT (i.e., Stage 3). We see that in nearly all scenarios, SIFT leads to a significant performance improvement.

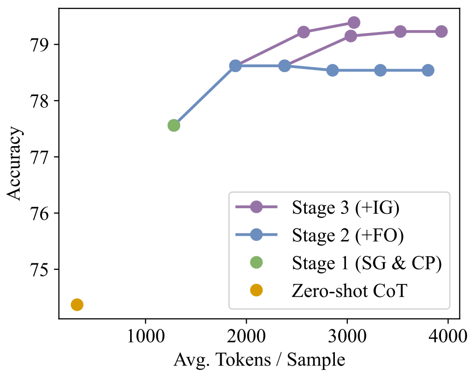


Figure 7: Iterative optimization results for SIFT. The performance improves as the number of tokens per sample increases across different stages. Significant gains are observed in the first repeats of Stage 2 and Stage 3.

Prediction-Consistency: Consistency is applied separately to predictions generated using *Sticker* alone and those generated with *Query* + *Sticker*, considering their respective samples. (iii) SIFT-Consistency: End-to-end sampling is conducted across the entire SIFT to ensure consistency. We test Llama3.2-3B-Instruct (Dubey et al., 2024) on GSM8K (Cobbe et al., 2021) with a temperature of 0.6, a top-p of 0.9, and 10 sampling iterations.

The results of these configurations are presented in Table 1. It is observed that our method can be combined with SC to achieve better performance. Specifically, Integrating SIFT across all dimensions results in performance improvements. Notably, SIFT-Consistency provides the most significant boost, demonstrating that the simplest sampling method—end-to-end—can lead to substantial performance gains for SIFT.

4.4 Ablation

Evolution of Consensus Across Optimization Stages. The efficacy of SIFT hinges on improving agreement between predictions derived from *Sticker-only* and *Query* + *Sticker* representations through iterative refinement. To quantify this alignment, We select Llama3.2-3B-Instruct (Dubey

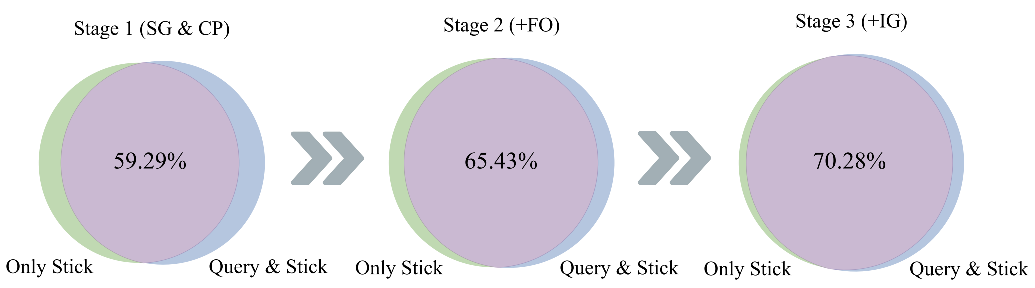


Figure 8: Venn diagrams illustrating the accuracy of predictions obtained from the "Only Sticker" and "Query & Sticker" representations at each stage. The percentages represent the accuracy where both methods correctly predict the same outcomes. From Stage 1 to Stage 2, the accuracy increases by 6.14%, and from Stage 2 to Stage 3, it increases by 4.85%. The results show the significant impact of Forward Optimization (FO) and Inverse Generation (IG) in improving prediction alignment from the two representations.

Consistency Dimension	Stage 1	Stage 2	Stage 3
Greedy	77.56	78.62	79.23
(i) Sticker	78.85	79.65	80.29
(ii) Prediction	85.37	86.20	86.28
(iii) SIFT	—	—	88.25

Table 1: Performance comparison of different consistency integration strategies for SIFT across multiple stages. The results show that integrating SIFT with Self-Consistency (Wang et al., 2023a) leads to significant performance improvements, with SIFT-Consistency achieving the highest accuracy boost.

