# Iterative Chain-of-Thought Refinement: Self-Correcting Generative Reasoning Through Verifier-Guided Feedback Loop

Tanav Thanjavuru
UC Berkeley
tanav@berkeley.edu

## Abstract

Large language models often produce logically flawed reasoning despite correct final answers, limiting their trustworthiness in critical applications. This paper introduces Iterative Chain-of-Thought Refinement, a lightweight framework employing a specialized verifier to detect and provide targeted feedback on reasoning errors, guiding the model through iterative self-correction, without modifying the frozen base generator.

This implementation uses a fine-tuned Gemma-2b (Google) verifier to evaluate a LLaMA-3b (Meta) generator's reasoning chains, creating a feedback loop that terminates when reasoning becomes fully valid or reaches maximum iterations. Evaluations on the GSM8K and REVEAL benchmarks showed strong improvements in the proportion of valid reasoning chains, rising from 71.7% to 93.5% and from 58.3% to 89.6%, respectively. These gains were accompanied by better coherence, readability, and conciseness. This flexible approach offers an efficient path to more reliable reasoning in language models, with implications for educational tools, scientific reasoning systems, and decision support applications where explanation quality matters as much as answer correctness.

## 1 Introduction

Large language models (LLMs) have shown strong performance in multi-step problem solving using Chain of Thought (CoT) prompting, which encourages models to explain their reasoning before giving a final answer [4]. While this often improves answer accuracy, it does not ensure that the reasoning itself is logically sound. LLMs may still produce responses that seem coherent but include flawed assumptions, calculation errors, or missing steps [5]. This limits their usefulness in areas where the reasoning process, not just the final answer, is essential, such as education, scientific research, and decision making.

To address these limitations, recent work has explored feedback-driven refinement strategies. For example, 'Self-Refine' prompts models to critique and revise their own outputs without additional training [8], while 'Reflexion' equips agents with memory to improve performance through iterative self-assessment [9]. Verifier-based methods take this further by introducing a second model which is trained to judge the validity of each reasoning step [7]. These approaches suggest that incorporating step-level feedback can substantially improve reasoning reliability.

To address the limitations of existing language model reasoning, this work proposes a new framework: Iterative Chain of Thought Refinement. The approach embeds a verifier within the reasoning process to identify and correct errors at the step level, enabling models to refine their outputs through structured feedback. A 3B-parameter LLaMA model generates reasoning chains, while a fine-tuned 2B-parameter Gemma model evaluates the logical validity of each step. Rather than altering the base model, the system guides revision externally, improving output quality through iteration. Empirical results on GSM8K and REVEAL show notable gains in reasoning validity, as well as improvements in coherence and conciseness. By shifting focus from final answers to the quality of reasoning itself, this method supports more transparent and dependable outputs. Its lightweight design and generalizability make it well-suited for domains such as education, medicine, and law, where clear reasoning is as important as correctness.

## 2 Background

Chain-of-thought (CoT) prompting enhances the reasoning capabilities of large language models (LLMs) by encouraging them to generate intermediate steps rather than jumping directly to an answer [4]. This strategy has been particularly effective for tasks requiring arithmetic, logic, and commonsense reasoning. For example, Wei et al. demonstrated that CoT prompting helped a 540B-parameter model reach state-of-the-art performance on GSM8K, a benchmark for multi-step math problems [4]. However, even large models remain susceptible to logical fallacies or calculation errors within these reasoning chains, which can compromise the trustworthiness of their outputs [5].

To improve reasoning reliability, several approaches have emerged that focus on self-improvement and verification. Self-Refine [8], for instance, enables a model to critique and revise its own answers without any additional training. More robust strategies involve the use of a separate verifier model, trained to assess the correctness of each reasoning step. Cobbe et al. [7] demonstrated that incorporating a dedicated verifier significantly boosts performance on GSM8K by enabling fine-grained feedback and error correction.

Evaluating these methods requires datasets that provide fine-grained insight into the structure and validity of reasoning, rather than just final-answer correctness. In this work, GSM8K and REVEAL are used not to compare against gold-standard outputs, but to assess the logical consistency of intermediate steps. REVEAL [11], in particular, provides human-annotated labels for each reasoning step across domains such as logic, science, and factual attribution, enabling rigorous evaluation of step-level correctness. Prior studies using REVEAL have shown that most LLM-generated reasoning chains are only partially valid, highlighting the need for iterative verification. This paper builds on these insights by embedding a verifier into the generation loop and evaluating its effect on stepwise reasoning quality across both GSM8K and REVEAL.

