More Yap Less Meaning: Uncovering Self-Improvement Behavior in LLMs

Anonymous ACL submission

Abstract

Recently, Large Language Models (LLMs) have made rapid progress across various domains and applications. However, their capability for self-improvement, i.e., whether they are adept at recognising and correcting flaws in their own reasoning, remains dubious. In this study, we address this question by constructing a test of sufficiency to rigorously examine LLMs' self-correction capabilities. We propose a minimal three-step self-correction pipeline that collects initial LLM answers, prompts the same model to generate hints for its incorrect responses given the ground truth, and feeds the model the same question with its own feedback to refine the initial answer. We evaluate a variety of instruction-tuned and reasoning Language Models in this experimental setup on arithmetic and logical reasoning benchmarks. Our findings show that LLMs with injected hint sentences yield only a 4.4% gain over initial question-answering accuracy. Even though the correct answer was provided alongside the model's incorrect reasoning, Language Models fail to understand what was missing in their reasoning and show minimal semantic difference between hints that lead to corrections and ones that do not. Furthermore, our experiments show that longer hints are positively correlated with incorrect final answers, suggesting that longer deliberation on problems can hinder the reasoning process, meaning that LLMs do not necessarily scale in performance with a larger compute budget.

1 Introduction

Large language models (LLMs) have demonstrated strong performance across a broad range of complex NLP tasks compared to prior state-of-the-art approaches (Kalyan, 2023; Wang et al., 2018). This progress is largely enabled by the attention mechanism in Transformer architectures (Vaswani et al., 2017), large-scale pretraining (Kaplan et al., 2020; Kalyan, 2023), and alignment with human preferences through post-training techniques (Kalyan, 2023). Introduction of Chain-of-Thought prompting has further strengthened LLMs' ability to solve non-trivial reasoning tasks by generating step-by-step solutions (Wei et al., 2022). However, LLMs are not perfectly accurate and still produce factual or logical errors (Hadi et al., 2023). In particular, they are prone to make mistakes in problems, requiring extensive multi-step reasoning, where a single incorrect intermediate step can lead to an incorrect final answer, making failures more likely than in single-step tasks (Miao et al., 2023). To address these limitations, one proposed solution is self-correction, a paradigm in which LLMs improve their initial outputs using feedback generated from their own previous responses (Madaan et al., 2023; Miao et al., 2023; Gou et al., 2023; Shinn et al., 2023; Welleck et al., 2023; Huang et al., 2022). However, other studies question the effectiveness of this approach and suggest that LLMs are not yet capable of reliably identifying their own mistakes and improving their performance (Huang et al., 2024; Jiang et al., 2024; Valmeekam et al., 2023). Motivated by this uncertainty, we formulate our research question: Are LLMs truly capable of recognizing the limitations of their own reasoning and generating feedback to improve their responses? If such awareness exists, a model given the correct answer after an incorrect prediction should be able to identify why its reasoning failed. Consequently, it should then produce feedback or hints that help it arrive at the correct solution. We propose a sufficiency test to evaluate the self-improvement abilities of LLMs through a structured three-step evaluation framework, illustrated in Figure 1.

First, the model answers questions from arithmetic and logical reasoning benchmarks, producing chain-of-thought reasoning and final answers. Incorrectly answered questions are then extracted. In the second stage, the model is given its original rea-

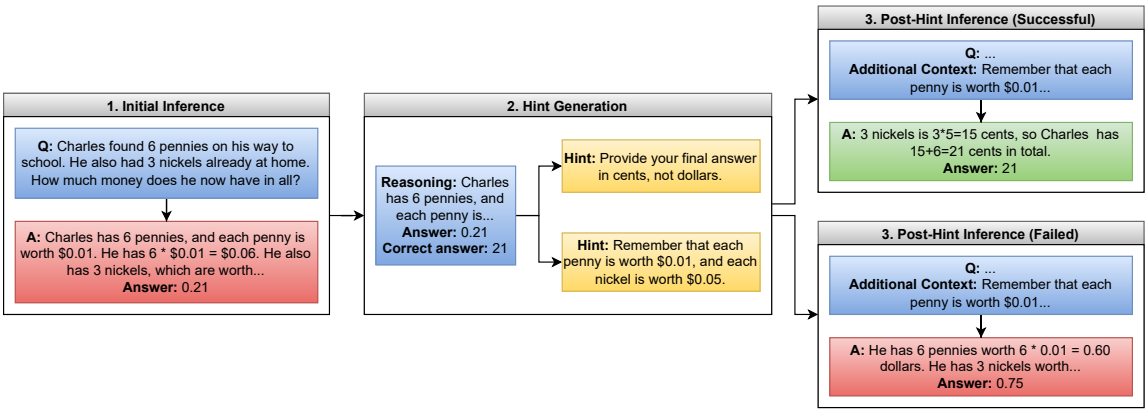

Figure 1: Overview of the three-step self-correction evaluation pipeline.

