FEEDBACK FRICTION: LLMs Struggle to Fully Incorporate External Feedback

Dongwei Jiang[*†] **Alvin Zhang**[*†] **Andrew Wang** **Nicholas Andrews** **Daniel Khashabi**
djiang21@jhu.edu bzhang90@jh.edu
Johns Hopkins University
[*]Equal contribution [†]Corresponding authors

Abstract

Recent studies have shown LLMs possess *some* ability to improve their responses when given external feedback. However, it remains unclear how effectively and thoroughly these models can incorporate extrinsic feedback. In an ideal scenario, if LLMs receive near-perfect and complete feedback, we would expect them to *fully* integrate the feedback and reach correct solutions. In this paper, we systematically investigate LLMs' ability to incorporate feedback by designing a controlled experimental environment. For each problem, a solver model attempts a solution, then a feedback generator with access to *near-complete* ground-truth answers produces targeted feedback, after which the solver tries again. We evaluate this pipeline across a diverse range of tasks, including math reasoning, knowledge reasoning, scientific reasoning, and general multi-domain evaluations with state-of-the-art language models including Claude 3.7 with extended thinking. Surprisingly, even under these near-ideal conditions, *solver models consistently show resistance to feedback*, a limitation that we term FEEDBACK FRICTION. To mitigate this limitation, we experiment with sampling-based strategies like progressive temperature increases and explicit rejection of previously attempted incorrect answers, which yield improvements but still fail to help models achieve target performance. We analyze FEEDBACK FRICTION and find that models' confidence on specific questions, measured by semantic entropy, predicts feedback resistance: high-confidence predictions remain resistant to external correction. We hope that highlighting this issue in LLMs will help future research in self-improvement.

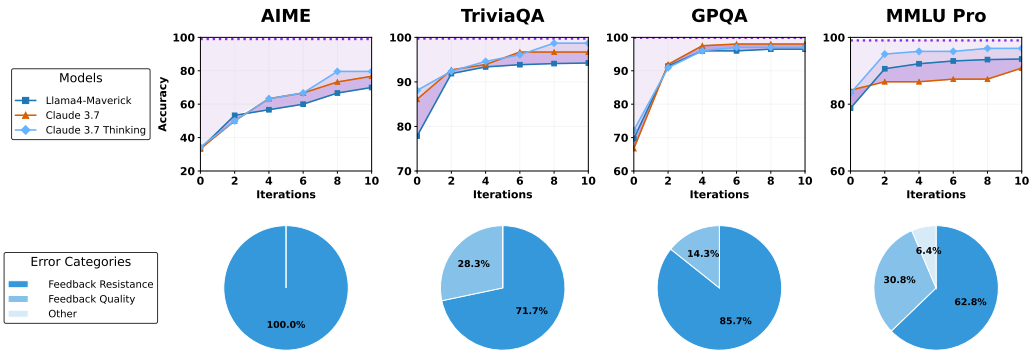


Figure 1: **Top:** Accuracy of various *solver* models when iteratively exposed to feedback from a *feedback* model (GPT-4.1 mini) with access to ground-truth answers. The horizontal dotted line ····· represents the target accuracy models could theoretically attain if they successfully incorporated all valid feedback, excluding those with identified issues (details in §4.1). Despite receiving high-quality feedback, **solver models consistently plateau below their target accuracy**. **Bottom:** Breakdown of questions that remained unsolved by the strongest solver model (Claude 3.7 Thinking) after multiple self-correction attempts. **Feedback resistance is responsible for the majority of persistent errors.**

https://doi.org/10.52202/085713-1831

1 Introduction

The prospect of self-improving Large Language Models (LLMs) has sparked both excitement and debate. Questions persist about LLMs' inherent ability to self-improve without external guidance [1, 2], yet several studies have shown that LLMs can boost their performance when provided with accurate external feedback during test time without any parameter updates or training [3–5]. However, while prior studies establish the *existence* of performance gains from external feedback, *the upper bounds* of such improvement remains largely unexplored. The extent of this improvement would have far-reaching implications for applications such as scientific discovery [6–8] and complex planning [9, 10], where repeated refinement cycles with feedback are critical for success. This raises a critical question: *can models fully incorporate correct feedback to self-improve and reach their maximum potential?*

Answering this question is not straightforward, as self-improvement performance depends on two connected factors: *feedback quality* and *the ability to incorporate feedback*. To decouple these two factors, we create a controlled experimental environment that provides near-optimal conditions for feedback incorporation—one where models receive high-quality, targeted guidance based on complete ground-truth information. Figure 2 demonstrates our setup. The *solver* model attempts to solve problems iteratively, receives feedback from a strong *feedback* model on each incorrect answer, and retries again for up to 10 consecutive iterations. The feedback generator has access to the complete history of all previous attempts and responses, enabling targeted guidance that addresses persistent errors.

While this controlled environment with high-quality feedback provides an ideal testing ground, the effectiveness of feedback incorporation may depend significantly on the *quality* of the feedback. To assess them, we implement three increasingly sophisticated feedback mechanisms to determine how much feedback quality impacts model performance and whether any level of feedback enables models to reach target accuracy. Binary Correctness Feedback (F_1) provides simple indication of correctness (e.g., "the answer is wrong"). Self-Generated Reflective Feedback (F_2) has the model itself analyze potential errors using correct answers and available solution steps (e.g., "you correctly set up the equation but made an error when computing the derivative..."). Finally, Strong-Model Reflective Feedback (F_3) uses a more capable external model (GPT-4.1 mini) to generate feedback. Details of our evaluation framework and feedback generation process can be found in §2.

We conduct a systematic study using all three forms of feedback across strong frontier models including Llama-3.3-70B-Instruct, Llama-4-Scout-17B-16E-Instruct, Llama-4-Maverick-17B-128E-Instruct-FP8, Claude 3.7 Sonnet, and Claude 3.7 Sonnet with Extended Thinking. We evaluate these models across diverse tasks, including AIME 2024, MATH-500, TriviaQA, PopQA, MMLU, MMLU Pro, GPQA, and two synthetic digit multiplication tasks. While higher-quality feedback does improve self-improvement performance, a fundamental limitation persists. As shown in Figure 1, even with our best feedback mechanism—Strong-Model Reflective Feedback (F_3)—models consistently fall short of the target accuracy (i.e., the accuracy of an ideal model if it successfully incorporated all the given feedback). We name this weakness FEEDBACK FRICTION (§3.2).

Our analysis in §4 reveals several key insights into FEEDBACK FRICTION. First, we categorize errors that persist after multiple feedback iterations and find that *feedback resistance* (i.e., models failing to incorporate clear and accurate feedback) *is the dominant failure mode across all tasks* (§4.1). Second, we attempt to mitigate it by avoiding repeated wrong answers through sampling strategies. While performance improves across tasks, models still fall below their target accuracy (§4.2). Finally, we investigate why models resist feedback (§4.3). Our investigations rule out several apparent causes, including reasoning complexity, data familiarity, and whether specific questions consistently challenge all models. However, we uncover two key findings: First, using semantic entropy to measure confidence, we find that less confident models show greater relative improvements from feedback, indicating confident models are more resistant to correction. Second, models consistently claim to understand feedback and express willingness to update their beliefs (>95%) yet fail to actually incorporate corrections—revealing a disconnect between stated intentions and actual behavior. In summary, having shown that state-of-the-art LLMs consistently resist external feedback, we identify important patterns in this resistance while ruling out several intuitive explanations.[1]

[1]Code for this work is available at: `https://github.com/JHU-CLSP/Feedback-Friction`

https://doi.org/10.52202/085713-1831

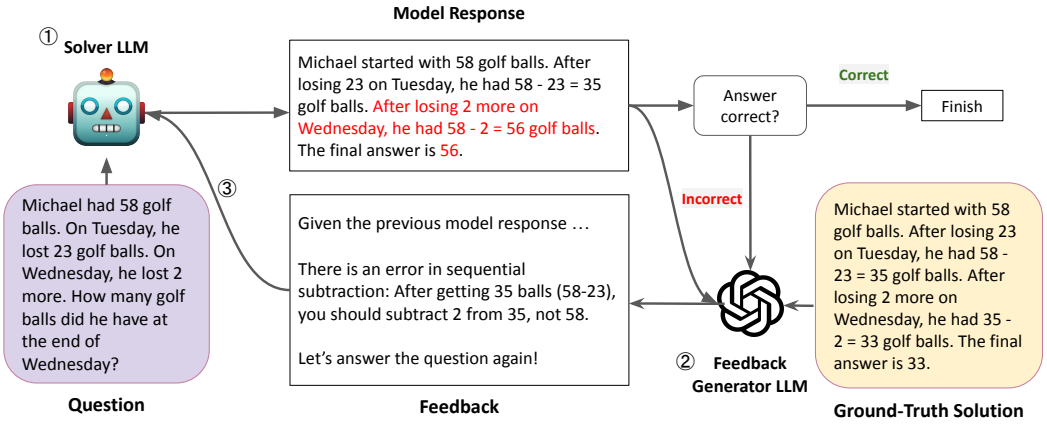


Figure 2: **Iterative self-improvement loop.** The process involves: ① a *solver* model generating an answer, ② a *feedback* model generating feedback given incorrect responses and the ground-truth correct answer, and ③ the *solver* attempting again with this feedback. This cycle repeats for up to 10 iterations or until a correct answer is produced.

2 A controlled framework to surface FEEDBACK FRICTION

Our framework employs two key components: an iterative self-improvement loop that allows models multiple opportunities to correct their mistakes (§2.1), and a spectrum of feedback mechanisms with varying levels of detail and guidance (§2.2).

2.1 Setup for iterative self-improvement loop

Given a task T with evaluation dataset $D = \{(x_i, y_i)\}_{i=1}^{m}$ and evaluation metric f, we establish an iterative improvement protocol with two distinct models: a solver model M_{solver} and a feedback generator model M_{feedback}.

For each input x_i, the solver model produces an initial answer $a_1(x_i)$ using the standard task prompt. We evaluate the correctness of this answer using $f\big(a_1(x_i), y_i\big)$, where y_i is the ground truth. If the answer is incorrect, the feedback generator M_{feedback} creates targeted guidance g_1 based on the current answer and ground-truth information.

For iteration $k \geq 1$, we construct the prompt $p_{k+1}(x_i) = \text{concat}(x_i, \text{history}_k)$, where history_k contains all previous answers and feedback pairs $\{(a_1, g_1), (a_2, g_2), ..., (a_k, g_k)\}$. This process repeats for up to $k = 10$ iterations or until the correct answer is generated. In our empirical experiments, we observed that performance improvements tend to plateau within 10 iterations, suggesting a practical upper limit for the iterative refinement process.

The overall accuracy for the dataset at iteration k is measured as the fraction of all problems solved correctly: $\text{Acc}_k = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}[f\big(a_k(x_i), y_i\big) = 1]$, where $\mathbf{1}[\cdot]$ is the indicator function.

By construction, since we iterate over the incorrect responses only, the accuracy sequence $\{\text{Acc}_1, \text{Acc}_2, ..., \text{Acc}_K\}$ is monotonically non-decreasing, as we retain correct answers across iterations and only modify incorrect ones. This protocol, illustrated in Figure 2, provides a controlled environment for measuring how effectively models incorporate feedback while maintaining consistent evaluation criteria throughout the improvement process.

2.2 Designing different feedback mechanisms for iterative self-improvement

We investigate three distinct feedback mechanisms for the self-improvement process, each offering progressively greater guidance and error specificity. All mechanisms are designed to identify errors *without directly revealing the correct answer*, ensuring a fair evaluation of the model's ability to incorporate feedback.

Binary Correctness Feedback (F_1). The simplest form provides only correctness information:

$$F_1^{\text{binary}}(x_i, y_i) = \text{“The answer is wrong!”} \quad \text{if } f\big(a(x_i), y_i\big) = 0$$

https://doi.org/10.52202/085713-1831

where $a(x_i)$ is the solver model's answer and f is the evaluation function.