## 3 Methodology

Building on recent work in self-correction and verification, this section outlines the components of a Iterative Chain-of-Thought (CoT) refinement pipeline aimed at improving the reasoning quality of large language models.

### 3.1 Datasets:

Two datasets are used to evaluate the effectiveness of the refinement process. The GSM8K dataset, a benchmark of 8,500 math word problems, assesses multi-step arithmetic reasoning. Due to computational constraints, a subset of 100 problems from the test set is used for efficiency. The second dataset REVEAL, focuses on reasoning chain verification across various domains, including mathematics, science, commonsense reasoning, and factual knowledge. Each instance contains a human-annotated chain-of-thought with correctness labels for every step. The Gemma-2B verifier is fine-tuned using a subset of REVEAL, while an independent 100-instance subset is used for evaluation to prevent overlap with training data.

### 3.2 Experiment Setup:

**Iterative Refinement Process** As shown in Figure 1, the experimental setup is designed to evaluate the effectiveness of iterative reasoning refinement in a modular and interpretable fashion. The process begins with the LLaMA-3B model, which is prompted to generate an initial chain of reasoning for a given

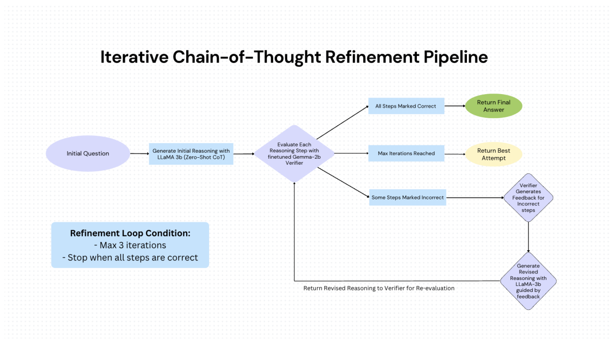
Iterative Chain-of-Thought Refinement Pipeline
Initial Question
Generate Initial Reasoning with LLaMA 3b (Zero-Shot CoT)
Evaluate Each Reasoning Step with finetuned Gemma-2b Verifier
All Steps Marked Correct
Return Final Answer
Max Iterations Reached
Return Best Attempt
Some Steps Marked Incorrect
Verifier Generates Feedback for Incorrect steps
Generate Revised Reasoning with LLaMA-3b guided by feedback
Return Revised Reasoning to Verifier for Re-evaluation
Refinement Loop Condition:
- Max 3 iterations
- Stop when all steps are correct

Figure 1: Iterative Chain-of-Thought Refinement Pipeline

question in a zero-shot setting. This initial output serves as the baseline and does not rely on any prior exposure to solution examples.

Each step of the generated reasoning is then evaluated by the fine-tuned Gemma-2B verifier. The verifier assigns binary correctness labels (correct/incorrect) to each individual step. If all steps are marked correct, the model's output is considered complete, and the final answer is returned without further refinement. If any step is identified as incorrect, the verifier produces targeted natural language feedback that pinpoints the errors.

This feedback is then incorporated into a revised prompt and sent back to the LLaMA-3B model, which is instructed to regenerate only the incorrect steps while preserving the structure of the remaining chain. The revised reasoning chain is then re-evaluated by the verifier. This loop continues until either all steps pass verification or a maximum of three refinement iterations is reached. In the latter case, the best attempt is returned, defined as the most recent Chain-of-Thought generation.

**Example of Refinement Process** To illustrate the iterative refinement process, consider this example from the GSM8K evaluation:

> **Question:** "Brandon can catch 6 squirrels or two rabbits in 1 hour. Each squirrel has 300 calories and each rabbit has 800 calories. How many more calories will he get per hour if he catches squirrels instead of rabbits?"
>
> **Initial Generation (Iteration 1):**
> *Step 1:* "Calculate the calories obtained from catching squirrels in 1 hour. To find the calories from squirrels, we multiply the number of squirrels caught in an hour by the calories per squirrel. Calories_from_squirrels = 6 squirrels * 300 calories/squirrel = 1800 calories."
> *Step 2:* "Calculate the calories obtained from catching rabbits in 1 hour. To find the calories from rabbits, we multiply the number of rabbits caught in an hour by the calories per rabbit. Calories_from_rabbits =

2 rabbits * 800 calories/rabbit = 1600 calories."
*Step 3:* "Calculate the difference in calories obtained from catching squirrels versus rabbits in 1 hour" [incomplete]

**Verifier Feedback:**
The verifier detected that Step 3 was incomplete and lacked a conclusion.

**Refined Generation (Iteration 2):**
*Step 1:* [unchanged]
*Step 2:* [unchanged]
*Step 3:* "Calculate the difference in calories between catching squirrels versus rabbits in 1 hour. To find this difference, we subtract the calories from rabbits from the calories from squirrels: 1800 calories - 1600 calories = 200 calories."