soning, its incorrect answer, and the ground-truth answer, and is asked to generate hint sentences that could help it reach the correct solution on the first attempt. In the final stage, the model reattempts those problems for which hints were generated, using its own feedback as additional context. We evaluate the change in accuracy after injecting these hints and perform a qualitative analysis of the hint sentences, examining their length and thematic patterns among both effective and ineffective cases to reveal insights into LLMs' self-improvement capabilities. Towards this end, our contributions are the following: **(i)** We establish a novel sufficiency test that allows us to rigorously evaluate LLMs for their self-correction capabilities. **(ii)** We evaluate a variety of smaller-scale LLMs across diverse mathematical and commonsense reasoning domains, showing that LLMs have rather limited capabilities for self-reflection. **(iii)** We further identify a negative correlation between generated hint lengths and answer correction, showing that LLMs do not necessarily scale in performance with an increase in compute budget.

2 Related Work

The term self-improvement can be understood in multiple ways, mainly referring to a process in which an LLM refines its initial response based on the feedback provided after the first generation (Pan et al., 2024). Recent studies distinguish between two types of feedback provided to an LLM: intrinsic and extrinsic. The extrinsic feedback incorporates inputs from external knowledge sources (Gao et al., 2023; Yu et al., 2023), external tools (Gou et al., 2023; Charalambous et al., 2023), and external evaluation metrics (Jung et al., 2022; Welleck et al., 2023). Relatedly, Reflexion (Shinn et al., 2023) converts external reward or evaluation signals into verbal reflections that are stored in memory and used to guide subsequent retries. The main limitation of this method is that high-quality data is not always available for most tasks (Madaan et al., 2023). In contrast, intrinsic feedback is generated by the language model itself and does not rely on any external information, and relies solely on the model's current knowledge. There are several studies exploring LLMs self-improvement using this type of feedback. One is Self-Refine (Madaan et al., 2023), an iterative algorithm that aims to refine the LLM's initial answer based on feedback given by the same LLM over multiple passes. Similarly, SelfCheck (Miao et al., 2023) proposes a zero-shot self-verification approach that regenerates and compares intermediate reasoning steps to detect potential errors without external supervision. Nevertheless, the intrinsic self-improvement method is poorly studied and raises questions in the AI community (Jiang et al., 2024; Valmeekam et al., 2023). Our setup directly challenges the possibility of purely intrinsic self-correction by framing the task as a sufficiency test for self-improvement. Because the model is given the ground-truth answer, identifying flaws in its initial reasoning should be easier than in a purely intrinsic setting, where no such signal is available. This sufficiency-test framing yields a clean upper-bound argument: failure with oracle access implies failure without it. In other words, if a model cannot detect and address reasoning errors even when the correct answer is provided, it is unlikely to do so reliably when the answer is not available. In addition, prior work on R1-style reasoning models shows that, at some point, the accuracy of the model is negatively correlated with

its reasoning length (Marjanović et al., 2026). Our research shows that the same pattern holds for the self-correction paradigm across instruction-tuned and reasoning LLMs.

3 Methodology

Sufficiency Test Given an LLM $\mathcal{M}$, the answering prompt $\mathcal{P}_{Ans} = [T_1, \ldots T_{|\mathcal{P}_{Ans}|}]$ and the question tokens $\mathcal{Q}_i = [T_1 \ldots T_{|\mathcal{Q}_i|}]$ from some dataset $\mathcal{Q}_i \in \mathcal{D}$,we generate an initial LLM answer in the following way:

$\mathcal{A}^i_{init} = \mathcal{M}(\mathcal{P}_{Ans}, \mathcal{Q}_i)$ Afterwards, we filter out only the samples where the model incorrectly predicted the answer to further test its self-correction, i.e. $\mathcal{D}_{incorrect} = \{\mathcal{Q}_i : \mathcal{A}^i_{init} \neq y^i_{gt}\}$, Where y^i_{gt} is the ground truth answer of the question $\mathcal{Q}_i$. Afterwards, we prompt the model $\mathcal{M}$ to generate hint sentences, that does not contain the actual correct answer $\mathcal{H} = [T_1, \ldots T_{|\mathcal{H}|}]$, with a specialized prompt $\mathcal{P}_{hint} = [T_1, \ldots T_{|\mathcal{P}_{hint}|}]$ that explicitly contains model's incorrect chain-of-thought, initial incorrect answer $\mathcal{A}^i_{init}$ and the ground truth answer y^i_{gt} for the designated question $\mathcal{Q}_i$, formalized as $\mathcal{H} = \mathcal{M}(\mathcal{P}_{hint}, \mathcal{Q}_i, \mathcal{A}^i_{init})$. If a generated hint contains answer leakage, the hint is rejected and the model is prompted to regenerate it, with a maximum of 3 attempts. If the model fails to produce an answer-free hint after all attempts, ground-truth answers are masked using the <MSK> token.