Self-Generated Reflective Feedback (F_2). In this approach, the solver model analyzes its own response using available information:

$$F_2^{\text{self}}(x_i, y_i) = M_{\text{solver}}\big(\text{concat}(x_i, a(x_i), y_i, s_i, p_{\text{prompt}})\big)$$

where p_{prompt} is the instruction: "Please give me feedback on which solution step is wrong and how to get to the correct answer without revealing the answer." Here, s_i represents the ground-truth solution process when available. For datasets without detailed solutions, only the answer y_i is provided.

Strong-Model Reflective Feedback (F_3). We employ a more capable external model with access to the same information:

$$F_3^{\text{strong}}(x_i, y_i) = M_{\text{strong}}\big(\text{concat}(x_i, a(x_i), y_i, s_i, p_{\text{prompt}})\big)$$

where M_{strong} represents a more powerful model than M_{solver} in providing feedback.

Examples illustrating the various feedback types can be found in Appendix A).

3 Experimental results

In this section, we present comprehensive experimental results evaluating FEEDBACK FRICTION. We first describe our experimental setup, including tasks, prompts, inference setups, and model configurations (§3.1). We then demonstrate how models consistently plateau below target performance regardless of the feedback mechanisms employed (§3.2).

3.1 Experimental setup

Tasks and metrics. We employ nine diverse tasks for evaluation, deliberately choosing objective tasks with clear ground-truth answers to ensure reliable evaluation of feedback incorporation.[2] Our tasks include: AIME 2024 [11] and MATH-500 [12] for mathematical problem-solving, TriviaQA [13] and PopQA [14] for knowledge reasoning, MMLU [15] and MMLU Pro [16] for multi-domain evaluation, GPQA [17] for complex scientific reasoning, and two synthetic digit multiplication tasks. The first synthetic task involves 5-digit multiplication (e.g., 78934 × 62851), while the second task applies hexadecimal multiplication rules to decimal numbers (i.e., first mapping 0-9 to themselves and 10-15 to their hexadecimal digits ($10 \rightarrow A$, ..., $15 \rightarrow F$), then performing the arithmetic as if operating in base-16—creating a counterfactual [18] arithmetic setting that challenges models' learned numerical reasoning patterns. For both tasks, ground-truth solutions are generated using deterministic templates that break down the multiplication into smaller multiplication and addition operations. We include these synthetic tasks specifically to remove potential confounding variables from semantic context (more details about these tasks can be found in Appendix D).

Other than the two synthetic tasks, AIME 2024, GPQA, and MATH-500 include complete solutions, while the others provide only final answers. For MMLU and MMLU Pro, the multiple-choice format raises concerns about whether models might solve problems through simple elimination strategies rather than genuine reasoning (e.g., selecting A in the first iteration, B in the second, C in the third, etc). However, our analysis reveals that models exhibit surprising choice persistence even when provided with corrective feedback. Rather than systematically eliminating options or switching between choices, models often remain anchored to their initial selections (typically one or two specific answer choices) across multiple feedback iterations, even when that choice is demonstrably incorrect. We use the same prompts, few-shot demonstrations, answer parsing mechanism, and metrics from `lm-evaluation-harness` and `llama-evaluate` where applicable to maintain established evaluation practices and enable fair comparison with prior work. For MMLU, MMLU Pro, PopQA, and TriviaQA, we sample 10% of the total data points, as running the full dataset through our 10-iteration improvement process would be prohibitively time-consuming. Our preliminary experiments confirmed that this subset yields performance metrics nearly identical to those obtained from the complete dataset.

Prompt design and experimental controls. Our experimental framework employs carefully structured prompts to ensure consistent feedback delivery across iterations. For the solver model,

[2]Using another LLM to evaluate more subjective tasks like instruction following or translation could lead to issues like reward hacking and unreliable assessments.

we use task-specific system prompts (detailed in Appendix B) and construct iterative prompts that include complete interaction history. We investigate whether prompt structure and organization affect feedback incorporation by comparing our standard single-prompt approach against a multi-turn conversation format that structures the same information across multiple dialogue exchanges. In the conversation format, we reformatted the iterative improvement process to mimic natural dialogue, with alternating turns between solver attempts (as user) and feedback responses (as assistant). Results showed marginal differences compared to single-prompt formatting, suggesting that FEEDBACK FRICTION persists regardless of interaction structure.

Beyond the feedback mechanisms discussed in §2.2, to maintain evaluation integrity while preserving feedback quality, we implement comprehensive answer masking to prevent feedback from directly revealing ground truth solutions. We use "[masked]" as the replacement token for filtered content, applying targeted masking that preserves intermediate steps and reasoning guidance while preventing direct answer revelation. For example, we replace numerical answers with "[masked]" (e.g., "The final answer is [masked]" instead of "The final answer is 42") while preserving solution steps.[3]

Models and inference. As for *solver models,* we employs strong models including LLaMA-3.3 70B Instruct [19], Llama-4-Scout-17B-16E, Llama-4-Maverick-17B-128E-Instruct-FP8 [20], Claude 3.7 Sonnet and Claude 3.7 Sonnet with extended thinking [21]. Claude 3.7 with extended thinking is a variant that employs an extended reasoning before generating final responses, allowing the model to engage in more deliberate problem-solving through explicit step-by-step thinking. All models are instruction-tuned versions of their respective base models, specifically optimized for handling natural language instructions and maintaining consistent output formatting.

For the Llama models, during inference, we use temperature 0 to ensure deterministic outputs and conduct inference using vLLM [22] with each model's corresponding chat template. All inference is performed on a single H100 instance equipped with eight 80GB GPUs. For Claude models, we access them through Anthropic's API. For Claude 3.7, we use temperature 0 to ensure deterministic outputs, while for Claude 3.7 with extended thinking, we use temperature 1 as suggested by Claude. We also explore the effects of varying these temperature settings in our experiments to mitigate FEEDBACK FRICTION (§4.2).

For *feedback models,* we use different models depending on the feedback type. For Strong-Model Reflective Feedback (F_3), we utilize GPT-4.1 mini [23] as the feedback generation model. From our internal testing, GPT-4.1 mini's feedback performance is on-par with Claude 3.7. Due to the higher cost of Claude 3.7, we use GPT-4.1 mini for generating the strongest feedback. We also considered o4-mini [24] as the feedback generator model due to its reportedly superior reasoning capabilities, but our experiments showed it delivered comparable feedback quality while incurring substantially higher computational costs, leading us to proceed exclusively with GPT-4.1 mini.

3.2 Main findings

FEEDBACK FRICTION persists across model scales and tasks. Figure 3 shows results using our strongest feedback mechanism (Strong-Model Reflective Feedback (F_3)) across all datasets and all models. Our results reveal a striking pattern: Despite receiving high-quality feedback, all solver models consistently plateau below their target accuracy, which is the accuracy of an ideal model if it successfully incorporates all the given feedback (see §4.1 for calculation details). Claude 3.7 Thinking achieves the highest initial accuracy on several tasks (AIME, TriviaQA, GPQA, and MATH-500), while Claude 3.7 shows competitive performance across most benchmarks. However, both Claude variants exhibit the same fundamental plateauing behavior as the Llama models. The performance typically improves rapidly through the first 2-4 iterations before significantly slowing down. This FEEDBACK FRICTION is particularly pronounced on complex reasoning tasks like AIME and GPQA, where even the best-performing models remain 15-25% and 3-8% below their respective theoretical ceilings despite 10 correction opportunities. The synthetic tasks reveal particularly interesting patterns across model families. In the standard 5-Digit Multiplication task, both Claude models reach near-perfect accuracy after significant initial improvement, outperforming the Llama models. However, the Hexadecimal-5-Digit-Multiplication task reveals extreme difficulty with feedback incorporation

[3]We verified that models do not attempt to predict the "[masked]" token during inference

https://doi.org/10.52202/085713-1831

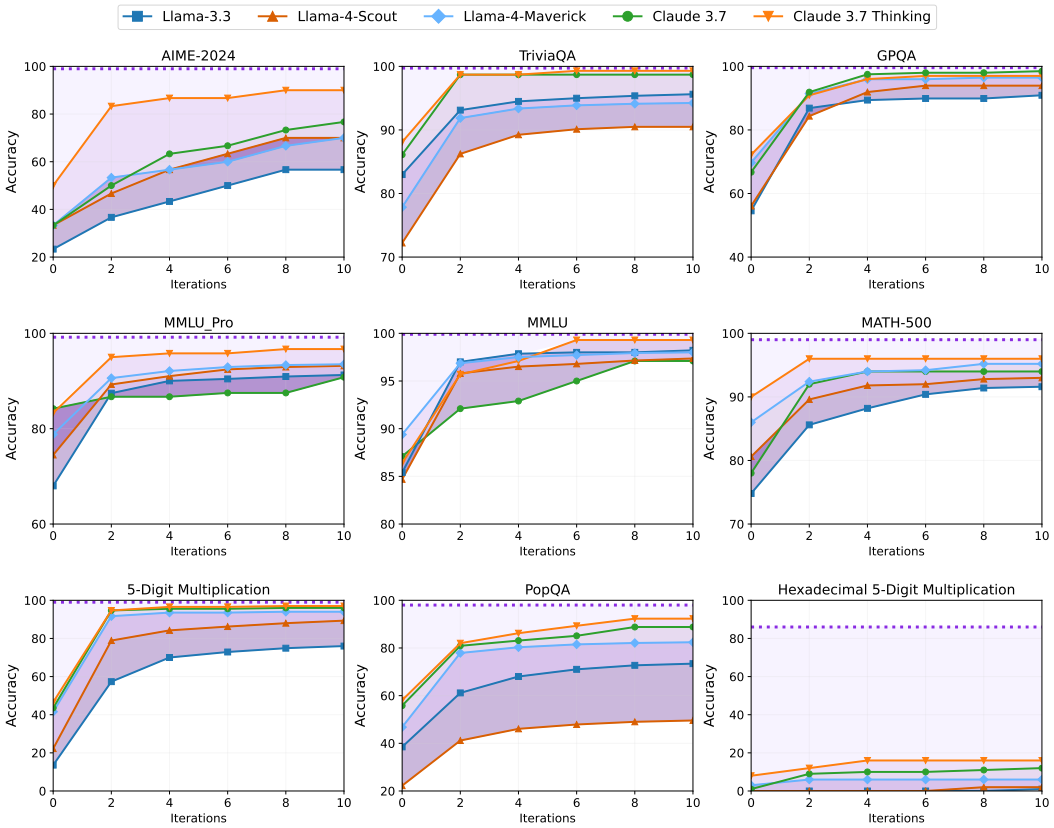


Figure 3: The performance of frontier models we tested with Strong-Model Reflective Feedback (F_3) across nine different tasks. Models are given multiple attempts with feedback that incorporates both the final answer and complete solution (when available). The dotted line ····· represents the target accuracy that models could theoretically achieve if they fully incorporated all feedback (details in §4.1). **Results demonstrate that despite strong feedback, models consistently plateau below their target accuracy across all tasks.**

across all models—no model exceeds 20% accuracy even after 10 iterations, highlighting severe limitations in feedback integration for counterfactual arithmetic systems.[4]

While feedback quality determines the achievable performance ceiling, the extent of its impact is task dependent. Figure 4 illustrates how different feedback mechanisms affect performance across models and tasks. Due to the cost of running extensive experiments with Claude models, we focus the feedback quality analysis on the Llama model families. All tasks show clear benefits from increasingly sophisticated feedback.