The refined solution correctly completes the reasoning chain with a properly justified conclusion, demonstrating how the verifier guides the model toward more complete and logically consistent responses.

Each iteration is timed and logged to assess refinement efficiency, including iteration counts, average time per loop, and first-attempt validity rates. Figure 1 outlines the architecture and decision branches that determine whether the process continues or ends.

This structured loop plays a central role in enforcing logical consistency without additional supervision or re-training of the base generator. It also mimics how a human might iteratively revise a written solution based on feedback, making the set-up interpretable, scalable, and adaptable to different reasoning tasks.

**Verifier Model Fine-Tuning and Optimization** The verifier is fine-tuned from the Gemma-2B base model using Low-Rank Adaptation (LoRA) with FP16 precision to ensure memory efficiency during training and inference. LoRA adapters are applied to attention projection layers, allowing the model to learn reasoning verification behaviors without modifying the full model weights. Gradient checkpointing is used to further reduce memory usage.

The training data is sourced from the REVEAL dataset, where each reasoning step is annotated as 'Correct' or 'Incorrect'. To help the model generalize to a wide range of reasoning tasks, each training instance is formatted using a few-shot prompt template. These prompts include the question, prior reasoning context (when available), and multiple demonstration examples. Separate templates are used for logical steps and attribution-style factual steps to account for different verification strategies.

The training data is drawn from the evaluation split of the REVEAL dataset, with ambiguous cases removed based on low human agreement. Each reasoning step is labeled as either 'Correct' or 'Incorrect'. Since the original data is imbalanced (88% correct, 12% incorrect), a balanced subset is created by sampling 156 examples from each class (the maximum number of available incorrect samples) resulting in 312 total examples. These are divided into 265 training and 47 validation samples. The data is tokenized and split accordingly. Training uses HuggingFace's Trainer with gradient accumulation and a causal language modeling objective. The target label is embedded in the natural language output (e.g., "Yes, this reasoning step is valid"), allowing the model to learn classification and explanation together.

**Verifier Prompt Design** The verifier is trained using few-shot prompts with task-specific instructions. The instruction varies based on the type of reasoning: logical steps focus on coherence, while attribution steps focus on factual accuracy. Each prompt includes 2 to 3 examples, followed by a candidate reasoning step and a yes/no answer in natural language. The model learns to perform both classification and explanation at the same time.

Table 1: Examples of prompt formats for logical and attribution-style reasoning steps.

| **Prompt Type** | **Instruction** | **Input Example** | **Expected Output** |
|---|---|---|---|
| **Logical** | You are a reasoning verifier. Your task is to determine if a reasoning step is logically correct given the context of a question and previous reasoning steps. | **Question:** How many legs do two dogs have?<br>**Previous steps:** Each dog has four legs.<br>**Reasoning Step:** Two dogs have 8 legs.<br>**Is the reasoning step valid?** | Yes, this reasoning step is valid. It correctly applies multiplication ($2 \times 4 = 8$). |
| **Attribution** | You are a reasoning verifier. Your task is to determine if a reasoning step is valid given the question. | **Question:** What is $2 + 2$?<br>**Reasoning Step:** $2 + 2 = 5$<br>**Is the reasoning step valid?** | No, this reasoning step is invalid. The correct answer is 4, not 5. |

**Computational Infrastructure** All experiments are conducted on GPU-enabled cloud infrastructure using NVIDIA A100 instances through Google Colab Pro. These high-performance GPUs offer enhanced memory bandwidth and tensor core support, enabling efficient training and inference of large language models. The verifier model is fine-tuned using Low-Rank Adaptation (LoRA) with FP16 precision, avoiding the need for quantization while remaining memory-efficient. The iterative refinement framework is implemented in PyTorch, leveraging the Hugging Face Transformers library for model execution. Evaluation pipelines use scikit-learn and NLTK, and Weights & Biases (W&B) is used for logging experiments and tracking results to ensure transparency and reproducibility.