To finalise our minimal self-correction loop, we add the generated hint sequence $\mathcal{H}$ after the question and generate a final answer in the following way: $\mathcal{A}^i_{final} = \mathcal{M}(\mathcal{P}_{Ans}, \mathcal{Q}_i, \mathcal{H})$

4 Experimental Setup

Models In our experiment, we evaluate a set of open-weight language models from multiple families with sizes ranging from 1.5B to 8B parameters. We select both instruction-tuned (Meta-Llama-3.1-8B-Instruct (Dubey et al., 2024), Phi-4-mini-instruct (Microsoft, 2024), Qwen2.5-Math-1.5B-instruct, Qwen2.5-Math-7B-instruct (Yang et al., 2024), Gemma-2-2B-It (Gemma Team, Google DeepMind, 2024)), and reasoning-focused (DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-0528-Qwen3-8B, DeepSeek-R1-Distill-Llama-8B (DeepSeek-AI, 2025)) models. For decoding, we follow the temperature and top-p values recommended for each model family in their official releases. Additionally, we use different token budgets of 256, 512, 1024, 1536, and 2048 for the self-correction behaviour.

Benchmarks Our test set consists of question-and-answer problems drawn from diverse LLM benchmarks, including both multiple-choice and free-response formats, and mathematical and commonsense reasoning settings. For numerical reasoning we include GSM8K (Cobbe et al., 2021), ASDiv (Miao et al., 2020), and AQuA (Ling et al., 2017). To examine logical and commonsense reasoning, we use AR-LSAT (Analytical Reasoning) (Zhong et al., 2021) and the Sports Understanding subset from BIGBench (BIG-bench Authors, 2023). More details of the benchmarks can be seen in appendix B.

Evaluation Metrics and Prompts We evaluate self-improvement via the *accuracy gain* after hint injection. Given $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ with predictions $\hat{y}_i^{(0)}$ (initial) and $\hat{y}_i^{(h)}$ (post-hint), we define

$$\begin{aligned} \mathrm{Acc}_{\mathrm{init}} &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\hat{y}_i^{(0)} = y_i\right], \\ \mathrm{Acc}_{\mathrm{hint}} &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\hat{y}_i^{(h)} = y_i\right], \\ \Delta\mathrm{Acc} &= \mathrm{Acc}_{\mathrm{hint}} - \mathrm{Acc}_{\mathrm{init}}. \end{aligned} \tag{1}$$

Prompting details are in appendix C.

5 Results

5.1 Self-correction is hard for LLMs

We evaluate models' self-correction capabilities after receiving self-generated hints across a variety of differing benchmarks and domains. Where available, initial accuracies were consistent with those reported by the model authors, supporting the validity of our generation setup, as shown in Figure 7.

Across model families, datasets, and token budgets, we observe an average post-hint accuracy gain of 4.4%, as can be seen in Figure 3. A more detailed breakdown is shown in appendix A.

The results show that LLMs have limited self-correction capacity, suggesting that hint sentences were ineffective and did not capture the information needed to refine their reasoning. Manual evaluation of the hint sequences showed that they were either too general, contained logical inconsistencies, or lacked meaningful content. In particular, specialist models such as Qwen2.5-Math-1.5B-instruct and Qwen2.5-Math-7B-instruct often hallucinated

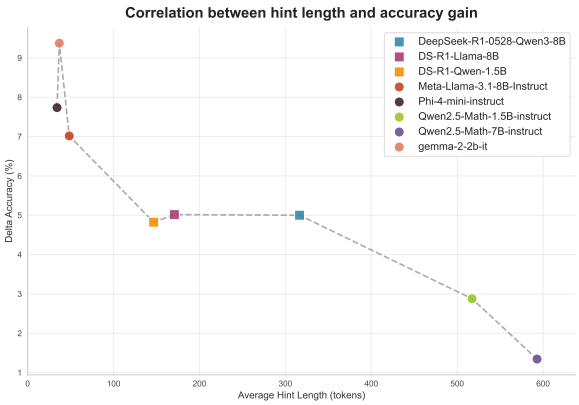

Figure 2: Relationship between average hint length and post-hint accuracy gain across all evaluated models and datasets. Colors correspond to individual models, and marker shapes distinguish model categories, with circles representing instruction-tuned models and squares denoting reasoning models. The horizontal axis shows the average number of generated hint tokens, while the vertical axis reports the change in accuracy after hint injection (Δ Accuracy).

problem continuations and repeated tokens. On average, instruction-tuned and reasoning models yield equal accuracy gains, suggesting that self-refinement capabilities are not associated with the model type, as seen in Figure 5.