The impact of high-quality feedback is most pronounced on complex reasoning tasks. For AIME, MMLU Pro, and GPQA, Strong-Model Reflective Feedback (F_3) outperforms binary feedback by significant margins across all models. Llama-4-Maverick shows the strongest overall performance, achieving 73.3% accuracy on AIME and 96.5% on GPQA with Strong-Model Reflective Feedback (F_3)—-improvements of +26.7% and +10.6% over binary feedback, respectively. Llama-4-Scout demonstrates the largest relative gains, particularly on AIME (+33.3%) and GPQA (+13.1%), compared to Llama-3.3 which shows more modest improvements (+26.7% on AIME, +6.6% on GPQA). Nevertheless, despite these substantial improvements, all models still plateau significantly below their theoretical performance ceiling.

[4]While it is true that models' imperfect understanding of hexadecimal arithmetic leads to imperfect feedback quality, this is reflected in the lower theoretical ceiling shown in the figure. Even accounting for this limitation, models still fall short of what they could achieve if they fully incorporated the available feedback.

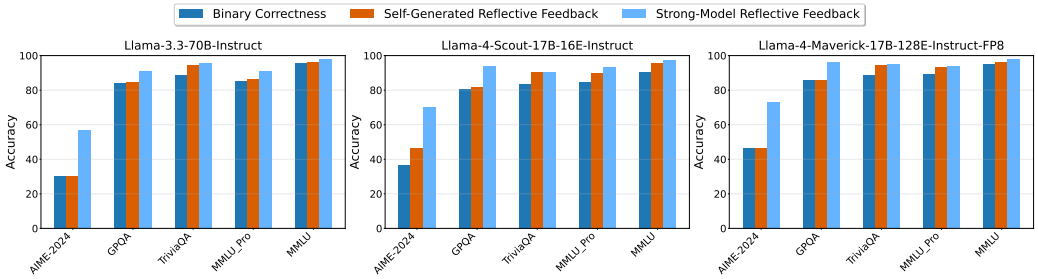


Figure 4: Performance comparison across benchmark datasets using different feedback mechanisms with Llama-3.3, Llama-4-Scout and Llama-4-Maverick. **Model performance progressively improves as feedback quality increases from Binary Correctness Feedback (F_1) to Strong-Model Reflective Feedback (F_3).**

4 Analysis of FEEDBACK FRICTION

We conduct a deeper analysis to better understand FEEDBACK FRICTION. We first categorize different cases where models fail to correct their mistakes despite multiple rounds of feedback (§4.1), then examine the extent to which we can alleviate this problem with sampling strategies (§4.2), and finally, we present several hypotheses and the experiments to understand FEEDBACK FRICTION (§4.3).

4.1 Feedback integration failures dominate persistent self-improvement errors

Error category development. We manually examine cases where LLMs fail to improve despite receiving high-quality feedback and identifies three main categories of error: (1) Most critically for our analysis, **feedback resistance failures** represent cases where models fail to accurately incorporate feedback despite multiple iterations. (2) **Feedback quality issues** encompass cases where the provided feedback is incorrect, ambiguous, or fails to address the key problematic steps in the solution. This can still occur because the generated feedback might miss crucial conceptual errors or introduce new inaccuracies, even though ground-truth is provided to the feedback generator. (3) We maintain an **"Other"** category for cases that don't clearly fit into either of the above categories. From our initial examination, this includes cases where the problem itself contains ambiguities or the solution is conceptually correct but fails due to style or formalization issues (e.g., providing the correct information but not in the expected format required by the evaluation metric).

Automated error categorization and validation. To systematically categorize errors at scale, we used OpenAI o4-mini as an automated annotator to classify complete self-improvement trajectories from Claude 3.7 and Claude 3.7 Thinking according to these predefined categories. To validate this automated approach, we conducted rigorous manual verification by randomly sampling 50 errors from each task and having two human annotators independently label them according to our defined categories. Our verification showed 96% agreement between human annotators and o4-mini's classifications, significantly higher than the 78% agreement rate achieved with GPT-4.1 mini on the same samples. This confirms o4-mini's reliability for this analysis task.

Table 1: Distribution of error categories (%) of unsolved problems after 10 iterations of self-improvement classified by o4-mini. FR: Feedback Resistance, FQ: Feedback Quality, OTH: other issues.

Dataset	Solver Model	FR	FQ	OTH
MMLU Pro	Claude 3.7	64.6	28.0	7.4
	Claude 3.7 Thinking	62.8	30.8	6.4
GPQA	Claude 3.7	100.0	0.0	0.0
	Claude 3.7 Thinking	85.7	14.3	0.0
TriviaQA	Claude 3.7	72.4	25.0	2.6
	Claude 3.7 Thinking	71.7	28.3	0.0
AIME 2024	Claude 3.7	100.0	0.0	0.0
	Claude 3.7 Thinking	100.0	0.0	0.0

Feedback resistance dominates error patterns. Table 1 presents the distribution of error categories across different tasks, as classified by o4-mini. Feedback resistance is consistently the dominant category across all tasks, accounting for 62.8-100% of errors. This finding suggests that the core challenge in achieving perfect performance lies not in the quality of feedback or problem complexity, but in fundamental limitations of how models process and incorporate corrective feedback. Detailed examples of each error category are provided in Appendix C.

https://doi.org/10.52202/085713-1831

4.2 Mitigating FEEDBACK FRICTION with sampling strategies

Given the persistent plateau in performance we observed across models and tasks, a natural question arises: *can we mitigate* FEEDBACK FRICTION *through existing strategies?* We explore sampling techniques as a potential solution to help models overcome their apparent resistance to feedback.

Progressive temperature increases show modest effectiveness. We first explore temperature-based sampling strategies where the sampling temperature increases with the iterations: 0.0 for iteration 0, 0.15 for iteration 1, 0.3 for iteration 2, and so forth. While other schedules (e.g., exponential increase, fixed higher temperature) could be explored, we chose this linear progression as a simple baseline that gradually introduces diversity while preserving early deterministic behavior. We hypothesize that progressive temperature increases would help models generate more diverse outputs, allowing them to escape from local optima in their output distributions and become more receptive to feedback.

Due to the cost of running extensive experiments with Claude models, we focus the feedback quality analysis on Llama-4-Scout and Llama-4-Maverick. As shown in Figure 5, this approach alone produced minimal improvements compared to the baseline in Figure 3. Analysis of logs revealed that while increased temperature successfully diversified model outputs, the additional exploration often failed to converge on correct answers due to the vast search space of possible responses.

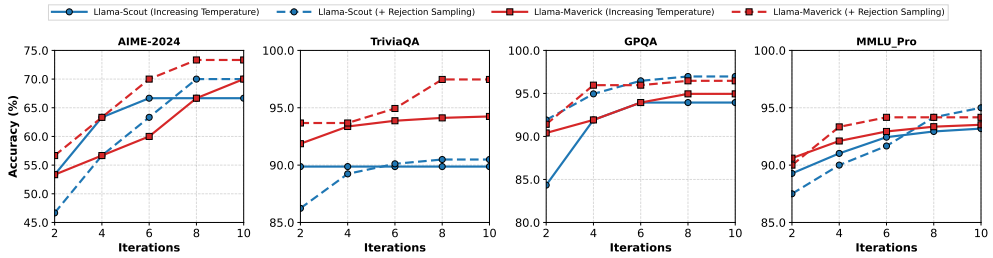


Figure 5: Results of using progressively increasing temperature and rejection sampling with Llama-4-Scout and Llama-4-Maverick. **Rejection sampling can provide additional improvements over temperature-based sampling alone across both multiple-choice and non-multiple-choice tasks.**

Combining temperature increases with rejection sampling yields better results. To enhance performance further, we implement a more targeted approach that combines increased temperature with rejection sampling. This method explicitly forces the model to explore new solution paths while avoiding previous attempts. Specifically, we instruct the model to generate 25 answers, and remove final answers that occurred in previous iterations (which by construction is incorrect, since we only continue iterating on problems that remain unsolved). If no answer remains after this filtering process, we randomly select one from those 25 answers. Otherwise, we randomly select one of the remaining novel answers as the final prediction.

As shown in Figure 5, the combined strategy yields substantive performance gains across both multiple-choice datasets and non-multiple-choice datasets compared to the baseline that only increases the temperature. While the magnitude of these improvements varied, the consistent pattern suggests that forcing models to explore new solution paths by rejecting previously used answers is beneficial. Despite these gains, we observed that all datasets still fall short of the target accuracy, indicating that sampling strategies alone cannot fully resolve model resistance to feedback.

4.3 Understanding FEEDBACK FRICTION

Our sampling strategies (§4.2), though promising, did not eliminate FEEDBACK FRICTION entirely. Developing more effective interventions requires a deeper understanding of the fundamental causes. In this section, we investigate several hypotheses for why models resist incorporating feedback despite multiple correction opportunities. Throughout our analyses, we provide error bars. They represent the standard error of a binomial proportion, calculated as $\sqrt{\text{acc} \cdot (1 - \text{acc})/n}$, where n is the number of samples in each evaluated group.

Model confidence vs. FEEDBACK FRICTION. Could excessive model confidence explain resistance to feedback in FEEDBACK FRICTION? To test this, we measure confidence using *semantic entropy* [25], a method that captures uncertainty at the meaning level rather than surface form

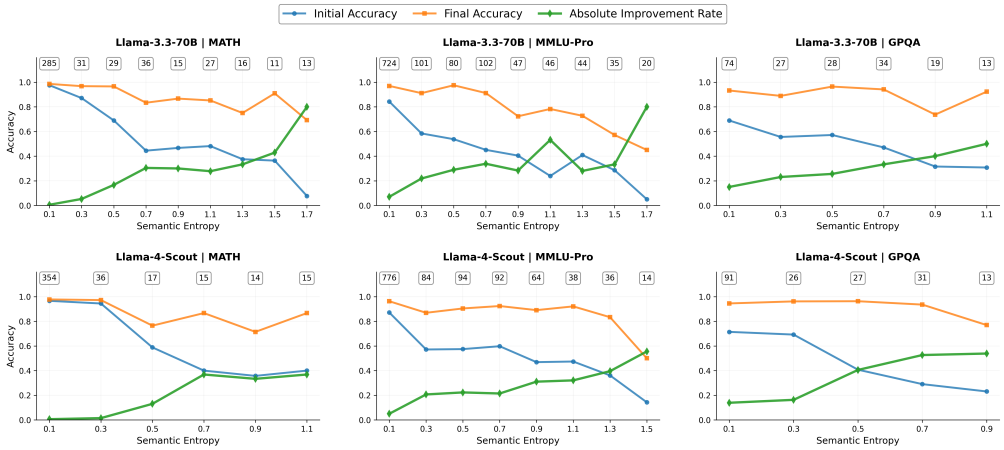


Figure 6: Relationship between semantic entropy and feedback incorporation across three benchmark tasks (MATH, MMLU-Pro, GPQA) for Llama-3.3-70B and Llama-4-Scout models. Semantic entropy (x-axis) measures model uncertainty, with lower values indicating higher confidence. Three metrics are shown: initial accuracy before any feedback (blue), final accuracy after iterative feedback (orange), and absolute improvement rate calculated as $(Final - Initial)/(1 - Initial)$ (green). Numbers in boxes indicate sample sizes per bucket. The absolute improvement rate generally increases with semantic entropy across all tasks and models, indicating that models with lower initial confidence (higher entropy) show greater relative improvements from feedback.

variations. Unlike token-level probability measures, semantic entropy accounts for the fact that models can express the same meaning in multiple ways, providing a more robust measure of true uncertainty.

To compute semantic entropy for each model response, we first generate multiple outputs $\{s_1, s_2, \ldots, s_n\}$ by sampling the model $n = 50$ times with temperature 0.7, where each s_i represents a complete answer given the feedback. We then cluster these outputs using final answers, which we find to have similar results as semantic equivalence grouping using bidirectional entailment. For each semantic cluster C_i, we estimate its probability as $\hat{P}(C_i|x) = \sum_{s \in C_i} P(s|x) / \sum_{j=1}^{K} \sum_{s \in C_j} P(s|x)$, where x represents the input (question, previous answer, and feedback), and compute semantic entropy as $H = -\sum_i \hat{P}(C_i|x) \log \hat{P}(C_i|x)$. Lower entropy indicates the model consistently generates semantically equivalent answers (high confidence), while higher entropy suggests diverse semantic meanings (low confidence).