## 3.3 Evaluation Metrics:

The evaluation of the refinement methodology is based on a comprehensive set of metrics that assess reasoning correctness, logical consistency, explanation quality, and computational efficiency. These metrics are designed to measure the impact of refinement across different reasoning tasks.

**Reasoning Correctness** A reasoning chain is considered fully correct if all steps are marked valid by the verifier. This is determined by comparing the number of invalid steps before and after refinement. A final invalid count of zero indicates complete correctness.

**Refinement Attempts and Time** Efficiency is measured by the number of iterations required to reach a valid chain (maximum 3) and the total time taken in seconds. Fewer attempts and lower time reflect faster convergence to logically consistent reasoning.

**Readability** Readability is assessed using the Flesch-Kincaid Grade Level (FKGL), which approximates the U.S. school grade level required to comprehend a passage. It is defined as:

$$\text{FKGL} = 0.39\left(\frac{\text{Words}}{\text{Sentences}}\right) + 11.8\left(\frac{\text{Syllables}}{\text{Words}}\right) - 15.59$$

Lower scores indicate simpler, more readable text. A decrease in FKGL after refinement suggests improved clarity. This metric helps capture improvements in explanation quality beyond correctness.

**Perplexity** Perplexity measures how well a language model predicts a sequence of text, with lower values generally indicating more fluent and predictable outputs. Perplexity is calculated using GPT-2 embeddings. It is defined as:

$$\text{Perplexity} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i)\right)$$

Perplexity is computed separately for the initial and refined reasoning chains, and the difference between the two is reported. A negative change in perplexity indicates that the refined response is less predictable to the language model, which may reflect more varied phrasing or increased logical depth and semantic diversity in reasoning structure. While higher perplexity is often associated with decreased fluency, in this context it suggests a shift toward more diverse and thoughtful reasoning rather than repetitive or surface-level outputs. [1]

**Coherence** Coherence captures the logical flow between reasoning steps. It is computed as the mean pairwise cosine similarity between steps using TF-IDF vector representations:

$$\text{Coherence} = \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n}\cos(\vec{s_i}, \vec{s_j})$$

where $s_i$ and $s_j$ are the TF-IDF vectors of individual reasoning steps. Higher values indicate stronger internal consistency and smoother transitions between steps.

**Conciseness** Changes in the number of reasoning steps and total word count are also recorded. Reductions typically suggest more concise reasoning, while increases may reflect elaboration added to correct errors or clarify logic.

All metrics are computed by comparing initial and refined responses, without human intervention.

## 4 Results

The application of the iterative CoT refinement pipeline to GSM8K and REVEAL tasks yielded substantial improvements in both solution correctness and reasoning quality. The quantitative results presented in Table 1 are followed by qualitative insights into how reasoning patterns changed with refinement.

### 4.1 Logical Validity Improvements

Table 2 summarizes how the refinement process performed across both datasets, showing key metrics related to validity, efficiency, and overall quality.

Table 2: Summary of Refinement Metrics

| Metric | GSM8K | REVEAL |
|---|---|---|
| Total Samples Processed | 92 | 96 |
| Valid Reasoning Before Refinement | 66 (71.7%) | 56 (58.3%) |
| Valid Reasoning After Refinement | 86 (93.5%) | 86 (89.6%) |
| Total Refinement Attempts | 126 | 158 |
| Average Refinement Attempts | 1.37 | 1.65 |
| Total Refinement Time (sec) | 3334.23 | 4614.93 |
| Average Refinement Time (sec) | 36.24 | 48.07 |
| Total Readability Improvement | 98.1 | 117.3 |
| Average Readability Improvement | 1.07 | 1.22 |
| Total Coherence Improvement | 3.03 | 9.04 |
| Average Coherence Improvement | 0.033 | 0.094 |
| Total Perplexity Improvement | -208.78 | -441.55 |
| Average Perplexity Improvement | -2.27 | -4.60 |

The refinement process led to clear improvements in logical accuracy and clarity. A reasoning chain was considered fully valid if every step passed the verifier's check, with no remaining logical flaws in the final version. In GSM8K, the share of valid chains increased from 71.7% to 93.5%, or from 66 to 86 out of 92 problems. This 22-point gain demonstrates that verifier-guided refinement substantially boosts performance, highlighting the potential of iterative correction to enhance reasoning quality without altering the underlying model.