Model	Stage 1	Stage 2	Stage 3	Stage 3 from Stage 1
Llama	77.56	78.62	79.23	74.07
Qwen	92.57	92.95	92.87	90.90

Table 2: Performance comparison of Llama3.2-3B-Instruct and Qwen2.5-7B-Instruct on GSM8K, with and without Stage 2. The results show a performance drop when skipping directly from Stage 1 to Stage 3.

et al., 2024) on the GSM8K dataset (Cobbe et al., 2021). We plot the accuracy of predictions obtained using "Only Sticker" and "Query & Sticker" after each stage, visualized in the Venn diagram in Figure 8. As shown, both FO and IG significantly improve the alignment of the predictions from the two representations. Specifically, the accuracy where both methods correctly predict the same outcomes increased by 6.14% from Stage 1 to Stage 2, and by an additional 4.85% from Stage 2 to Stage 3.

FO Required Before Adding IG. We investigate

Strategy	Accuracy
$P_{Q,S}$ `if` $P_{Q,S}$=P_S `else` P_Q	77.56
P_S `if` P_S=P_Q `else` $P_{Q,S}$	77.02
P_Q `if` P_Q=$P_{Q,S}$ `else` P_S	76.04

Table 3: Performance comparison of various CP strategies. Here, P_Q, P_S, and $P_{Q,S}$ represent the predictions generated from query, Sticker, and query augmented with Sticker, respectively. The first row of the table represents the strategy used in SIFT, which is shown to be the optimal approach.

whether it is possible to skip directly from Stage 1 to Stage 3. We select Llama3.2-3B-Instruct and Qwen2.5-7B-Instruct on GSM8K. All settings remain the same as in Section 4.1, except for skipping directly to Stage 3 after Stage 1. The results are shown in Table 2. As observed, skipping Stage 2 leads to a significant performance drop. This indicates that during the initial optimization of Sticker, FO is essential to align Sticker with the query, followed by aligning it with model cognition. This is consistent with our experience, where the effectiveness of Sticker depends primarily on its correctness—ensuring no *factual drift*—before considering its alignment with the model.

Comparison of SIFT and Standard Self-Consistency. Under the same sampling conditions (temperature = 0.6, top-p = 0.9), we compare the performance of standard Self-Consistency (SC) with SIFT. The evaluation is conducted using Llama3.2-3B-Instruct on GSM8K. For SIFT, we sample 10 times and take the average. The results are shown in Figure 9. Regarding the total tokens used by both methods, the performance curve of SIFT generally remains above that of SC. Notably,

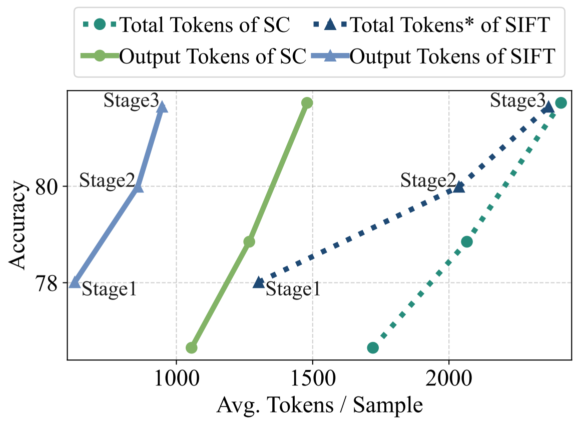


Figure 9: Comparison of SIFT and standard Self-Consistency (SC) in terms of accuracy versus average tokens per sample. The solid lines represent the output tokens used by SC (blue) and SIFT (red), while the dashed lines indicate the total tokens consumed. The "*" symbol in the legend denotes that the total tokens for SIFT fluctuate due to the additional formatting and example constraints used during inference. SIFT achieves comparable accuracy to SC while using significantly fewer output tokens, demonstrating its efficiency.