We conduct this analysis across 5 digits multiplication, GPQA, MATH, MMLU-Pro, and TriviaQA using semantic entropy with bucket size 0.2. Figure 6 shows some of the results (full results in Appendix J) and reveals a consistent pattern: absolute improvement rate (the green trends) increases with semantic entropy, rising from near-zero at low entropy (high confidence) to 0.4-0.8 at higher entropy levels (low confidence). These results suggest that models with lower confidence (higher semantic entropy) have both more room for improvement and greater receptiveness to feedback, while highly confident models show minimal improvement despite receiving the same quality feedback.

Model self-perception versus actual behavior. To probe deeper into FEEDBACK FRICTION mechanisms, we directly asked solver models about their understanding of feedback and willingness to incorporate it. After receiving feedback, we prompt models with: (1) "Do you understand this feedback?" and (2) "Will you update your belief or understanding about this problem?"

Surprisingly, across all models and tasks, solver models consistently claimed to understand the feedback ($> 95\%$ "yes" responses) and expressed willingness to update their beliefs ($> 96\%$ "yes" responses). However, in subsequent iterations, these same models failed to actually incorporate the feedback. This disconnect between stated intention and actual behavior reveals a fundamental issue: models exhibit self-assessment failure where they believe they are incorporating feedback while demonstrably failing to do so (the prompts used for probing and example conversations can be found in Appendix I). This disconnect between stated intention and actual behavior reveals a fundamental implementation failure: models express both understanding and willingness to change but fail to execute these changes in practice.

https://doi.org/10.52202/085713-1831

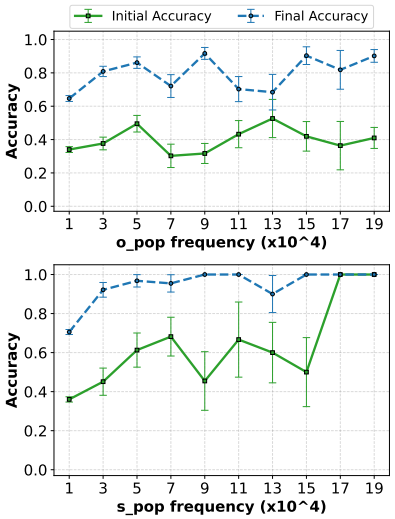


Figure 7: Accuracy of Llama3.3 on PopQA with respect to s_pop and o_pop

Data familiarity and FEEDBACK FRICTION Prior work [14, 26] suggests that language models perform better with familiar entities and topics encountered frequently during training. Are these models more resistant to feedback about familiar entities? We investigated whether this familiarity bias contributes to FEEDBACK FRICTION using PopQA's popularity metrics as proxies for entity familiarity. This dataset includes monthly Wikipedia page views for subject entities s_{prop} and object entities o_{prop}. We then analyzed how accuracy changes during iterative self-improvement correlate with the popularity of the subject (s_prop) and object (o_prop) entities.

We use Llama-3.3 for this experiment since it was released closer to the publication date of PopQA. These popularity metrics provide a particularly relevant measure of potential training data frequency for this model. As shown in Figure 7, we found no consistent pattern between entity popularity and accuracy. We provide further supporting evidence in Appendix F with additional statistical testing and alternative popularity metrics.

Further analyses. We conducted additional analyses to investigate whether problem complexity or inherent question characteristics correlate with feedback integration failure. Both hypotheses yielded negative results. Complete details are provided in Appendix G and Appendix H.

5 Related Work

Self-Improvement with LLMs. Self-improvement in AI has roots in early work using GANs for self-generated feedback [27, 28]. LLMs have expanded these capabilities across code generation [29], reasoning [30**?** , 31], instruction following [32–34], and other domains [35–37]. Approaches vary between training time improvements through self-correction [38] or data generation [39, 40], and inference time improvements via feedback [3, 30] or other models [41]. While studies question intrinsic self-improvement without external feedback [1, 2], consensus exists that LLMs can self-improve with ground-truth feedback [5, 42]. In this work, we probe these limits and investigate barriers to full feedback integration.

The elasticity–plasticity tension. LLMs face a trade-off between retaining prior knowledge and integrating new information. In continual learning, they exhibit catastrophic forgetting when updates conflict with existing beliefs [43–45]. In model editing, extensive edits distort broader knowledge networks [46, 47]. Alignment techniques like RLHF instill desired behaviors but models remain tied to pre-trained distributions or revert via adversarial prompts [48, 49]. In agentic settings, LLMs fail to adapt when encountering external failures [50]. Conversely, they show excessive plasticity through sycophantic behavior in conversations [51]. We revisit this dilemma through feedback integration during self-improvement and unveil controlling factors.

Feedback for LLMs. Feedback mechanisms include intrinsic self-evaluation [52–55] and extrinsic feedback via tools [56–58], information sources [59, 60], or ground-truth answers [3, 61]. While correct feedback is key [5, 62], LLMs struggle to incorporate it when contradicting prior knowledge [63], handling refutations [64], assessing uncertainty [65], or facing misleading feedback [66, 67]. Different from previous work, in this paper, we focus on high-quality feedback, investigating why LLMs fail to fully incorporate it.

6 Conclusion

Our study reveals a fundamental limitation in LLMs' ability to incorporate external feedback. Despite receiving high-quality feedback over multiple iterations, models consistently plateau below their theoretical performance ceiling across diverse reasoning tasks. Despite extensive analysis, the precise mechanisms underlying feedback resistance remain elusive, which we leave to future work. Understanding and addressing these limitations remain essential for developing more adaptable AI systems capable of genuine and sustained self-improvement.

Limitations

Better understanding of FEEDBACK FRICTION While our study identifies key insights into FEEDBACK FRICTION—including the correlation between semantic entropy and feedback resistance and the disconnect between models' stated understanding and actual behavior, we lack a definitive mechanistic explanation for why models resist incorporating feedback. A better understanding of FEEDBACK FRICTION likely involves complex interactions between feedback understanding, instruction following, and belief updating. Future work would benefit from more sophisticated mechanistic interpretability techniques, such as causal intervention methods and circuit analysis [68, 69], to understand the specific computational pathways through which feedback resistance emerges and persists across model architectures.

Limited mitigation strategies Despite extensive experimentation with sampling strategies, our attempts to fully mitigate feedback friction yields only modest improvements. Recent work has explored techniques to enhance diversity in LLM outputs through reinforcement learning [70] and training-free prompting strategies [71], which may help models generate more varied responses to feedback. Another promising approach is to perform supervised fine-tuning or reinforcement learning that could (1) enhance the solver model's receptiveness to feedback incorporation, and (2) improve the feedback generator's ability to provide more effective guidance [72]. However, the computational constraints of our experimental setup, particularly the large model sizes we tested (including 70B+ parameter models like Claude 3.7 and Llama-4-Maverick) and the associated computational costs, prevented us from conducting fine-tuning experiments that might meaningfully address feedback resistance. We leave that exploartion to future work.

Acknowledgment

This work is supported by ONR grant (N00014-241-2089). The GPUs were provided by the DSAI and ARCH clusters. We sincerely thank Yuqing Yang, Zhengping Jiang, and the broader JHU community for discussions and feedback.

References

[1] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2023. URL https://arxiv.org/abs/2310.01798.

[2] Dongwei Jiang, Jingyu Zhang, Orion Weller, Nathaniel Weir, Benjamin Van Durme, and Daniel Khashabi. Self-[in]correct: Llms struggle with discriminating self-generated responses, 2024. URL https://arxiv.org/abs/2404.04298.

[3] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. In *NeuralPS*, 2023. URL https://arxiv.org/abs/2303.11366.

[4] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://voyager.minedojo.org/assets/documents/voyager.pdf.

[5] Gladys Tyen, Hassan Mansoor, Peter Chen, Tony Mak, and Victor Carbune. Llms cannot find reasoning errors, but can correct them! *CoRR*, 2023. URL https://arxiv.org/abs/2310.01798.

[6] Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers, 2024. URL https://arxiv.org/abs/2409.04109.

[7] Akari Asai, Jacqueline He, Rulin Shao, Weijia Shi, Amanpreet Singh, Joseph Chee Chang, Kyle Lo, Luca Soldaini, Sergey Feldman, Mike D'arcy, David Wadden, Matt Latzke, Minyang Tian, Pan Ji, Shengyan Liu, Hao Tong, Bohao Wu, Yanyu Xiong, Luke Zettlemoyer, Graham Neubig, Dan Weld, Doug Downey, Wen tau Yih, Pang Wei Koh, and Hannaneh Hajishirzi. Openscholar: Synthesizing scientific literature with retrieval-augmented lms, 2024. URL https://arxiv.org/abs/2411.14199.

 https://doi.org/10.52202/085713-1831

[8] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

[9] Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning, 2023. URL https://arxiv.org/abs/2308.13724.

[10] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents, 2024. URL https://arxiv.org/abs/2402.01622.

[11] HuggingFaceH4. Aime 2024 dataset. https://huggingface.co/datasets/HuggingFaceH4/aime_2024, 2024.

[12] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

[13] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017. URL https://arxiv.org/abs/1705.03551.

[14] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. In *Annual Meeting of the Association for Computational Linguistics* (ACL), 2023. URL https://arxiv.org/abs/2212.10511.

[15] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2020.

[16] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

[17] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

[18] Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks, 2024. URL https://arxiv.org/abs/2307.02477.

[19] Meta AI. Llama 3. https://www.llama.com/models/llama-3/, 2024.

[20] Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025.

[21] Anthropic. Claude 3.7 sonnet and claude code. https://www.anthropic.com/news/claude-3-7-sonnet/, 2024.

[22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

[23] OpenAI. Introducing gpt-4.1 in the api. https://openai.com/index/gpt-4-1/, 2024.

[24] OpenAI. Introducing openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/, 2024.

[25] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630, 2024.

[26] R. Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L. Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *CoRR*, 2023. URL https://arxiv.org/abs/2309.13638.

[27] Sandeep Subramanian, Sai Rajeswar, Francis Dutil, Chris Pal, and Aaron Courville. Adversarial generation of natural language. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 241–251. Association for Computational Linguistics, 2017. URL https://aclanthology.org/W17-2629/.

[28] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473*, 2017. URL https://arxiv.org/abs/1609.05473.

[29] Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (stop): Recursively self-improving code generation, 2024. URL https://arxiv.org/abs/2310.02304.

[30] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *CoRR*, 2023. URL https://arxiv.org/abs/2303.17651.

[31] Deepak Nathani, David Wang, Liangming Pan, and William Yang Wang. Maf: Multi-aspect feedback for improving reasoning in large language models, 2023. URL https://arxiv.org/abs/2310.12426.

[32] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Model with Self Generated Instructions. In *Annual Meeting of the Association for Computational Linguistics* (ACL), 2023. URL https://arxiv.org/abs/2212.10560.

[33] Seonghyeon Ye, Yongrae Jo, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, and Minjoon Seo. Selfee: Iterative self-revising llm empowered by self-feedback generation. Blog post, May 2023. URL https://kaistai.github.io/SelFee/.

[34] Dongwei Jiang, Guoxuan Wang, Yining Lu, Andrew Wang, Jingyu Zhang, Chuyu Liu, Benjamin Van Durme, and Daniel Khashabi. Rationalyst: Mining implicit rationales for process supervision of reasoning, 2025. URL https://arxiv.org/abs/2410.01044.

[35] Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. Iterative translation refinement with large language models, 2024. URL https://arxiv.org/abs/2306.03856.

[36] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search, 2023. URL https://arxiv.org/abs/2305.03495.