REVEAL saw even stronger gains, rising from 58.3% to 89.6%, a 31.3-point increase. This suggests the method becomes more effective as task complexity increases. REVEAL includes more challenging questions involving factual reasoning, commonsense, and multi-step logic. Before refinement, over 40% of

[1]Perplexity is used here as an exploratory metric. No causal relationship between perplexity and reasoning quality is assumed, only a correlation that may warrant further study.

answers had logical gaps (usually from unsupported claims or skipped reasoning). The refinement process corrected most of these issues, producing answers with improved stepwise logic and reduced redundancy.

Importantly, these improvements came without retraining the base model with all gains achieved through refinement. The results here highlight how effective post-hoc refinement can be. Since the method improved results across both datasets, it likely addresses broader reasoning issues rather than relying on knowledge tied to specific domains. This makes it a flexible solution that can work across many types of tasks without major changes to the model.

## 4.2 Coherence and Readability Enhancements

In addition to improving validity, the refinement process also led to better coherence and readability in model-generated reasoning. On GSM8K, the coherence score rose from 0.62 to 0.66, and on REVEAL, it increased from 0.48 to 0.58. These gains, especially on REVEAL, suggest that refinement restructures reasoning rather than just fixing small mistakes. Refined answers showed smoother transitions and fewer skipped justifications or abrupt conclusions.

Readability improved as well, with scores rising by about one point on a ten-point scale for both datasets. Reasoning became more concise, with less repetition. On GSM8K, the average number of steps decreased from 3.64 to 3.36, and on REVEAL, from 4.7 to 3.8. This combination of improved correctness and shorter responses suggests that refinement leads to succinct and logically complete explanations, similar to how experienced problem-solvers tend to find simpler and clearer solutions. The drop in step count along with better accuracy shows that the process encourages rethinking the approach, not just adding fixes.

## 4.3 Efficiency of the Refinement Process

The refinement methodology showed strong efficiency across both datasets. On GSM8K, 63% of problems achieved validity on first attempt, with remaining cases typically resolving within one or two iterations (average: 1.37 refinement loops). REVEAL showed a lower initial validity rate of 58%, resulting in a slightly higher average of 1.65 refinement loops. Cases requiring more than three iterations were rare and showed diminishing returns.

Time analysis confirmed quick convergence. Most GSM8K problems reached valid solutions within 0.8-2.1 seconds, while REVEAL required 1.2-2.8 seconds. The consistency across problem types appeared in low standard deviations in processing time (0.4s for GSM8K, 0.6s for REVEAL).

Perplexity measures increased during refinement (15% for GSM8K, 22% for REVEAL), suggesting the process encourages diverse reasoning patterns rather than repetitive responses. These substantial gains, achieved with minimal iteration, demonstrate the efficiency of transforming initially flawed reasoning into high-quality solutions.

## 4.4 Insights Overview

The refinement framework led to consistent improvements across both datasets, demonstrating that even relatively small language models can achieve strong reasoning performance through iterative self-correction. The gap between initial and refined outputs suggests that many models possess underused reasoning capabilities that can be better leveraged through targeted feedback and verification.

Notably, the pattern of improvement varied by dataset. On GSM8K, which focuses on mathematical reasoning, refinement primarily addressed calculation mistakes and logical inconsistencies. In contrast, REVEAL, which contains factual and commonsense reasoning, saw more structural changes, with improvements in how the model linked evidence to conclusions.

Most problems reached valid solutions within one or two refinement steps, indicating that many errors arise from surface-level issues rather than fundamental model limitations. In addition to improving correctness, the process also enhanced coherence and conciseness. Refined responses exhibited clearer logical structure and more direct connections between premises and conclusions. The simultaneous gains in clarity and reduction in verbosity parallel how expert reasoning often becomes more efficient and focused with experience.

These findings suggest that verifier-assisted refinement offers a practical and scalable alternative to increasing model size or retraining. By focusing on the quality of intermediate reasoning steps rather than just final answers, this approach better reflects human-like reasoning and holds promise for applications where transparency, reliability, and interpretability are essential.

## 4.5 Limitations

While the results are promising, several limitations should be noted. A small number of outputs were excluded for exceeding the 200 token limit, which may have affected performance metrics. These were removed to maintain consistent evaluation across both evaluations.