5.2 More Yap Less Content

We analyze the relationship between reasoning length and model performance by comparing the number of generated reasoning tokens with post-hint accuracy gain. Our findings, as shown in Figure 2, indicate that hint token length is negatively correlated with post-hint accuracy gain: the longer the hints are on average for a dataset–model pair, the lower the post-hint accuracy gain for that pair. Qualitative analysis of hints shows that in longer hints, the model starts to hallucinate, often producing repetitive reasoning fragments, instruction leakage ("let me think" or formatting rules), unrelated logical statements, or continuing the problem text beyond the required context. This suggests that if the model is not capable of understanding its reasoning flaw at first, it is unlikely to do so after longer generations. The results support prior observations that extended generation can introduce noise rather than improve solution quality.

Correct	%	Incorrect	%
model reasoning	16.1	hockey player	15.4
total cost	16.1	model reasoning	15.4
basketball player	12.9	answer boxed	12.8
hockey player	12.9	basketball	12.8
total distance	12.9	total cost	12.8
answer boxed	9.7	basketball player	10.3
calculate total	9.7	calculate total	10.3
cookies	9.7	help model	10.3

Table 1: Semantic similarity (top keywords) of hint sentences that resulted in correct or incorrect final predictions.

5.3 Useful and useless hints are semantically similar

To understand what distinguishes successful hints from those that do not capture useful information for the model, we employ the KeyNMF algorithm (Kristensen-McLachlan et al., 2024) with the embedding model paraphrase-MiniLM-L3-v2 (Reimers and Gurevych, 2019; Wang et al., 2020) to extract representative keywords and latent semantic topics from generated hints. This analysis in table 1 shows that there exists only a minimal semantic difference between successful and ineffective hints. Hence, classifying corrective feedback based solely on lexical heuristics or predefined reasoning and hinting patterns is complicated, highlighting the intricacy of the self-correction task.

6 Conclusion

We introduced a three-stage sufficiency test pipeline that incorporates both intrinsic and extrinsic approaches: the model is given the ground truth answer during the hint-generation stage (extrinsic), while the final self-improvement relies on the feedback generated by the model. Our results suggest that current LLMs struggle to generate feedback that improves their predictions, even when provided with ground-truth answers. Failure in this setup allows for stronger conclusions about the limitations of self-improvement. This indicates that LLMs are not yet able to reliably identify the limitations of their own reasoning. Furthermore, our analysis reveals that increased token length in generated hints is frequently correlated with poor outcomes. This observation contributes to prior work that longer outputs are not inherently more informative and may even lead to worse performance.

Limitations

Our study has several limitations. Although we evaluate multiple open-weight models from different families and parameter scales, our analysis is restricted to relatively small models (1.5B–8B). Therefore, our conclusions should be interpreted as evidence about self-improvement in smaller language models, rather than as a definitive statement about all frontier-scale systems. In addition, although we attempted to control answer leakage by rejecting hints that explicitly reveal the solution and by masking the ground truth when necessary, leakage detection is imperfect. Subtle semantic leakage may still remain, while some harmless hints may be rejected by conservative filtering. This means that the hint-generation stage may still contain artifacts introduced by the prompt design and filtering procedure.

References

BIG-bench Authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Yiannis Charalambous, Norbert Tihanyi, Ridhi Jain, Youcheng Sun, Mohamed Amine Ferrag, and Lucas C. Cordeiro. 2023. A new era in software security: Towards self-healing software via large language models and formal verification. *Computing Research Repository*, arXiv:2305.14752.

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*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Computing Research Repository*, arXiv:2501.12948.

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

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508, Toronto, Canada. Association for Computational Linguistics.

Gemma Team, Google DeepMind. 2024. Gemma: Open models based on gemini research and technology. *Computing Research Repository*, arXiv:2403.08295.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. CRITIC: Large language models can self-correct with tool-interactive critiquing. *Computing Research Repository*, arXiv:2305.11738.

Muhammad Usman Hadi, Qasem Al Tashi, Rizwan Qureshi, and 1 others. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *TechRxiv*.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *Computing Research Repository*, arXiv:2210.11610.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Dongwei Jiang, Jingyu Zhang, Orion Weller, Nathaniel Weir, Benjamin Van Durme, and Daniel Khashabi. 2024. SELF-[IN]CORRECT: LLMs struggle with discriminating self-generated responses. *Computing Research Repository*, arXiv:2404.04298.

Jaehun Jung, Lianhui Qin, Sean Welleck, Faeze Brahman, Chandra Bhagavatula, Ronan Le Bras, and Yejin Choi. 2022. Maieutic prompting: Logically consistent reasoning with recursive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1266–1279, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Katikapalli Subramanyam Kalyan. 2023. A survey of gpt-3 family large language models including chatgpt and gpt-4. *arXiv preprint arXiv:2310.12321*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Computing Research Repository*, arXiv:2001.08361.

