SIFT: Grounding LLM Reasoning in Contexts via Stickers

Anonymous ACL submission

Abstract

This paper identifies that misinterpreting 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. 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 Sticker, SIFT generates two predictions—one from the Sticker alone 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., MATH, AIME) 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**%. Code will be public after acceptance.

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.

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) 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 complex 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 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 the reasoning process via *self-verification*. For example, the model dynamically paraphrases critical constraints (e.g., convert-

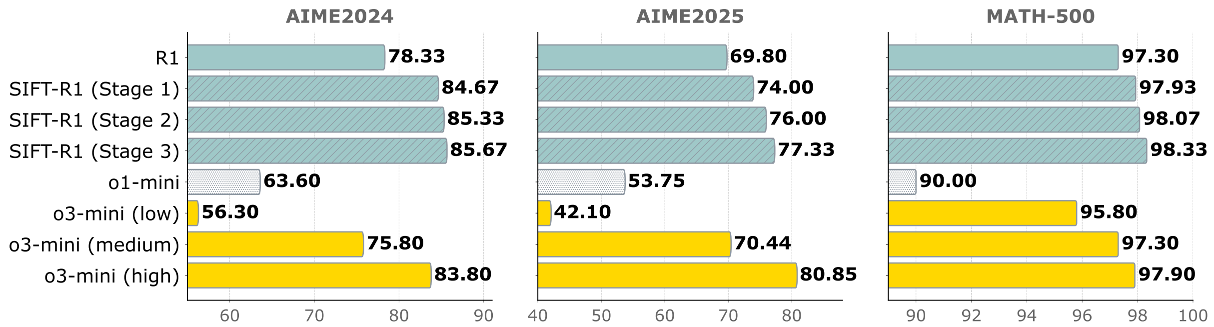

Figure 2: Applying SIFT to DeepSeek-R1 yields highly competitive pass@1 accuracy on AIME 2024, AIME 2025, and MATH-500. Results for the o-series on AIME are referenced from Ye et al. (2025).

ing "at least 3 days" to "minimum duration ≥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 random 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 training-based alignment, search and planning enhancement, and inference-time augmentation.

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 explo-

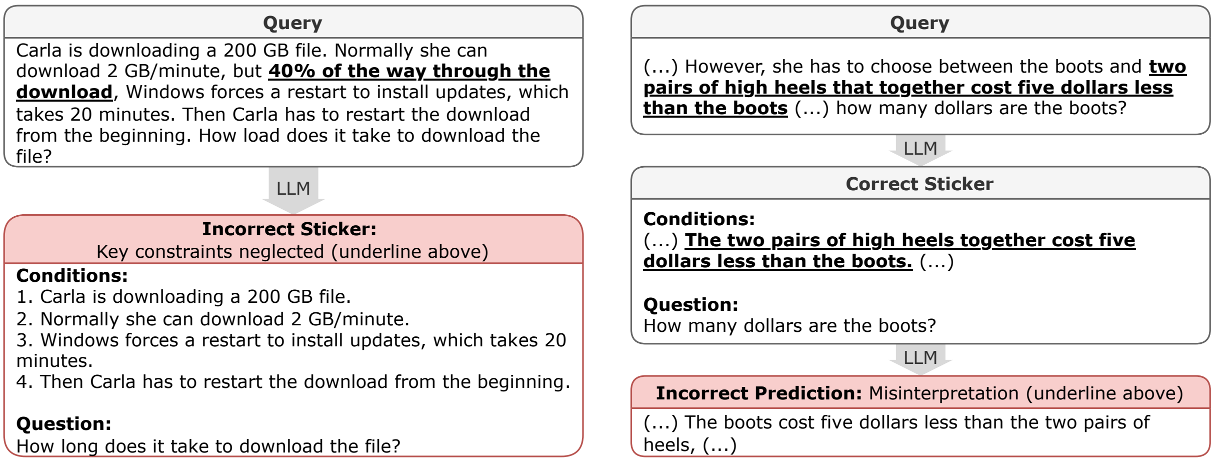

Figure 3: Illustration of factual drift in our investigation on Stickers. **Left**: During query-to-sticker generation. **Right**: During prediction generation from the sticker.

ration 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 focus on refining *how* models reason rather than ensuring they address the *correct problem*. Our approach differs by prioritizing factual comprehension to ensure proper problem understanding before answer generation.

3 Method

We first presents the factual drift issue during LLM reasoning and then elaborates on the proposed Stick to the Facts (SIFT) approach. Find more discussion on the definition of Sticker in Appendix A.