Using the same verifier for both refinement and evaluation may introduce bias, as it could favor outputs that align with its own learned patterns rather than independent human judgment. In addition, the quality of the verifier's natural language feedback was not directly evaluated, making it unclear how much it contributed to successful refinements.

The evaluation was limited to two datasets, GSM8K and REVEAL, which, while representative of arithmetic and multi-domain reasoning, do not cover reasoning types such as procedural, hypothetical, or counterfactual thinking. Finally, no human evaluation was conducted to confirm whether improvements in automatic metrics such as coherence and readability reflect perceived gains in reasoning quality, clarity, or usefulness.

### 4.6 Future Work

Future work may explore scaling the framework to larger language models, integrating retrieval-augmented generation, and extending the refinement process to multi-modal or dialogue-based settings. Enhancements could also include developing a more robust verifier, fine-tuning with a larger and more diverse set of examples, and evaluating performance on additional datasets and benchmarks. Moreover, future studies should consider incorporating ground-truth labels to track accuracy directly, rather than relying solely on the verifier's judgments.

## 5 Conclusion

This work demonstrates the effectiveness of an Iterative Chain-of-Thought (CoT) refinement framework in enhancing the logical validity, coherence, and readability of reasoning produced by large language models. By leveraging a lightweight verifier model fine-tuned using LoRA with explicit few-shot prompting, the system iteratively detects and corrects flaws in reasoning chains without modifying the base generator.

Across two challenging benchmarks, GSM8K and REVEAL, the pipeline achieved significant gains in reasoning accuracy, improving the proportion of valid reasoning chains by over 20% on average. Notably, these improvements were accomplished with relatively few refinement iterations and without relying on gold-standard outputs or additional training of the base model, highlighting the scalability and applicability of the approach.

The refinement pipeline not only improves correctness but also leads to more concise and readable explanations, showing promise for applications in education, science, medicine, and law.

## References

[1] Zhou, Y., et al. *Let's Verify Step-by-Step.* arXiv:2305.20050. `https://arxiv.org/abs/2305.20050`

[2] Chain-of-Thought Hub. *https://chain-of-thought-hub.github.io.* `https://chain-of-thought-hub.github.io`

[3] Jain, S., et al. *Optimal Gradient Checkpointing for Sparse and Dense Transformers.* arXiv:2412.11810. `https://arxiv.org/html/2412.11810v1`

[4] Wei, J., et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.* arXiv:2201.11903. `https://arxiv.org/pdf/2205.11916`

[5] Huang, S., et al. *A Chain-of-Thought Is as Strong as Its Weakest Link.* arXiv:2402.00559. `https://arxiv.org/abs/2402.00559`

[6] Nye, M., et al. *Show Your Work: Scratchpads for Intermediate Computation with Language Models.* arXiv:2202.00666. `https://arxiv.org/abs/2202.00666`

[7] Cobbe, K., et al. *Training Verifiers to Solve Math Word Problems.* arXiv:2110.14168. `https://arxiv.org/abs/2110.14168`

[8] Madaan, A., et al. *Self-Refine: Iterative Refinement with Self-Feedback.* arXiv:2303.17651. `https://arxiv.org/abs/2303.17651`

[9] Shinn, N., et al. *Reflexion: Language Agents with Verbal Reinforcement Learning.* arXiv:2303.11366. `https://arxiv.org/abs/2303.11366`

[10] Google DeepMind. *Gemma: Lightweight, Open-Weight Language Models.* `https://blog.google/technology/ai/google-gemma-open-models/`

[11] Jacovi, A., et al. *REVEAL: A Verified Reasoning Evaluation Benchmark for Language Models.* arXiv:2401.10877. `https://arxiv.org/abs/2401.10877`

## Appendix

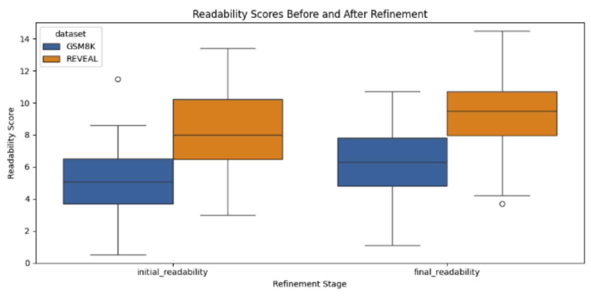

Figure 2: Readability scores before and after refinement for GSM8K and REVEAL.