Ross Deans Kristensen-McLachlan, Rebecca M. M. Hicke, Márton Kardos, and Mette Thunø. 2024. Context is key(nmf): Modelling topical information dynamics in chinese diaspora media. *Computing Research Repository*, arXiv:2410.12791.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Computing Research Repository*, arXiv:2303.17651.

Sara Vera Marjanović, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, Nicholas Meade, Dongchan Shin, Amirhossein Kazemnejad, Gaurav Kamath, Marius Mosbach, Karolina Stańczak, and Siva Reddy. 2026. DeepSeek-R1 thoughtology: Let's think about LLM reasoning. *Transactions on Machine Learning Research*.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2023. Selfcheck: Using LLMs to zero-shot check their own step-by-step reasoning. *Computing Research Repository*, arXiv:2308.00436.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Microsoft. 2024. Phi-4 technical report. *Computing Research Repository*, arXiv:2412.08905.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. 2023. Can large language models really improve by self-critiquing their own plans? In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*.

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

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations (ICLR)*.

An Yang and 1 others. 2024. Qwen2.5 technical report. *Computing Research Repository*, arXiv:2412.15115.

Wenhao Yu, Zhihan Zhang, Zhenwen Liang, Meng Jiang, and Ashish Sabharwal. 2023. Improving language models via plug-and-play retrieval feedback. *Computing Research Repository*, arXiv:2305.14002.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2021. Ar-lsat: Investigating analytical reasoning of text. *arXiv preprint arXiv:2104.06598*.

A Appendix

We report the initial accuracies of all of the models across different datasets and token budgets in Figure 4 and Figure 7. Our initial accuracies match the numbers reported by the respective models, thus serving as a sanity check for our study.

Additionally, we report the accuracy gains after hint-based correction across all evaluated models and datasets in Figure 6.

B Benchmarks

GSM8K and ASDiv consist of diverse grade school math word problems, which have a single canonical final answer. AQuA includes multiple-choice math questions covering a broad range of topics and difficulty levels. AR-LSAT (Analytical Reasoning) (Zhong et al., 2021) is a dataset of analytical reasoning questions collected from LSAT exams (1991–2016). It consists of logic problems such as

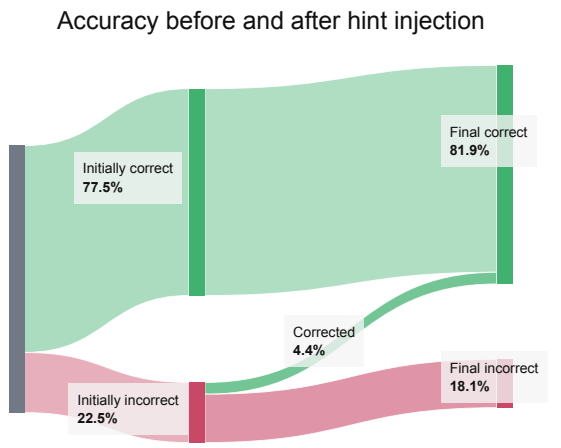

Figure 3: Correction percentages after hint injection averaged across all of the models, tasks and token budgets.

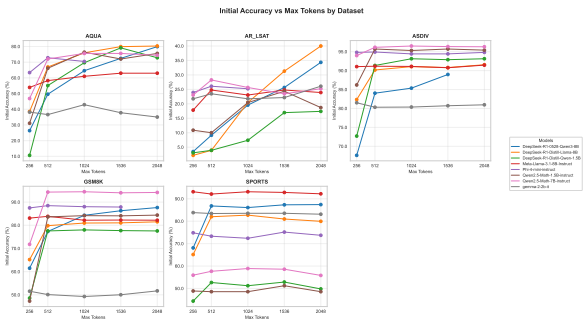

Figure 4: Initial accuracies of all of the models across different tasks and token budgets.

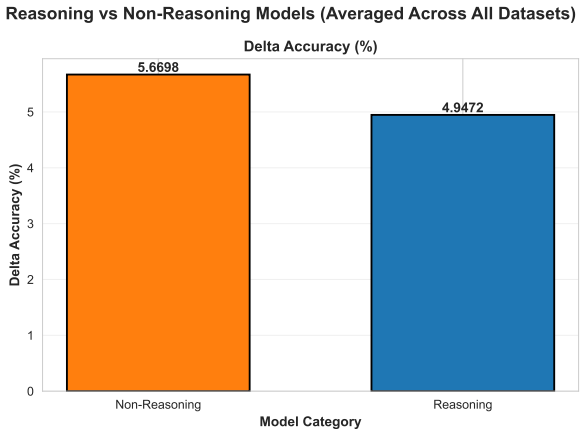

Figure 5: Delta Accuracy of reasoning vs non-reasoning models averaged across all of the datasets.