3.1 Factual Drift in LLM Reasoning

We define *factual drift* as the phenomenon where the LLM reasoning fails due to misaligned comprehension 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. Specifically, we analyze the error statistics of both Qwen2.5-7B-Instruct (Yang et al., 2024) and Llama3.2-3B-Instruct (Dubey et al., 2024) on samples from the GSM8K test set (Cobbe et al., 2021). For each model, we distinguish between two primary error types: those resulting from factual drift and those

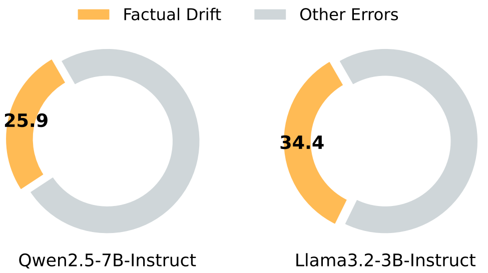

Figure 4: Distribution of error types for Qwen2.5-7B-Instruct and Llama3.2-3B-Instruct on the GSM8K test set. The factual drift errors are highlighted in orange and account for a non-negligible proportion in both models.

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 5: Self-verification occurs during DeepSeek-R1's reasoning, where the model revisits the query, focusing on key information, and paraphrases it.

arising from other causes. To annotate these errors, we utilize GLM-4-Plus (GLM et al., 2024), with prompts detailed in Appendix B. The resulting distributions of error types for both models are summarized in Figure 4. As shown, a non-negligible proportion of errors in both models can be attributed to factual drift, highlighting its significance as a failure mode in LLM reasoning.

Another example is from our experiment on developing Stickers. When we use Llama3.2-3B-Instruct (Dubey et al., 2024) to construct Stickers for GSM8K test data (Cobbe et al., 2021), 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

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 $P_1 \not= \rightsquigarrow$ **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 $P_2 \not= \rightsquigarrow$ **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** $P_3 \not= \rightsquigarrow$ **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 EQUIVALENT$(P_S, P_{Q,S})$ **then**
  `// Consensus validation`
  return $P_{Q,S}$
else
  return $\rightsquigarrow$
end

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 5, 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