[37] Shukang Yin, Chaoyou Fu, Sirui Zhao, Tong Xu, Hao Wang, Dianbo Sui, Yunhang Shen, Ke Li, Xing Sun, and Enhong Chen. Woodpecker: hallucination correction for multimodal large language models. *Science China Information Sciences*, 67(12), December 2024. ISSN 1869-1919. doi: 10.1007/s11432-024-4251-x. URL http://dx.doi.org/10.1007/s11432-024-4251-x.

[38] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning, 2024. URL https://arxiv.org/abs/2409.12917.

[39] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *CoRR*, 2024. URL https://arxiv.org/abs/2401.01335.

 https://doi.org/10.52202/085713-1831

[40] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *CoRR*, 2024. URL https://arxiv.org/abs/2401.10020.

[41] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *International Conference on Learning Representations* (ICLR), 2023. URL https://arxiv.org/abs/2211.00053.

[42] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies, 2023. URL https://arxiv.org/abs/2308.03188.

[43] Simone Clemente, Zied Ben Houidi, Alexis Huet, Dario Rossi, Giulio Franzese, and Pietro Michiardi. In praise of stubbornness: The case for cognitive-dissonance-aware knowledge updates in llms. *arXiv preprint arXiv:2502.04390*, 2025.

[44] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

[45] Yifei Ming, Senthil Purushwalkam, Shrey Pandit, Zixuan Ke, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. Faitheval: Can your language model stay faithful to context, even if "the moon is made of marshmallows", 2025. URL https://arxiv.org/abs/2410.03727.

[46] Qi Li, Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Xinglin Pan, and Xiaowen Chu. Should we really edit language models? on the evaluation of edited language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[47] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning (ICML)*, 2022.

[48] Jiaming Ji, Kaile Wang, Tianyi Qiu, Boyuan Chen, Jiayi Zhou, Changye Li, Hantao Lou, Josef Dai, Yunhuai Liu, and Yaodong Yang. Language models resist alignment: Evidence from data compression. *arXiv preprint arXiv:2406.06144*, 2024.

[49] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[50] Andrew Wang, Sophia Hager, Adi Asija, Daniel Khashabi, and Nicholas Andrews. Hell or high water: Evaluating agentic recovery from external failures, 2025. URL https://arxiv.org/abs/2508.11027.

[51] Sungwon Kim and Daniel Khashabi. Challenging the evaluator: Llm sycophancy under user rebuttal, 2025. URL https://arxiv.org/abs/2509.16533.

[52] Aman Madaan, Niket Tandon, Prakhar Gupta, Gabriel Ilharco, Siddharth Singh, Tushar Khot, Hannaneh Hajishirzi, Wen-tau Yih, and Yih Tau. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

[53] Shehzaad Dhuliawala, Tushar Khot, Ashish Sabharwal, and Peter Clark. Chain of verification: How large language models perform reasoning with external tools. *arXiv preprint arXiv:2305.00053*, 2024.

[54] Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. Large language models can self-correct with key condition verification, 2024. URL https://arxiv.org/abs/2405.14092.

[55] Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation, 2023. URL https://arxiv.org/abs/2307.03987.

[56] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing, 2024. URL https://arxiv.org/abs/2305.11738.

[57] Shuyang Jiang, Yuhao Wang, and Yu Wang. Selfevolve: A code evolution framework via large language models, 2023. URL https://arxiv.org/abs/2306.02907.

[58] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug, 2023. URL https://arxiv.org/abs/2304.05128.

[59] Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5823–5840, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.320. URL https://aclanthology.org/2023.acl-long.320/.

[60] Wenhao Yu, Zhihan Zhang, Zhenwen Liang, Meng Jiang, and Ashish Sabharwal. Improving language models via plug-and-play retrieval feedback, 2023. URL https://arxiv.org/abs/2305.14002.

[61] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks, 2023. URL https://arxiv.org/abs/2303.17491.

[62] Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms, 2024. URL https://arxiv.org/abs/2406.01297.

[63] Kevin Wu, Eric Wu, and James Zou. Clasheval: Quantifying the tug-of-war between an llm's internal prior and external evidence, 2024. URL https://arxiv.org/abs/2404.10198.

[64] Jianhao Yan, Yun Luo, and Yue Zhang. Refutebench: Evaluating refuting instruction-following for large language models, 2024. URL https://arxiv.org/abs/2402.13463.

[65] Sophia Hager, David Mueller, Kevin Duh, and Nicholas Andrews. Uncertainty distillation: Teaching language models to express semantic confidence, 2025. URL https://arxiv.org/abs/2503.14749.

[66] Rongwu Xu, Brian S. Lin, Shujian Yang, Tianqi Zhang, Weiyan Shi, Tianwei Zhang, Zhixuan Fang, Wei Xu, and Han Qiu. The earth is flat because...: Investigating llms' belief towards misinformation via persuasive conversation, 2024. URL https://arxiv.org/abs/2312.09085.

[67] Boshi Wang, Xiang Yue, and Huan Sun. Can chatgpt defend its belief in truth? evaluating llm reasoning via debate, 2023. URL https://arxiv.org/abs/2305.13160.

[68] Yuqing Yang and Robin Jia. When do llms admit their mistakes? understanding the role of model belief in retraction, 2025. URL https://arxiv.org/abs/2505.16170.

[69] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023. URL https://arxiv.org/abs/2202.05262.

[70] Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations, 2025.

[71] Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R. Tomz, Christopher D. Manning, and Weiyan Shi. Verbalized sampling: How to mitigate mode collapse and unlock llm diversity, 2025.

[72] Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Retroformer: Retrospective large language agents with policy gradient optimization, 2024. URL https://arxiv.org/abs/2308.02151.

 https://doi.org/10.52202/085713-1831

[73] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, 2023. URL `https://arxiv.org/abs/2306.05685`.

A Examples of different feedback types

We provide a complete worked example illustrating the feedback types differences in our study in Figure 8. The problem (AIME 2024-I-10) involves finding a segment length in a circle with tangent lines. The solver incorrectly assumes the triangle is right-angled and applies an unjustified formula.

F2 identifies two specific formula errors: using cos A instead of sin A, and using angle (A+C) instead of B in the Law of Cosines. F3 provides six numbered issues with extensive explanations and a step-by-step correction roadmap.

Original Question:
Let ABC be a triangle inscribed in circle ω. Let the tangents to ω at B and C intersect at point D, and let AD intersect ω at P. If AB=5, BC=9, and AC=10, AP can be written as m/n, where m and n are relatively prime integers. Find m + n.

Solver Model's Attempt:
Step 1: We need to find the length of AP, where AD intersects ω at P. We should utilize properties of inscribed angles, tangents to circles, and similar triangles.
Step 2: Recall that the tangent to a circle is perpendicular to the radius drawn to the point of tangency. By using the properties of tangents, we can relate the lengths of segments BD, CD, and AD.
Step 3: Apply the power of a point theorem. For point D, we have DB · DC = DA · DP.
Step 4: To find DB and DC, use the fact that DB and DC are tangents to the circle and BC is a chord. Let x = BD = CD and use the given side lengths AB = 5, BC = 9, and AC = 10.
Step 5: Given AB = 5, BC = 9, and AC = 10, we notice that $5^2 + 9^2 = 10^2$, which means △ABC is a right-angled triangle with ∠BCA = 90°.
Step 6: With △ABC being a right triangle, we can find the radius R of circle ω using R = abc/(4Δ), where Δ is the area. The area Δ = (1/2) · 5 · 9 = 45/2, and R = (5 · 9 · 10)/(4 · 45/2) = 5.
Step 7: Since the triangle is right-angled at C, we focus on using BC, AC, and AB to find AP through areas and similar triangles.
Step 8: By Power of a Point from D, DB · DC = DA · DP. Also, note that DA = DP + AP.
Step 9: We can use the fact that △ABD and △ACD are similar to △PBD and △PCD respectively, or directly apply Power of a Point and similarity to find AP.
Step 10: For a right-angled triangle inscribed in a circle, the mean proportional theorem can be applied. Using AB · AC = AD · AP or similar relations, we compute AP.
Step 11: Direct computation of AP using AB = 5, BC = 9, AC = 10. From the relation, AP = (AB · AC)/(AB + AC) = (5 · 10)/(5 + 10) = 50/15 = 10/3.
Step 12: To get the sum m + n where AP = m/n, we have m = 10 and n = 3, so m + n = 10 + 3 = 13.

The final answer is: 13

Binary Correctness Feedback (F1):
The answer is wrong!

Self-Generated Reflective Feedback (F2) Using the Solver Model Itself:
Here are the critical errors in your reasoning:

1 **Expression for CD**
You set CD = (BC/2) / cos A. In fact, in isosceles ΔBDC (with base BC and base-angles A), BC = 2·CD·sin A, so CD = BC/(2 sin A), not BC/(2 cos A).
2 **Law of Cosines on ΔACD**
You wrote $AD^2 = AC^2 + CD^2 - 2\cdot AC\cdot CD\cdot\cos(A + C)$. But ∠ACD is not A + C. By the alternate-segment theorem (or inscribed-angle chase) one finds ∠ACD = ∠ABC = B.
Hence the correct relation is $AD^2 = AC^2 + CD^2 - 2\cdot AC\cdot CD\cdot\cos B$.

These two mis-steps propagate through your arithmetic for AD, DP and then AP. Once you replace BC/(2 cos A) with BC/(2 sin A) to find CD, and use ∠ACD = B in the LoC for AD, the subsequent power-of-a-point calculation DP = CD^2/AD and then AP = |DP − AD| will fall into line and yield the true rational value of AP.

Strong-Model Reflective Feedback (F3) Using GPT-4.1:
Thank you for sharing the solution and your reasoning! Here's some feedback identifying the issues in the previous approach and guidance on the steps to correct:
1 **Step 5 and 6 (Right triangle and radius):** The claim that the triangle is right angled because $5^2 + 9^2 = 10^2$ is actually incorrect in this problem. Since $5^2 + 9^2 = 25 + 81 = 106$ which is not equal to $10^2 = 100$, the triangle is **not** right angled. This means the calculations that follow assuming a right angle, including the radius computation based on right triangle properties, cannot be used.
2. **Step 7 (Using incorrect triangle configuration):** The reasoning in Step 7 mistakenly treats the triangle as right angled and misapplies the Pythagorean theorem to find lengths related to points B,D,C. So, lengths BD, DC, etc., cannot be simplified as done there.
3. **Step 11 (Formula for AP):** The step that writes AP = (AB · AC)/(AB + AC) is not a known or justified direct formula and comes without proof or reference. Without the triangle being right angled or a direct similarity/proportionality argument, such a formula does not hold here.
4. **Ignoring the angle relationships from tangent lines:** The most crucial fact is that the intersection point D of the tangents at B and C satisfies ∠BDC = ∠BAC (the tangents form an angle equal to the arc they subtend). This fact is a cornerstone to finding exact expressions for lengths AD, CD, and ultimately AP.
5. **Missing Law of Cosines computations to find AD and CD:** To correctly solve the problem, use the Law of Cosines in △ACD and △BCD, incorporating the angle relationships from the tangents and arcs to express AD and CD in terms of known sides and cosines of angles like ∠A and ∠B.
6. **Power of a Point at D:** Applying the power of a point theorem at D is key, but DP × AD = DB × DC. Since DB = DC (both are tangent lengths from point D to the circle), this relation helps you solve for DP. Then AP = AD - DP.

These corrections will lead to the proper final expression for AP.