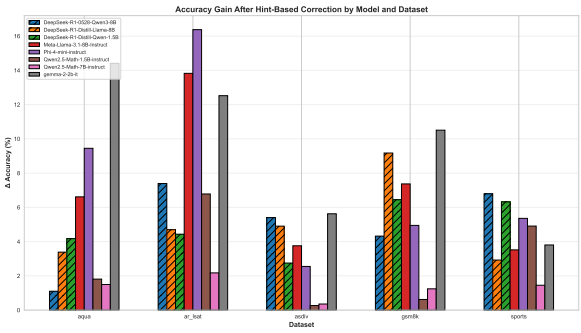

Figure 6: Post-hint accuracy gain (Δ Accuracy) aggregated across token budgets, shown per model and dataset.

assignment, ordering, rule-based constraints, and grouping. The Sports Understanding subset from BIGBench (BIG-bench Authors, 2023) assesses the model's ability to distinguish between plausible and implausible statements related to sports.

C Prompting

During the initial and Post-hint inference stages, we use the Chain of Thought prompting technique to encourage explicit reasoning and to obtain intermediate steps required for subsequent hint generation stages. We use dataset-specific prompt templates with unified formatting rules to ensure consistent results parsing which can be found in appendix A. In the initial and post-hint inference stages, to retrieve model's chain-of-thought and final answer correctly, those parts of the output were prompted to be included in special tags. To help models follow the required format, we employ a few-shot prompting approach, demonstrating six examples during the initial and post-hint inference stages and four examples during hint generation. This setup ensures stable reasoning behavior while maintaining consistent output structure across datasets. We provide the exact prompts used at all stages of the pipeline. Variables enclosed in {} denote placeholders filled at runtime.

C.1 Initial and Post-Hint Answer Generation Prompts

We use dataset-specific answering prompts to standardize model outputs and ensure reliable parsing across all models. Each prompt enforces a two-block structure: a reasoning block delimited by `<think>...</think>` and a final answer block delimited by `<ans>...</ans>`. Variables enclosed in {} denote placeholders filled at runtime. For the post-hint setting, we append the generated hint to the original question while keeping the same output constraints; this isolates the effect of hint injection from formatting differences.

AR-LSAT AR-LSAT is evaluated in a multiple-choice format; therefore, we require the final output to be a single option letter (A–E) and forbid any re-

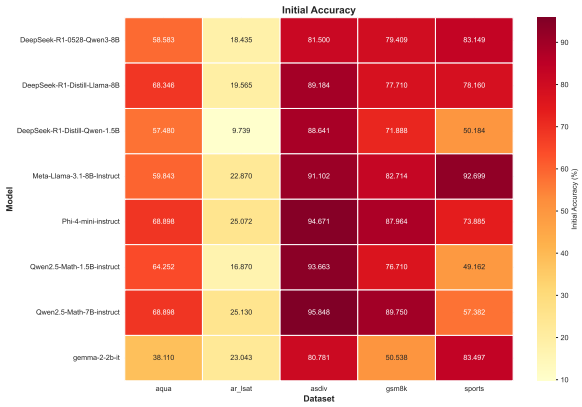

Figure 7: Initial Accuracy (%) Across Models and Datasets

statement of the question/options to avoid leakage into the answer field (table 2).

ASDiv and GSM8K ASDiv and GSM8K require numeric answers. We enforce a strict numeric-only <ans> field (no units, no words) to prevent spurious formatting variations and to keep answer extraction consistent across models (table 3).

AQuA AQuA is also multiple-choice. We apply the same constrained letter-only answer format as AR-LSAT, again ensuring that the final line contains only the option letter for robust evaluation (table 4).

Sports For the sports plausibility task, the target label is binary. We map plausibility to {0,1} in the <ans> block and explicitly state the decision rule in the prompt to minimize ambiguity (table 5).

C.2 Hint Generation Prompts

Hints are generated using the model's original reasoning, its predicted (incorrect) answer, and the ground-truth answer. The hint prompts are designed to provide *helpful guidance without revealing the correct answer*: we disallow explicit mention of the correct option/number and enforce short outputs (1–3 sentences) wrapped in a single <hint>...</hint> block. This design encourages minimal, targeted interventions that can be injected back into the answering prompt while avoiding label leakage.

AR-LSAT For AR-LSAT, the hint must not reference option letters or uniquely identify one choice; it should instead supply neutral guidance that could correct the observed reasoning mistake without giving away the answer (table 6).

ASDiv and GSM8K For numeric word problems, we additionally prohibit directly computing or stating the final numerical result. Hints focus on identifying the correct quantities, operations, and any missing constraints or conversions that led to the initial error (table 7).