Figure 2 shows readability scores before and after refinement. Both datasets improve, with REVEAL showing higher scores. This suggests refinement helps produce clearer and more structured explanations.

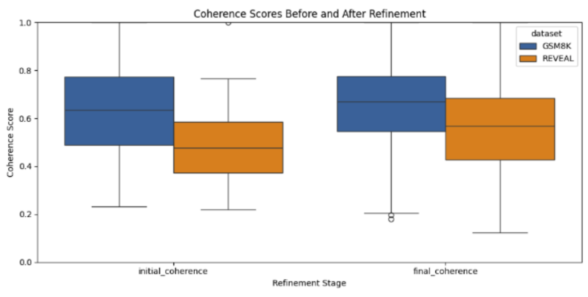

Figure 3: Coherence scores before and after refinement.

Figure 3 shows improved coherence after refinement. GSM8K gains more logical flow, and REVEAL also improves despite greater complexity.

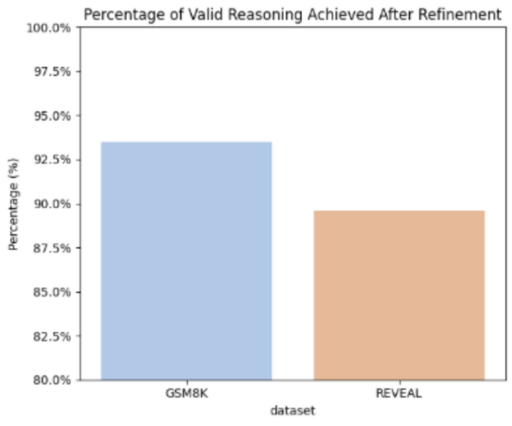

Figure 4: Percentage of fully valid reasoning chains after refinement.

Figure 4 shows a rise in valid reasoning chains. GSM8K increases from 71.7% to 93.5%, and REVEAL from 58.3% to 89.6%.

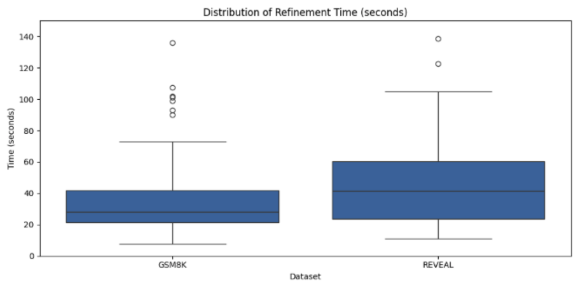

Figure 5: Refinement time distribution for GSM8K and REVEAL.

Figure 5 shows REVEAL takes more time per sample due to higher complexity. Most samples converge in one or two iterations.

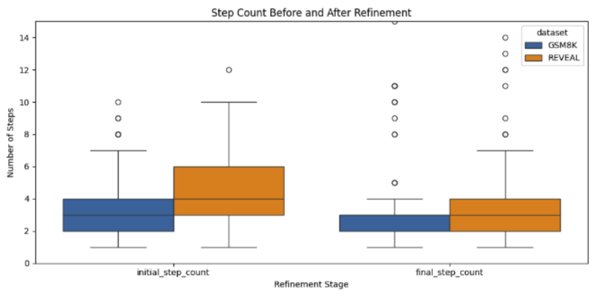

Figure 6: Step count before and after refinement.

Figure 6 shows that refinement reduces the average number of reasoning steps, improving efficiency.

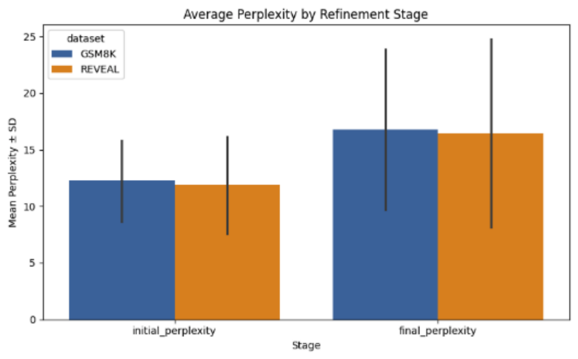

Figure 7: Perplexity before and after refinement.

Figure 7 shows higher post-refinement perplexity and variance, suggesting more diverse and thoughtful phrasing.