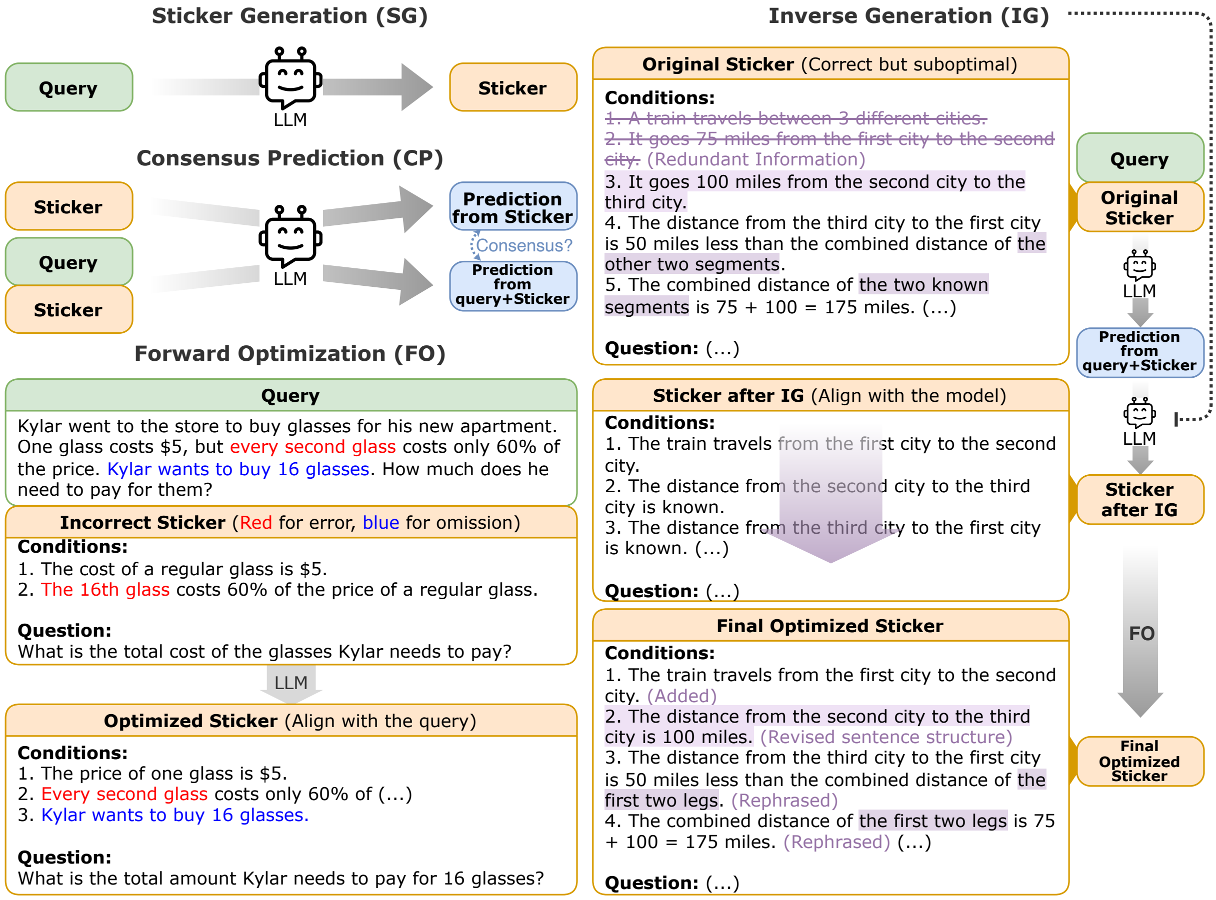

Figure 6: Four core operations in SIFT: (i) Sticker Generation (SG), (ii) Consensus Prediction (CP), (iii) Forward Optimization (FO), (iv) Inverse Generation (IG).

develop the novel SIFT framework to address this.

3.2 Stick to the Facts (SIFT)

Below, we introduce SIFT, with the algorithmic procedure summarized in Algorithm 1. Refer to Figure 6 for the visualization of the four involved operators and Appendix E for the used prompts.

Sticker Generation (SG). To address the factual drift issue identified in LLM reasoning, we focus on encoding the core information of the query into a compact and explicit form, which we call the Sticker. This process emphasizes the essential constraints and facts from the original query, aiming to make critical information more salient to the model and reduce the risk of misinterpretation or omission during downstream reasoning.

Consensus Prediction (CP). Once a Sticker is generated, the model can produce answers in two ways: using the Sticker alone, or using both the Sticker and the original query as input. If the answers differ, this indicates high uncertainty or potential misalignment in the model's understanding—suggesting possible factual drift. If the answers agree, there is a lower risk of factual drift and the prediction is more likely to be reliable. We formalize this procedure as Consensus Prediction (CP), with details summarized in Algorithm 2, which serves as a factual validation mechanism.

Unlike traditional self-consistency methods that aggregate diverse reasoning paths (Wang et al., 2023a), CP focuses on verifying semantic invariance across different problem representations.

Forward Optimization (FO). Despite careful initial construction, Sticker Generation itself may still be subject to factual drift, where key constraints are inaccurately captured or misunderstood. To mitigate this, we introduce Forward Optimization (FO): starting from the generated Sticker, we refine it further using both the original query and the initial Sticker as context. This step helps to better anchor the Sticker to the true semantics of the source query, correcting misinterpretations and clarifying ambiguous information (e.g., fixing "the 16th glass" to "every second glass" as in Figure 6).

Inverse Generation (IG). A noteworthy observation in LLM reasoning is that contexts with identi-