AQuA For AQuA, we use the multiple-choice hint template that avoids option letters and avoids stating the correct value explicitly, while still pointing the model toward the correct reasoning step (table 8).

Sports For sports plausibility, we constrain hints to neutral background knowledge (e.g., the athlete's sport/position and the relevant meaning of sport-specific terms) and forbid verdict language such as "plausible/implausible" to reduce answer leakage (table 9).

C.3 Initial and Post-Hint Answer Generation Prompts

Section	Content (verbatim-style, wrapped)
Goal	Your goal is to answer the multiple-choice question by showing step-by-step reasoning between <think> and </think> and writing the final answer choice between <ans> </ans>.
Required output format	You MUST output exactly two blocks, in this order, with no extra text before, between, or after them: <think>step-by-step reasoning</think> <ans>X</ans>
Rules	Rules: - X must be exactly one uppercase letter: A, B, C, D, or E. - Do NOT repeat the question, options, or any example text. - Do NOT output anything else after stating the final answer.
Anti-echo constraint	The following are examples of correctly formatted outputs, you must NOT echo those examples or continue providing your own examples.
Examples	(Excluded from this table for brevity; see codebase / full prompt listing.)
Final instruction + placeholder	-- Answer the following single question. On the last line return only the correct option letter (A-E). Do NOT output anything else after stating the final answer to this question. Question: {question}

Table 2: Instruction template segmented into components (examples excluded).

Section	Content (verbatim-style, wrapped)
Goal	Your goal is to solve math word problem by showing your step-by-step reasoning between <think> and </think> and writing the final numeric answer between <ans> </ans>.
Required output format	You MUST output exactly two blocks, in this order, with no extra text before, between, or after them: <think>step-by-step reasoning</think> <ans>X</ans>
Rules	Rules: - X must be the final numeric answer with no words and no units. Only number. - Do NOT repeat the question or any example text. - Do NOT output anything else after stating the final answer.
Anti-echo constraint	The following are examples of correctly formatted outputs, you must NOT echo those examples or continue providing your own examples:
Examples	(Excluded from this table for brevity; see codebase / full prompt listing.)
Final instruction + placeholder	-- Now answer this single question. Strictly follow the output requirements. Do NOT output anything else after stating the final answer to this question. Question: {question}

Table 3: Math word-problem instruction template segmented into components (examples excluded).

Section	Content (verbatim-style, wrapped)
Goal	Your goal is to answer the multiple-choice question by showing step-by-step reasoning between <think> and </think> and writing the final answer choice between <ans> </ans>.
Required output format	You MUST output exactly two blocks, in this order, with no extra text before, between, or after them: <think>step-by-step reasoning</think> <ans>X</ans>
Rules	Rules: - X must be exactly one uppercase letter: A, B, C, D, or E. - Do NOT repeat the question, options, or any example text. - Do NOT output anything else after stating the final answer.
Anti-echo constraint	The following are examples of correctly formatted outputs, you must NOT echo those examples or continue providing your own examples.
Examples	(Excluded from this table for brevity; see Appendix / full prompt listing.)
Final instruction + placeholder	-- Answer the following single question. On the last line return only the correct option letter (A–E). Do NOT output anything else after stating the final answer to this question. Question: {question}

Table 4: Multiple-choice instruction template segmented into components (examples excluded).

Section	Content (verbatim-style, wrapped)
Goal	Your goal is to answer given plausibility question by showing step-by-step reasoning between <think> and </think> and writing final answer choice between <ans> </ans>.
Required output format	You MUST output exactly two blocks, in this order, with no extra text before, between, or after them: <think>concise step-by-step reasoning</think> <ans>X</ans>
Decision rule for X	Decision rule for X: - X must be exactly one digit: 1 or 0. - Output 1 if the sentence is reasonably true in a real-world sports setting. - Output 0 if it's unlikely or inconsistent in a real-world sports setting.
Rules	Rules: - Do NOT repeat the question or any example text. - Do NOT output anything else after stating the final answer.
Anti-echo constraint	The following are examples of correctly formatted outputs, you must NOT echo those examples or continue providing your own examples.
Examples	(Excluded from this table for brevity; see Appendix / full prompt listing.)
Final instruction + placeholder	-- Now answer this single question. Strictly follow the output requirements. Do NOT output anything else after stating the final answer to this question. Question: {question}

Table 5: Sports plausibility instruction template segmented into components (examples excluded).