Figure 8: Examples of different feedback types.

https://doi.org/10.52202/085713-1831

B Prompts and evaluation details for problem solving and feedback generation

B.1 Prompts used for the solver model

We developed different prompting strategies for the solver model to incorporate feedback and generate new solutions across iterations. The system prompts in Figure 9 were used for the solver model across different tasks:

```
Math-500 and AIME 2024:
You are a smart assistant that solves math problems. Please think step by step to solve the problem. If you think you're ready
to output the answer, you can wrap your answer with \\boxed{}. Please follow this format

TriviaQA:
You are a smart assistant that solves trivia questions. If you think you're ready to output the answer, you can just output an
answer.

MMLU and MMLU Pro:
The following are multiple-choice questions about {category}. Let's think step by step. Please explain your reasoning clearly as
you work toward the answer. When you're ready, conclude your answer with the phrase: \"The answer is (X)\" where X is the
correct letter choice. Make sure to always include parentheses around the letter.

GPQA:
The following are multiple-choice questions. Let's think step by step. Please explain your reasoning clearly as you work toward
the answer. When you're ready, please finish your answer with \"The answer is (X)\" where X is the correct letter choice. Make
sure to always include parentheses around the letter.

PopQA:
You are a smart assistant that answers fact-based questions. If you think you're ready to output the answer, you can just output
an answer.

5-Digit Multiplication:
You are a smart assistant in solving multiplication questions. Please think step by step. If you think you're ready to output
the answer, you can wrap your answer with \\boxed{}. Please follow this format.

Hexadecimal 5-Digit Multiplication:
You are a smart assistant in solving hexadecimal multiplication questions. Please think step by step. If you think you're ready
to output the answer, you can wrap your answer with \\boxed{}. Please follow this format.
```

Figure 9: System prompts used for the solver model across all tasks.

At the initial generation round, we directly use the question as the prompt for the solver model. For multiple-choice questions, we format the question by concatenating the question and answer choices with their corresponding labels (i.e., A to D for MMLU and GPQA, A to J for MMLU Pro). In subsequent rounds, we provide the solver model with its complete previous history and the corresponding feedback from the feedback generator model. We clearly label each iteration so the solver model can track all its previous attempts. The general template for the iterative prompt structure is provided in Figure 10.

```
Question:
[Original Question]

Prompt for Generation after the Initial Round:
You are given the full history of your previous attempts and the feedback provided for each attempt.
History:

Attempt at (iteration 1) and the corresponding feedback:

Answer: [answer1] Feedback: [Feedback 1 Depends on Feedback Strategy].

Attempt at (iteration 2) and the corresponding feedback:

Answer: [answer2] Feedback: [Feedback 2 Depends on Feedback Strategy].

Question: [Original Question]. Please answer the question again.


* [answer1] and [answer2] corresponds to the model's first two failed attempts. [Original Question] refers to the initial question the
model attempts to answer.
```

Figure 10: Prompt used for iterative self-improvement

B.2 Evaluation details for problem solving

We employ few-shot prompting across all tasks to provide consistent context for the solver model. For TriviaQA and PopQA, we randomly sample 5 questions without replacement as few-shot examples. For MMLU and MMLU Pro, we similarly sample 5 questions from the corresponding question category to ensure domain-relevant examples.

For PopQA, we employ an LLM-as-a-judge approach [73] to assess answer correctness. This is necessary because PopQA provides limited answer aliases (extensive alternative phrasings for exact string matching) compared to TriviaQA. Without this approach, models would be penalized for minor formatting differences rather than genuine comprehension errors, leading to an underestimation of their true problem-solving capabilities. For other tasks, we follow the same evaluation metrics provided by lm-eval-harness.

B.3 Prompts used for feedback generation

We implement three distinct feedback generation strategies as described in §2.2. For Binary Correctness Feedback (F_1), we provide minimal information: "Your answer was incorrect. Please answer the question again."

For Self-Generated Reflective Feedback (F_2) and Strong-Model Reflective Feedback (F_3), we employ identical prompt templates that differ only in the model used for generation. The feedback generator receives the complete interaction history, including all previous solver attempts and corresponding feedback. When available, we provide the feedback model with detailed solution explanations that justify the correct answer; for datasets lacking such explanations, we provide only the ground truth answer. This approach ensures the feedback model has sufficient context to generate targeted, informative guidance while maintaining consistency across feedback types, and its template is shown in Figure 11.

```
Prompt for Generating the Feedback:
There was a mistake in answering the following question:

[Original Question]

You are provided with the full history of previous attempts made by a separate model, along with corresponding feedback.
History:
Attempt at (iteration 1) and the corresponding feedback:

Answer: [answer1] Feedback: [Feedback 1 Depends on Feedback Strategy].

Attempt at (iteration 2) and the corresponding feedback:

Answer: [answer2] Feedback: [Feedback 2 Depends on Feedback Strategy].

Most Recent Answer: [answer3]

The correct final answer is: [Ground Truth Answer]
The correct reasoning process that leads to this answer is: [Solution with Process for this Question]

Please provide feedback identifying which step(s) were incorrect or how to get to the correct answer WITHOUT revealing
the correct final answer or the content of the correct option.

* [answer1] [answer2] [answer3] refer to all previous attempts the model had. [answer3] is the most recent attempt.
[Ground Truth Answer] is the answer for this question. [Solution with Process for this Question] is the detailed solution
provided for the question in the dataset.
```

Figure 11: Prompt used for generating the feedback.

B.4 Answer masking in feedback

To ensure fair evaluation, we implement comprehensive answer masking to prevent feedback from directly revealing ground truth solutions while preserving feedback quality. Our approach allows feedback to contain detailed solution steps and guidance but strictly prohibits explicit disclosure of final answers. We use "[masked]" as the replacement token for filtered content.

Multiple-choice questions. We mask all possible representations of the correct choice letter. For example, if the correct answer is A, we filter variants including (A), `\boxed{A}`, **A**, etc.

Open-ended questions. For TriviaQA, we filter all terms matching the words in "aliases" and "normalized aliases" answer fields. For PopQA, we mask entries from the "possible answers" answer field. For mathematical tasks (5-digit multiplication and MATH-500), we mask standalone numerical answers and those in `\boxed{}` notation. Hexadecimal multiplication follows similar patterns. For multiplication tasks, we additionally mask intermediate partial products to prevent reduction to simple addition problems (detailed in Appendix D).

 https://doi.org/10.52202/085713-1831

B.5 Error Categorization Prompt

The prompt template used for categorizing persistent model errors after 9 iterations is shown in Figure 12.

```
System Prompt:
You are an error categorizer specialized in analyzing why Language Learning Models (LLMs) "
fail to self-improve when solving problems. When provided with an LLM's prediction trajectory and the feedback it receives,
you will categorize the errors into one of six categories:
1. Problem is Impossible to Solve
   - The problem itself is fundamentally flawed
   - External tools are required (e.g., calculator for complex calculations, search engine for obscure facts)
2. Problem is Too Complicated
   - The problem exceeds the model's knowledge scope
   - Example: A level 5 math problem beyond the model's training
3. Feedback is Wrong
   - The feedback generator model provides incorrect guidance
   - The feedback fails to identify actual mistakes in the model's response
   - The feedback is too vague or generic to be helpful
4. Model is Not Following Feedback
   - The model fails to incorporate feedback properly
5. Style or Formalization Issue
   - Logical correctness but formatting or notation problems
6. Unknown
   - Cannot categorize the failure into above categories
End your response with exactly: 'The error is: [category]' where category is one of:\n- impossible to solve\n- too
complicated\n- feedback is wrong\n- model is not following the feedback\n- style or formalization issue\n- unknown

User Prompt:
The question and the interaction between the agent model and the feedback model is: {question}
The process answer is: {process_answer}
Please categorize the error. End your response with:\n'The error is: [category]

* {question} is the first 5 iterations of the answer-feedback pairs; {process_answer} is the best ground truth solution we can
provide from the dataset to the model.
```

Figure 12: Error Categorization

C Error categorization examples from iterative self-improvement

This section presents representative examples of persistent errors that prevent models from achieving correct solutions despite multiple feedback iterations. We illustrate the main error categories identified in our analysis: feedback resistance (where models fail to incorporate valid corrections, see Figure 13) and feedback quality issues (where the provided guidance is incorrect or misleading, see Figure 14).

```
Question:
Which crop began the Third Agricultural Revolution?
Choices:
(A) Rice
(B) Wheat
(C) Barley
(D) Millet

Model Answer:
The Third Agricultural Revolution, also known as the Green Revolution, involved the development and introduction of high-yielding
varieties of wheat, as well as other crops like maize and rice. However, among the given options, wheat is specifically notable
for being one of the initial crops that sparked this revolution, particularly with the work of Norman Borlaug.The answer is (B)

Feedback:
The reasoning in the answer focused correctly on the importance of a certain crop during the Third Agricultural Revolution and
mentioned high-yielding varieties and Norman Borlaug's work. However, the key misstep was in associating that revolutionary
agricultural development specifically with one crop among the options without fully considering the initial crop that triggered
the Green Revolution. While multiple crops were involved later, the revolution is generally attributed to breakthroughs related
to a specific cereal that was first targeted for increasing yields in the early stages.To improve the answer, it would be
important to pinpoint which crop was actually the catalyst for the Green Revolution by reviewing the historical timeline and the
initial crop that demonstrated dramatic yield improvements through the introduction of new agricultural techniques and varieties.
This would help identify the earliest crop connected to sparking the Third Agricultural Revolution rather than focusing solely on
one widely known crop associated with it later.

Model Revised Answer:
The Third Agricultural Revolution, also known as the Green Revolution, is often attributed to the development of high-yielding
varieties of crops. Among the given options, wheat is specifically notable for being one of the initial crops that sparked this
revolution, particularly with the work of Norman Borlaug. His development of high-yielding, disease-resistant wheat varieties led
to significant increases in food production and is considered a catalyst for the Green Revolution.The answer is (B)
```

Figure 13: Llama-4-Scout resisting feedback from GPT-4.1 mini in MMLU

Question:
A homeowner entered into a contract with an insurance company to purchase a fire insurance policy for her new home. The policy provided that the insurance company promised to pay up to $500,000 if the said house was destroyed by fire or fire-related explosion, while the homeowner promised to pay a quarterly premium of $400. The homeowner failed to make the last two quarterly payments before her house burned to the ground. The insurance company refused to pay the homeowner for the loss because of her failure to make the last premium payments. In an action by the homeowner against the insurance company to recover for the loss of her house, the homeowner's best theory of recovery is that
Choices:
(A) although she failed to make the last two premium payments, there was a bargained-for exchange.
(B) the insurance company's duty to pay was not expressly conditioned on the homeowner's duty to make the payments.
(C) the insurance company was under an independent duty to pay for the loss.
(D) the homeowner did not receive notice of cancellation.

Model Answer:
The answer is (D)

Feedback:
The primary error in the previous reasoning was focusing on the homeowner's lack of notice of cancellation rather than examining the nature of the contractual obligations and the conditional duties within the insurance policy.\n\nIn analyzing this scenario, it's important to understand that insurance contracts often contain conditions precedent\u2014the insurer's duty to pay is typically contingent upon the insured's fulfillment of certain obligations, such as timely payment of premiums.
The mistake was not sufficiently weighing how failure to pay premiums affects the insurer's duty to perform under the contract.
The key is to assess whether the insurance company's obligation is dependent upon the homeowner's premium payment, and if that obligation exists independently or is conditioned on such payment.
Therefore, the analysis should have involved a closer look at the contractual terms and the interdependence of duties, rather than focusing chiefly on issues related to cancellation notices.

O4-mini's judgement:
The feedback repeatedly mischaracterized the contract\u2019s structure by insisting the insurer\u2019s duty was a condition precedent on timely premium payments, thereby directing the model away from the correct theory that the insurer owed an independent duty to pay. In other words, the guidance itself was incorrect and prevented the model from reaching the right answer. The error is: feedback is wrong

Figure 14: Wrong feedback provided by GPT-4.1 mini judged by o4-mini

D Synthetic digit multiplication task details

D.1 5-digit multiplication

We construct a controlled arithmetic dataset consisting of 450 5-digit multiplication problems following the template: ‘‘`Calculate the following question: 19365` $\times$ `12534.`’’

Feedback generation We employ a deterministic, human-designed template based on the distributive property to generate ground truth solutions. This template systematically decomposes each multiplication into partial products, providing a clear step-by-step solution pathway. Our template-based approach serves two key purposes: (1) demonstrating structured problem decomposition strategies for complex arithmetic, and (2) ensuring feedback correctness and interoperability. In Figure 15, we illustrate an example template solution, which serves as the reference for feedback generation. The feedback model compares solver outputs against this structured breakdown to identify specific computational errors.