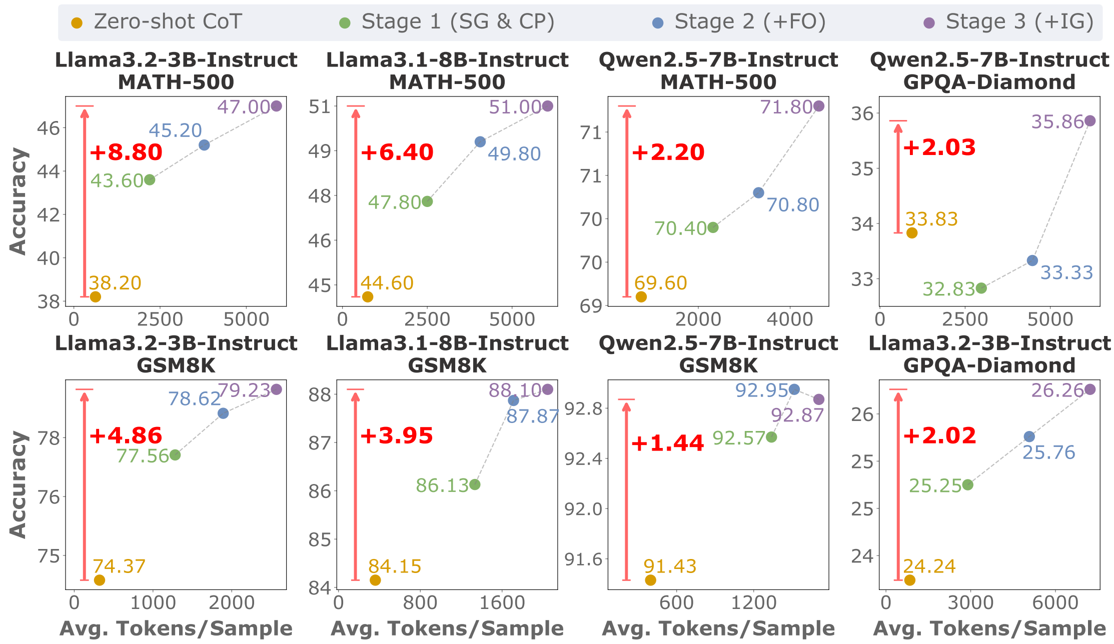

Figure 7: Comparison of SIFT and traditional Zero-shot CoT across multiple models and datasets. We divide SIFT into three stages: Stage 1 only uses SG & CP, while Stage 2 and Stage 3 optimize the Sticker through forward (+FO) and inverse (+IG) direction, respectively. The bidirectional arrows in the 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.

cal semantics but different surface forms can produce different outcomes. To further address potential factual drift and better align the Sticker with the model's internal preferences, we propose Inverse Generation (IG). In this step, a new Sticker is constructed based on the model's own prediction, allowing the representation to better reflect the reasoning patterns favored by the LLM. For example, as shown in Figure 6, an original Sticker might express a condition as "It goes 100 miles from the second city to the third city," while the model, in its own prediction, rephrases it as "The distance from the second city to the third city is 100 miles." Although both statements share the same meaning, their surface forms differ, with the latter more consistent with the model's reasoning patterns. This process facilitates the refinement of the Sticker, making its expression more closely aligned with the model.

4 Experiments

This section first validates 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 and appendix D).

4.1 Enhancing LLM Reasoning with SIFT

Models & Datasets. For details on the models and datasets used in our experiments, see Appendix C.

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 and 7. As observed, SIFT consistently delivers robust and significant performance improvements compared to traditional Zero-shot CoT across all settings. From a methodological perspective, as

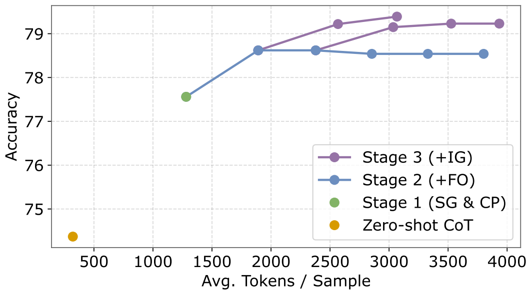

Figure 8: Iterative optimization results for SIFT. The performance improves as the number of tokens per sample increases across different stages. Significant gains are observed in the first repeats of Stage 2 and Stage 3.

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.

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 8. 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) 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 consistently 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

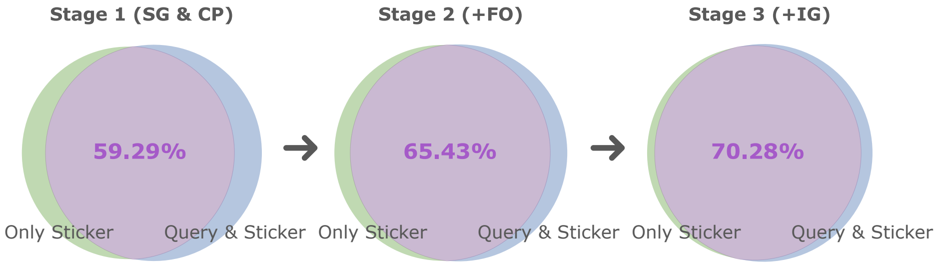

Figure 9: Venn diagrams illustrating the accuracy of predictions obtained from the "Only Sticker" and "Query & Sticker" representations at each stage. The percentages represent the accuracy where both methods correctly predict the same outcomes (i.e., the overlapping purple region). From Stage 1 to Stage 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.