Section	Content (verbatim-style, wrapped)
Task	Your job is to look at the model's reasoning, the model's chosen answer, and the correct answer, then write a short HINT that would have helped the model arrive at the correct option if that hint had been included in the original question.
Hint constraints	IMPORTANT RULES FOR THE HINT: - Do NOT include the correct answer, the correct option letter, or any wording that points to one unique option. - Do NOT mention any option letters (A-E) or say any option is wrong/right. - The hint must read like neutral background information or guidance, not a verdict.
Examples	(Excluded from this table for brevity; see Appendix / full prompt listing.)
Generation instruction	-- Now, based on the reasoning, answer, and the correct answer, write a hint to this question.
Inputs / placeholders	Question: {question} Reasoning: {chain_of_thought} Predicted incorrect answer: {predicted_answer} Correct answer: {correct_answer}
Output format requirements	Return ONLY 1-3 hint sentences, wrapped inside a single <hint>...</hint> block, with no extra commentary and WITHOUT stating correct answer. Do NOT state the correct option letter.

Table 6: Hint-generation instruction template segmented into components (examples excluded).

Section	Content (verbatim-style, wrapped)
Task	Your job is to look at the model's reasoning, the model's calculated answer, and the correct answer, then write a short HINT that would have helped the model arrive at the correct answer if that hint had been included in the original question.
Hint format constraints	IMPORTANT FORMAT RULES FOR THE HINT: - Do NOT say whether anything is correct or incorrect. - Do NOT write the correct answer in the hint sentences. - Do NOT mention "the correct answer", "your previous answer", or phrases like "you should answer ...". - Do NOT directly compute or state the final numerical result. - The hint must read like neutral background information or guidance, not a verdict.
What to focus on	Focus on: - Clarifying what each quantity in the problem represents and how they relate. - Highlighting the correct operations and order of operations needed (e.g., which numbers to add, subtract, multiply, or divide first). - Pointing out important constraints, unit conversions, or intermediate steps that were overlooked or misapplied.
Examples	(Excluded from this table for brevity; see Appendix / full prompt listing.)
Generation instruction	-- Now, based on the question, reasoning, predicted answer, and the correct answer, write a new hint.
Inputs / placeholders	Question: {question} Reasoning: {chain_of_thought} Predicted incorrect answer: {predicted_answer} Correct answer: {correct_answer}
Output format requirements	Return ONLY 1-3 hint sentences, wrapped inside a single <hint>...</hint> block, with no extra commentary and WITHOUT stating correct answer. Do NOT try to solve the problem.

Table 7: Numeric hint-generation instruction template segmented into components (examples excluded).

Section	Content (verbatim-style, wrapped)
Task	Your job is to look at the model's reasoning, the model's chosen answer, and the correct answer, then write a short HINT that would have helped the model arrive at the correct option if that hint had been included in the original question.
Hint constraints	IMPORTANT RULES FOR THE HINT: - Do NOT write the correct answer or the correct option letter in the hint sentences. - Do NOT directly compute or state the final numerical result. - Do NOT mention any option letters (A-E) or say any option is wrong/right. - The hint must read like neutral background information or guidance.
Examples	(Excluded from this table for brevity; see Appendix / full prompt listing.)
Generation instruction	-- Now, based on the reasoning, answer, and the correct answer, write a hint to this question.
Inputs / placeholders	Question: {question} Reasoning: {chain_of_thought} Predicted incorrect answer: {predicted_answer} Correct answer: {correct_answer}
Output format requirements	Return ONLY 1-3 hint sentences, wrapped inside a single <hint>...</hint> block, with no extra commentary and WITHOUT stating correct answer. Do NOT state the correct option letter.

Table 8: Multiple-choice hint-generation template segmented into components (examples excluded).

Section	Content (verbatim-style, wrapped)
Task	Your job is to look at the model's reasoning, the model's predicted answer, and the correct answer, then write a short HINT that would have helped the model arrive at the correct answer if that hint had been included in the original question.
Hint format constraints	IMPORTANT FORMAT RULES FOR THE HINT: - Do NOT say whether the sentence is plausible or not. - Do NOT use the words: "plausible", "implausible", "likely", "unlikely". - Do NOT mention "the correct answer", "your previous answer", or "you should answer ...". - The hint must read like neutral background information or guidance, not a verdict.
What to focus on	Focus on: - Clarifying what sport or position the named player is associated with. - Explaining any sport-specific terms or rules that are relevant.
Output style requirements	Return ONLY 1-3 sentences of hint text, wrapped inside a single <hint>...</hint> block, with no bullet points and no extra commentary.
Examples	(Excluded from this table for brevity; see Appendix / full prompt listing.)
Generation instruction	-- Now, based on the question, reasoning, predicted answer, and the correct answer, write hint sentences.
Inputs / placeholders	Question: {question} Reasoning: {chain_of_thought} Predicted incorrect answer: {predicted_answer} Correct answer: {correct_answer}
Final constraint	Return ONLY 1-3 hint sentences, wrapped inside a single <hint>...</hint> block, with no extra commentary and WITHOUT stating correct answer. Do NOT state if the sentence is plausible or not.

Table 9: Sports plausibility hint-generation template segmented into components (examples excluded).