Answer masking strategy To maintain task difficulty, we also mask intermediate partial answers before providing feedback to the solver model. The final feedback combines the masked template solution with model-generated guidance tailored to the specific errors observed.

Question:
Calculate the following question: 32183 * 40672.

Manually Generated Feedback:
The original question is:
Calculate the following question: 19365 * 12534.
Step1: After breaking down the numbers: 19365 = 19000 + 365, 12534 = 12000 + 534
Stwp2: Then we apply the distributive property:
19365 x 12534 = (19000 + 365) x (12000 + 534)
Compute partial products:
Step3: 19000 x 12000 = 228000000
Step4: 19000 x 534 = 10146000
Step5: 365 x 12000 = 4380000
Step6: 365 x 534 = 194910
Step7: Sum all partial products we have the answer: 242579910
Total: 228000000 + 10146000 + 4380000 + 194910 = 242579910

Figure 15: Templates for 5-digit multiplication solution.

D.2 Hexadecimal 5-Digit Multiplication

We also extend our synthetic arithmetic evaluation to hexadecimal multiplication, creating problems that challenge models' ability to work with non-standard number systems. The question tem-

https://doi.org/10.52202/085713-1831

```
Question:
Calculate the following question, where each number is represented in base 16: 69837 * 17635.

Manually Generated Feedback:
Multiplying 69837 by 17635 in base 16:
Step 1: Multiply 69837 by '5' \u2192 Partial product: 20F913, Shifted: 20F913, Intermediate Sum: 20F913
Step 2: Multiply 69837 by '3' \u2192 Partial product: 13C8A5, Shifted: 13C8A50, Intermediate Sum: 15D8363
Step 3: Multiply 69837 by '6' \u2192 Partial product: 27914A, Shifted: 27914A00, Intermediate Sum: 28EECD63
Step 4: Multiply 69837 by '7' \u2192 Partial product: 2E2981, Shifted: 2E2981000, Intermediate Sum: 30B86DD63
Step 5: Multiply 69837 by '1' \u2192 Partial product: 69837, Shifted: 698370000, Intermediate Sum: 9A3BDDD63
Final Product (hex) = 9A3BDDD63
```

Figure 16: Hexadecimal 5-Digit Multiplication process solution.

plate follows the format: ‘‘`Calculate the following question, where each number is represented in base 16: 69837 × 17635.`’’ All answers are expected in base-16 format.

Template-based feedback. Similar to decimal multiplication, we generate feedback using deterministic step-by-step templates. The multiplication process involves sequentially multiplying the first operand by each digit of the second operand (interpreting digits in base-16), then summing the resulting partial products with appropriate positional shifts. Figure 16 demonstrates an example template solution. We validate solution correctness by verifying that partial product summation matches results from standard base-16 calculators.

Masking strategy for hexadecimal multiplication. Given the increased computational complexity of hexadecimal arithmetic, we employ a more permissive masking approach. We mask only the final summation step while preserving intermediate partial products. This design balances providing sufficient guidance with maintaining the core challenge of hexadecimal computation.

E Analysis of model confidence and FEEDBACK FRICTION

Figure 17 presents confidence-accuracy relationships across four datasets: GPQA, MMLU, MMLU Pro, and TriviaQA. Each plot displays three metrics: initial accuracy at iteration 0, final accuracy after iterative feedback, and the improvement delta between them.

We define a model's *initial confidence* in its generated answer as the exponential of the average log-probability per token in the answer sequence:

$$\text{Initial Confidence} = \exp\left(\frac{1}{T}\sum_{t=1}^{T}\log p(a_t \mid a_{<t}, q)\right)$$

where:

- T is the number of tokens in the generated answer (with EOS excluded),
- a_t is the t-th token in the answer,
- $a_{<t}$ denotes the prefix of the answer up to (but not including) position t,
- q is the input question,
- $p(a_t \mid a_{<t}, q)$ is the model's probability of generating token a_t given the previous tokens and the input.

This corresponds to the geometric mean of the per-token probabilities assigned by the model, and serves as a quantitative measure of the model's confidence in its complete answer.

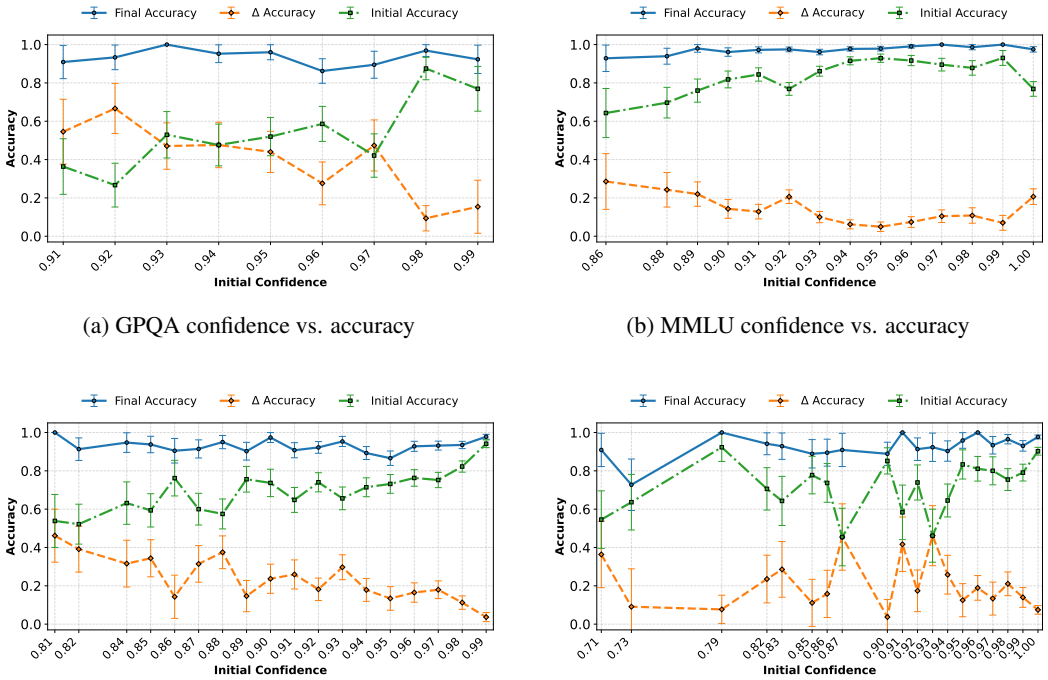
We find that **initial confidence strongly predicts initial accuracy** across all datasets. Higher confidence bins consistently correspond to higher initial performance (Figures 17a–17d), confirming that the model's confidence is a good predictor of its initial accuracy.

However, **confidence poorly predicts improvement potential**. The relationship between initial confidence and accuracy gains varies substantially:

- GPQA shows peak improvements at moderate confidence levels, with diminishing returns at higher confidence

- MMLU and MMLU Pro exhibit relatively flat improvement patterns with the less confident questions getting more improvements. Nevertheless, this may caused by the low initial accuracy.

- TriviaQA demonstrates erratic improvement fluctuations regardless of initial confidence



(a) GPQA confidence vs. accuracy

(b) MMLU confidence vs. accuracy

(c) MMLU Pro confidence vs. accuracy

(d) TriviaQA confidence vs. accuracy

Figure 17: Confidence vs. accuracy across different datasets using GPT-4.1 mini as feedback model and Llama-4-Scout as the solver.

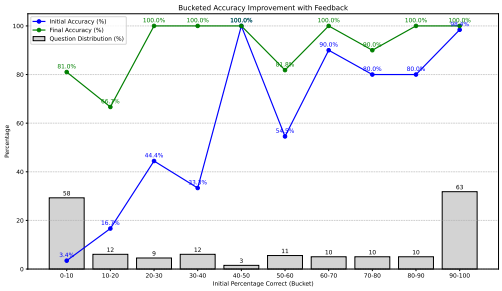
F Analysis of data familiarity and RIGID THINKING

After analyzing data familiarity using answer frequency in the PopQA dataset, we found no clear correlation between model performance and frequency of answer words in the training data. However, surface-level frequency may not fully capture a model's true familiarity with content, as it fails to account for context quality, semantic relationships, and other factors affecting knowledge acquisition during pre-training.

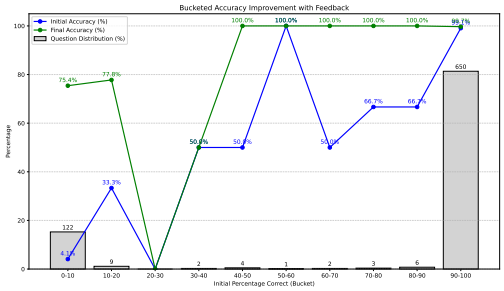
To better capture actual familiarity, we examine a more direct behavioral signal called in-domain performance: the model's accuracy in answering questions, measured using 100 generations per question with Llama-3.3. This behavioral familiarity metric reflects the cumulative effect of all factors contributing to the model's internalized knowledge.

Figure 18 illustrates the in-domain performance of Llama-3.3 across four benchmarks—GPQA, TriviaQA, 5-digit multiplication, and MMLU Pro. We bucket questions based on the model's initial accuracy and report both initial and final accuracy after iterative feedback. While the model shows improvement across all buckets, questions with higher behavioral familiarity (higher initial accuracy) consistently achieve higher final accuracy as well. This suggests that behavioral familiarity is a more informative predictor of both current performance and improvement potential than answer frequency alone. Nevertheless, we still cannot obverse any consistent patterns across all these datasets in the initial vs. final accuracy.

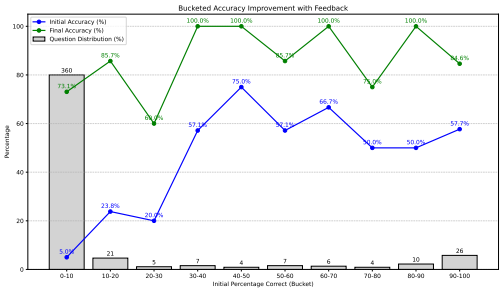
https://doi.org/10.52202/085713-1831



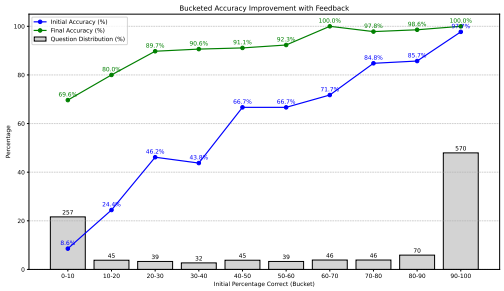
(a) GPQA in-domain performance



(b) TriviaQA in-domain performance



(c) 5-digit multiplication in-domain



(d) MMLU Pro in-domain performance

Figure 18: In-domain accuracy of Llama-3.3 across four benchmark tasks.

G Analysis of reasoning complexity and FEEDBACK FRICTION

To investigate whether the model's improvement over iterations correlates with question difficulty or the reasoning complexity, we compare the performance of Llama-4-Scout on two synthesized multiplication tasks: 5-digit and 6-digit problems. Unlike prior datasets, which lack clear separability in difficulty levels, these tasks were manually constructed with 450 questions each to ensure a well-defined difference in complexity.