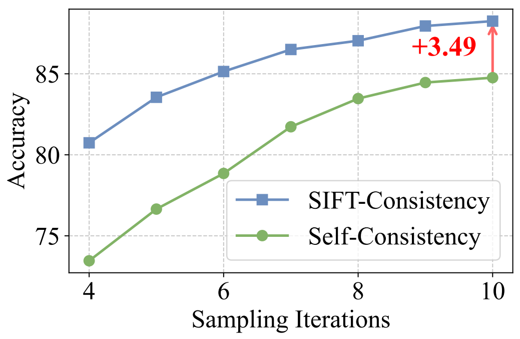


Figure 10: Comparison of SIFT-Consistency and Self-Consistency across different numbers of sampled responses per query. SIFT-Consistency consistently outperforms Self-Consistency.

the "Total Tokens*" in the legend indicates that the total number of tokens used by SIFT varies. This variability arises because when generating the Sticker, we may need to provide the output format and some examples to guide its responses. This additional input contributes significantly to the total token count, which fluctuates considerably. We believe that further optimization can significantly reduce the total tokens required by SIFT. Regarding output tokens, which are more costly during inference, SIFT demonstrates a clear advantage over SC. Specifically, SIFT achieves a comparable performance level while using only two-thirds of the output tokens required by SC, highlighting its efficiency.

Comparison of SIFT-Consistency and Standard Self-Consistency. In the same sampling environment (temperature = 0.6, top-p = 0.9), we compare the performance of standard Self-Consistency (SC) decoding with SIFT-Consistency, which integrates SIFT with SC. We conduct the evaluation using the Llama3.2-3B-Instruct model on the GSM8K dataset. The results are shown in Figure 10. As shown in the figure, SIFT-Consistency consistently outperforms standard SC across different sampling iterations.

Optimal Consensus Prediction Strategy. CP process, our strategy involves comparing predictions from *Sticker* and *query* + *Sticker*. If the predictions are consistent, we adopt the prediction from Query + Sticker; otherwise, we use the prediction directly from *query*. We validate this as the optimal strategy. Several alternative strategies were evaluated using Stage 1 results of Llama3.2-3B-Instruct on the GSM8K dataset, as shown in Table 3. The results demonstrate that our CP strategy is effective, aligning with the prior analysis in Section 3.2.

5 Conclusion

This study presents Stick to the Facts (SIFT), a training-free framework that anchors LLM reasoning to contextual facts through iterative self-refinement. This approach enhances the reliability of LLM reasoning, providing a practical solution for factually grounded reasoning without the need for additional data or training.

Limitations

This work focuses on the training-free setting. In the future, SIFT could be internalized into small LLMs through dedicated training, enabling more efficient on-device reasoning. Separately, SIFT can be applied to reduce the output token length of reasoning models, improving computational efficiency without compromising accuracy. Additionally, Inverse Generation in SIFT offers new inspiration for data generation in inverse synthesis tasks. Further studies are needed to generalize its effectiveness across a wider range of tasks.

References

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

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. *arXiv preprint arXiv:2110.14168*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. 2025. Rho-1: Not all tokens are what you need. *Preprint*, arXiv:2404.07965.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Mathematical Association of America. 2024. American invitational mathematics examination - aime 2024.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

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*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Chaojie Wang, Yanchen Deng, Zhiyi Lyu, Liang Zeng, Jujie He, Shuicheng Yan, and Bo An. 2024. Q*: Improving multi-step reasoning for llms with deliberative planning. *arXiv preprint arXiv:2406.14283*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023a. Self-consistency improves chain of thought reasoning in language models. *The Eleventh International Conference on Learning Representations*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022a. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

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.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *Preprint*, arXiv:2408.00724.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Yixin Ye, Yang Xiao, Tiantian Mi, and Pengfei Liu. 2025. Aime-preview: A rigorous and immediate evaluation framework for advanced mathematical reasoning. `https://github.com/GAIR-NLP/AIME-Preview`. GitHub repository.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2025. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, et al. 2024a. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024b. Generative verifiers: Reward modeling as next-token prediction. *Preprint*, arXiv:2408.15240.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. 2022. Solving math word problems via cooperative reasoning induced language models. *arXiv preprint arXiv:2210.16257*.

A More Results

We demonstrate how SIFT 's performance on DeepSeek-R1 evolves with an increasing average token count (see Figure 11).