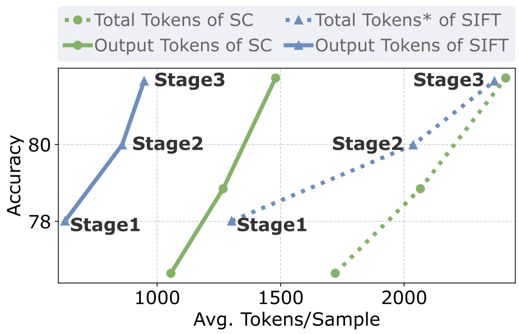

Figure 10: Comparison of SIFT and standard Self-Consistency (SC) in terms of accuracy versus average tokens per sample. The solid lines represent the output tokens used by SC (blue) and SIFT (red), while the dashed lines indicate the total tokens consumed. The "*" symbol in the legend denotes that the 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.

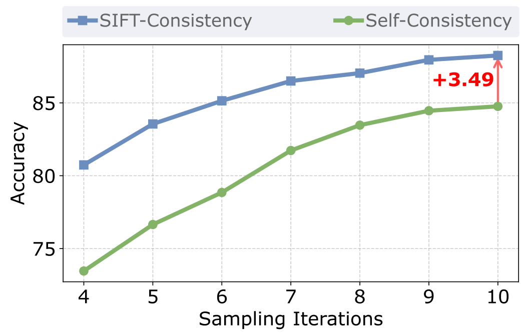

Figure 11: Comparison of SIFT-Consistency and Self-Consistency across different numbers of sampled responses per query. SIFT-Consistency consistently outperforms Self-Consistency.

alignment, We select Llama3.2-3B-Instruct (Dubey 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 9. As shown, both FO and IG significantly improve the alignment of the predictions from the two representations.

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 10. Regarding the total tokens used by both methods, the performance curve of SIFT generally remains above that of SC. 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.

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 Llama3.2-3B-Instruct on the GSM8K dataset. The results are shown in Figure 11. As shown in the figure, SIFT-Consistency consistently outperforms standard SC across different sampling iterations.

For more ablations, see Appendix D.

5 Conclusion

This study presents Stick to the Facts (SIFT), a training-free framework that grounds LLM reasoning in contextual facts through iterative self-refinement. Our approach enhances reasoning reliability without requiring extra data or training.

Limitations

This work focuses on the training-free setting and SIFT require additional tokens. 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*.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

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 Sticker Framework

The design of the *Sticker* framework stems from a critical gap in LLM reasoning: unstructured natural language queries often entangle factual conditions with problem-solving objectives, creating ambiguity that leads to factual misalignment. To resolve this, we explicitly separate the input queries into two components: *Conditions* and *Question*. These components form the structure of the Sticker. An example of an original query and its corresponding Sticker is showed in Figure 1.

B Prompts for Error Type Annotation

To annotate the error types in the GSM8K evaluation, we used GLM-4-Plus (GLM et al., 2024) with the following prompt. For each model prediction, the model is provided with the original question, the standard answer, and the student's (model's) answer. The prompt asks the model to determine whether the error was due to a misunderstanding of the question (factual drift, labeled as `read error`) or a reasoning/calculation mistake (labeled as `reason error`).

```
You are an experienced teacher. Below, I will provide the standard answer, the student's answer, and the original question. Please identify whether the student's error is due to misunderstanding the question or an actual mistake in reasoning or calculation.

If the student misunderstood the question, output: "read error".
If the student made a mistake in reasoning or calculation, output: "reason error".

Question: {question}
Standard Answer: {gold}
Student's Final Answer: {prediction}
```

C 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 AIME2024/2025 (of America, 2024).

D More Results

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.

FO Required Before Adding IG. We investigate 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.

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.

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.

Strategy	Factual Drift Error Rate (↓)
Vanilla CoT	25.93
SIFT (Stage 1)	15.30
SIFT (Stage 2)	15.09
SIFT (Stage 3)	14.73

Table 4: Factual drift error rates on GSM8K using Qwen2.5-7B-Instruct. The results show a progressive reduction in factual drift through the three stages of the SIFT method, compared to the baseline Vanilla CoT.

Factual Drift Mitigation. SIFT employs a two-stage optimization process (forward and backward passes) to refine Stickers, specifically designed to mitigate Factual Drift—a prevalent error type where model responses diverge from original facts. To quantify this effect, we evaluate Qwen2.5-7B-Instruct on GSM8K, measuring the percentage of incorrect answers where the first error is caused by Factual drift, as shown in Table 4.

E Prompting for SIFT

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

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.

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.