The initial accuracy of Llama-4-Scout is 2.2% on 5-digit multiplication and 0.889% on 6-digit multiplication. Interestingly, while the 6-digit task is objectively more difficult, we observe greater improvement across iterations compared to the 5-digit task. One possible explanation is that more difficult tasks offer more room for feedback-driven correction because solver model has less initial knowledge about how to solve them. However, we also observe cases where simpler questions yield higher final accuracy, suggesting that the relationship between task complexity and feedback effectiveness is non-monotonic and influenced by additional factors.

H Analysis of model type and FEEDBACK FRICTION

To better understand the overlap in failure cases among Llama-3.3, Llama-4-Scout, and Llama-4-Maverick, we compared their incorrect predictions across several benchmark datasets. Specifically, we report the number of shared mistakes between each pair of models, the number of questions all three models got wrong, the total number of unique mistakes (union), and the **Overlap Ratio**, defined as the proportion of all-three common errors to the total number of distinct errors.

The *Overlap Ratio* offers a normalized measure of agreement in model failures. As shown in Table 2, **AIME** shows the highest overlap (35.7%), suggesting a subset of examples that all three models consistently struggle with. Conversely, datasets such as **GPQA** and **5-digit Multiplication** exhibit minimal overlap (6.9% and 0.7%, respectively), indicating that the models tend to fail on different questions.

These findings suggest that model failures are often idiosyncratic rather than being concentrated around a universally difficult subset of examples. The relatively low overlap across datasets highlights the challenge of achieving robust self-correction: errors are not easily attributable to a common set of pitfalls, but rather reflect distinct weaknesses in each model's reasoning and generalization.

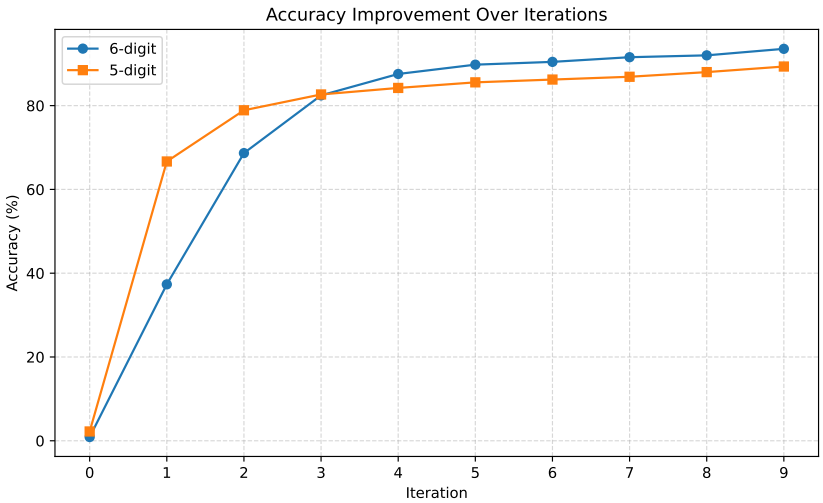


Figure 19: Comparison of the Improvement for 5-digit and 6-digit multiplication with GPT-4.1 mini as feedback model

Dataset	L3.3–Scout	L3.3–Maverick	Scout–Maverick	All-Three	Union	Overlap Ratio
AIME	8	9	5	5	14	0.357
TriviaQA	22	16	28	14	105	0.133
MATH-500	18	14	11	9	64	0.141
MMLU	15	12	19	11	55	0.200
MMLU Pro	49	40	43	30	163	0.184
GPQA	3	5	2	2	29	0.069
5-digit Mult.	21	3	1	1	141	0.007

Table 2: Pairwise and three-way common failure cases among Llama-3.3, Llama-4-Scout, and Llama-4-Maverick across datasets. Overlap Ratio is computed as the number of questions all three models failed on divided by the union of all distinct failures.

I Details of testing model's self-perception of FEEDBACK FRICTION

Figure 20 shows the prompts we used to probe whether the model understand the given feedback and whether it's willing to change its belief.

```
Feedback Understanding Prompt:
Question: {original_question}
Your previous answer: {previous_response}
Feedback received: '{previous_feedback}'
Do you understand this feedback? Please answer with 'Yes' or 'No' and briefly explain why.


Belief Update Prompt:
Question: {original_question}
Your previous answer: {previous_response}
Feedback received: '{previous_feedback}'
Have you updated your belief or understanding about this problem? Please answer with 'Yes' or 'No' and briefly explain what
changed.
```

Figure 20: An example of Llama-4-Scout explicitly acknowledging changing its belief while still making the same mistake in the next attempt.

Figure 21 shows an example where Llama-4-Scout explicitly acknowledges that "the actual boundary is defined by the perpendicular bisectors of the segments connecting O to the vertices, which can result in an oblique line" and claims its approach has been updated. Yet in its second attempt, it still describes the region as forming "a square with side length 1/2 min(AB, AD)" and arrives at the same incorrect answer of 1/4. This demonstrates the model can articulate understanding of the feedback while its actual problem-solving behavior remains unchanged. The correct answer is 1/2.

https://doi.org/10.52202/085713-1831

Original Question:
Rectangle ABCD has center O and AB/AD=k. A point is randomly chosen from the interior of rectangle ABCD. What is the probability that it is closer to O than to any of the four vertices?

Model's First Attempt:
The region closer to O than to any vertex is a smaller rectangle with side lengths 1/2 AB and 1/2 AD. This smaller rectangle has an area of 1/2 AB × 1/2 AD = 1/4 AB × AD. The area of the entire rectangle ABCD is AB × AD. Thus, the probability is 1/4 AB × AD / AB × AD = 1/4. The final answer is: 1/4

Feedback Provided:
The original solution assumes that the region closer to the center O than to any of the vertices forms a smaller rectangle aligned with the original rectangle, with side lengths exactly half of the original. However, the locus of points closer to O than to a particular vertex is not a straight boundary parallel to the sides; rather, it is defined by the perpendicular bisector of the segment connecting O and that vertex, which is generally an oblique line. The assumption that the smaller region is exactly one-quarter of the whole area because it is a rectangle of half the original sides is not correct.

Model's Statement on Belief Update:
Yes. My understanding of the problem has changed based on the feedback provided. Initially, I incorrectly assumed the region closer to the center O than to any vertex formed a smaller rectangle with sides exactly half the lengths of the original rectangle. However, the feedback highlighted that this assumption is incorrect and that the actual boundary is defined by the perpendicular bisectors of the segments connecting O to the vertices, which can result in an oblique line. This realization, combined with the importance of considering symmetry, transformations, and a more nuanced geometric partitioning of the rectangle, has updated my approach to solving the problem.

Model's Second Attempt:
By drawing these perpendicular bisectors, we can see that the region closer to O than to any vertex in this smaller rectangle forms a square with side length 1/2 min(AB, AD). The area of this smaller rectangle is 1/4 AB × AD, and the area of the region closer to O is 1/2 min(AB, AD) × 1/2 min(AB, AD). However, considering the setup and focusing on the probability aspect, we should approach it by understanding that the division of the rectangle into regions closer to O or the vertices is based on the perpendicular bisectors, which effectively creates a smaller square (when considering the closest region to O) within each of the four smaller rectangles. Given the symmetry and the nature of the division, the probability simplifies to the ratio of the area where points are closer to O than to any vertex, over the total area of the rectangle. This simplification leads to recognizing that the probability does not depend on the aspect ratio k but rather on the geometric properties of the rectangle and its division. The final answer is: 1/4

Figure 21: An example of Llama-4-Scout explicitly acknowledging changing its belief while still making the same mistake in the next attempt.

J Full results of semantic entropy

Figure 22 presents the complete self-correction analysis across all five benchmark tasks (5 digits multiplication, GPQA, MATH, MMLU-Pro, and TriviaQA) for both Llama-3.3-70B and Llama-4-Scout. Each subplot shows initial accuracy (blue), final accuracy (orange), and absolute improvement rate (green) as functions of semantic entropy with bucket size 0.2. Regions with fewer than 10 samples are marked with red shading and dashed boundaries to indicate unreliable data, with sample counts displayed in boxes above each plot.

Figure 22: Full results of semantic entropy over five benchmark tasks and two models.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction (pages 1-2) clearly state the paper's main claims, including the identification of a phenomenon called FEEDBACK FRICTION where LLMs show resistance to incorporating external feedback despite multiple iterations. The claims match the experimental results shown in the paper, particularly in Figures 1 and 3, which demonstrate models consistently plateauing below their theoretical accuracy ceiling. The limitations are also mentioned, as they note that even with their best strategies, models still fail to reach target accuracy.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations of their work in several places. The authors acknowledge that their sampling strategies only mitigate but do not eliminate FEEDBACK FRICTION (p.7, lines 249-252). They also note that despite investigating various hypotheses, they couldn't fully explain the causes of model stubbornness (p.8, lines 284-284). The synthetic tasks section (p.4) acknowledges limitations in task selection by explaining their choice of objective tasks over subjective ones to ensure reliable evaluation.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

https://doi.org/10.52202/085713-1831

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper is primarily empirical and does not include formal theoretical results or proofs that would require mathematical verification. While they develop the concept of FEEDBACK FRICTION, this is demonstrated through experimental evidence rather than mathematical theorems.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient detail to reproduce the main experimental results. Section 3.1 (p.4-5) describes the experimental setup, including tasks, metrics, models, and inference settings. The authors explain their continual self-improvement framework (p.3), feedback mechanisms (p.3-4), and sampling strategies (p.7). They specify using temperature 0 for deterministic outputs, vllm for inference, and GPT-4.1 mini for generating strong-model feedback.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The author mentioned they will share the code and data used for this paper upon paper acceptance, and they've shared an anonymized repo.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the experimental details needed to understand the results. Section 3.1 (p.4-5) describes the tasks used (AIME 2024, MATH-500, TriviaQA, PopQA, MMLU, MMLU Pro, GPQA, and two synthetic tasks), the models tested (LLaMA-3.3 70B Instruct, Llama-4-Scout-17B-16E, Llama-4-Maverick-17B-128E-Instruct-FP8), and inference settings (temperature 0, vllm, chat templates). They also explain how they sample 10% of data for certain datasets and why this is representative.

https://doi.org/10.52202/085713-1831

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports error bars in the analysis section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides information about the compute resources used in the experimental setup section, where they mentioned all their experiments are done on a single H100 with 8 GPUs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research in this paper appears to conform with the NeurIPS Code of Ethics. The paper studies LLMs' ability to incorporate feedback, which is foundational research without apparent ethical concerns. The authors use publicly available models and benchmark datasets, and their experimental methodology doesn't involve deception, harmful content generation, or human subjects that would raise ethical issues.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses the broader societal impacts of the work in the introduction and conclusion.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper does not release new models, datasets, or tools that would pose risks for misuse. The research uses existing LLMs and benchmark datasets to study their feedback incorporation capabilities. The authors don't create or release content that would require safeguards against misuse.

- The answer NA means that the paper poses no such risks.

https://doi.org/10.52202/085713-1831

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: The paper references the original sources of the models (such as Meta AI for Llama models and OpenAI for GPT-4.1 mini). Since these models and the benchmark have permissive licenses and terms of use that easily allow for third-party implementation, explicit mention of licensing details is unnecessary.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces two synthetic digit multiplication tasks, and details will be provided in the appendix on those two new tasks.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing or research with human subjects. The experimental methodology is entirely computational, using LLMs and benchmark datasets without human participants or annotations.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve human subjects research that would require IRB approval. All experiments are conducted using computational models and pre-existing datasets, with no human participants involved in the study.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper appropriately describes the LLMs used in the research, which are central to the study's methodology. Section 3.1 (p.4-5) provides details about the models used (LLaMA-3.3 70B Instruct, Llama-4-Scout-17B-16E, Llama-4-Maverick-17B-128E-Instruct-FP8, and GPT-4.1 mini). Since studying LLMs' ability to incorporate feedback is the core focus of the paper, these models and their configurations are thoroughly documented.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

https://doi.org/10.52202/085713-1831