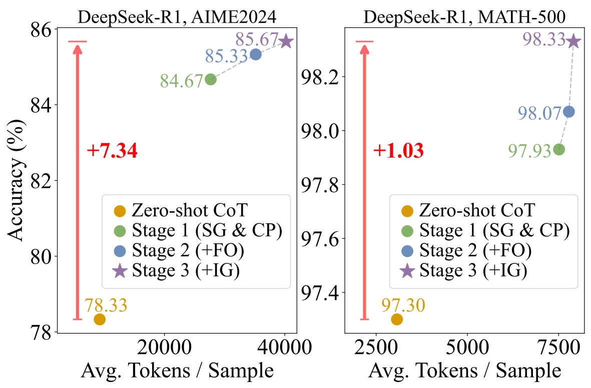


Figure 11: SIFT performance on DeepSeek-R1 with increasing average token count.

Prediction ⇒ Sticker

~~~
Given the prediction provided below, reverse-engineer
the abstract that led to it. The abstract should
include both the conditions and the question.
~~~

~~~
Abstract Format:

`
**Conditions:**
1. [Condition 1]
2. [Condition 2]
...(add more conditions as needed)

**Question:**
[Clearly state what is being asked.]
`
~~~

~~~
Requirements:
1. Conditions:
    - Clearly list all the given information.
    - Write each condition on a separate line,
numbered sequentially.
    - EACH CONDITION MUST BE ATOMIC AND INDIVISIBLE
(i.e., it cannot be divided into two sub-conditions).
    - DO NOT INCLUDE ANY PART OF THE REASONING
PROCESS!!!
2. Question:
    - Summarize what is being asked in one clear
sentence.
    - Remove all known conditions.
~~~

~~~
Example:

Prediction:(...)

Expected Output:(...)
~~~

~~~
Prediction to Process:

`
{prediction}
`
~~~

~~~
Please provide your output strictly following the
ABSTRACT FORMAT without other unnecessary words.
~~~

Figure 12: Prompt format for generating a Sticker inversely from the prediction.

B Prompting for SIFT

In this section, we present the complete prompt formats used in the SIFT process (see Figures 12 to 15 for details).

Query ⇒ Prediction

~~~
{Query}
Please reason step by step, and put your final answer
within \boxed{}.
~~~

Sticker ⇒ Prediction

~~~
{Sticker}

Please reason step by step, and put your final answer
within \boxed{}.
~~~

Query + Sticker ⇒ Prediction

~~~
{Query}
{Sticker}

Please reason step by step, and put your final answer
within \boxed{}.
~~~

Figure 13: Prompt format for generating predictions.

Query ⇒ Sticker

~~~
Extract fundamental elements from the following query
using atomic decomposition methodology.
~~~

~~~
Requirements:
1. Conditions: Clearly list all the given information.
Write each condition on a separate line, numbered
sequentially.
2. Question: Summarize what is being asked in one clear
sentence. Remove all known conditions.
~~~

~~~
Output Format:

`
**Conditions:**
1. [Condition 1]
2. [Condition 2]
...(add more conditions as needed)

**Question:**
[Clearly state what is being asked.]
`
~~~

~~~
Example:

Query:(...)

Expected Output:(...)
~~~

~~~
Query to Process:

`
{question}
`
~~~

~~~
Please provide your output strictly following the output
format without other unnecessary words.
~~~

Figure 14: Prompt format for generating a Sticker from the query.

Query + Sticker ⇒ Sticker

```
Given a query and a candidate abstract (which includes
conditions and a question), output an optimized
abstract.
```

```
Requirements:
1. Definitions of Conditions and Question:
    * Conditions: Clearly list all the given
information. Write each condition on a separate line,
numbered sequentially.
    * Question: Summarize what is being asked in one
clear sentence. Remove all known conditions.
2. Focus of Optimization: Compare the Original Query
with the candidate Abstract. Identify and fix:
    * Missing/incorrect/redundant conditions
    * Imprecise question phrasing
    * Mathematical/logical inconsistencies
    * Output format error
```

````
Output Format:

`
**Conditions:**
1. [optimized Condition 1]
2. [optimized Condition 2]
...(add more conditions as needed)

**Question:**
[Optimized question phrasing. Clearly state what is
being asked.]
`
````

```
Some Examples:(...)
```

````
Input to Process:

`
Original Query:
{question}

Candidate Abstract:
{abstract}
`
````

```
Please provide your output strictly following the output
format without other unnecessary words.
```

Figure 15: Prompt format for forward optimization of the Sticker.