Memory-Augmented Critique for Collaborative Self-Improvement in Super Tiny Language Models

Anonymous TACL submission

Abstract

Adapting Super Tiny Language Models (STLMs) to specialized tasks remains challenging due to their limited capacity and compute budgets. We propose a memory-augmented critique framework that enables collaborative, self-supervised refinement of STLMs through structured feedback. Our approach orchestrates multiple LoRA-tuned expert models alongside a separate judge model, leveraging a persistent memory module to store critiques. These critiques are subsequently transformed into instruction-tuning prompts for second-stage fine-tuning. Evaluated on dialogue summarisation (SAMSum), our method achieves statistically significant performance improvements: average ROUGE-1 F1 increases from 0.393 to 0.406 in the 3-expert setup ($p < 0.05$), with 66.7% of experts showing individual gains. Additionally, we identify a maximum memory size threshold 90% of the evaluation set, beyond which overfitting occurs. The results suggest that our critique-driven architecture offers a scalable, supervision-free alternative to standard fine-tuning for STLMs.

1 Introduction

Large Language Models (LLMs) have demonstrated exceptional performance across a wide range of natural language processing (NLP) tasks. However, adapting these models for specific applications typically demands extensive human-labelled data, costly fine-tuning, and iterative supervision. This reliance on human input imposes scalability limitations and hinders smaller models from autonomously generalising across domains.

Motivated by recent progress in self-improving learning paradigms, we propose a novel framework for iterative, feedback-driven training that enhances the performance of Super Tiny Language Models (STLMs) (Hillier et al., 2024) through structured critique and memory-guided refinement. Although conceptually similar to Self-Play Fine-Tuning (SPIN) (Chen et al., 2024), our method introduces a modular, multi-agent architecture coupled with a persistent memory module.

Specifically, we present a feedback-driven training pipeline [1] where multiple fine-tuned *expert* models generate responses for a set of tasks, and a separate *judge* model provides critique and improvement feedback. These critiques are stored in a shared memory structure, which is subsequently used to further refine the expert. This setup allows agents to learn not only from their own failures but also from collective peer insights, fostering collaborative, self-supervised refinement.

We evaluate our framework on extractive dialogue summarisation using the **SAMSum** (Gliwa et al., 2019) dataset. Our results demonstrate consistent performance improvements when using memory-based refinement. The contributions of this work are threefold:

- We propose a modular, memory-augmented critique-based framework for STLM training, independent of additional human supervision.
- We demonstrate that shared cross-expert feedback leads to more stable performance improvements compared to traditional fine-tuning or in-context updates.
- We provide empirical insights into how model performance varies with factors such as the number of experts, memory size, and feedback prompting strategies.

By leveraging collaboration and critique among small models, our framework lays the groundwork

[1] Code available at `https://github.com/rathorevedant99/nlp-polymind`

for scalable, low-resource alternatives to current LLM adaptation techniques.

2 Background

Improving Language Models. Improving the capabilities of language models (LMs) involves balancing scale, generalisation, and efficient task adaptation. While scaling improves reasoning and robustness, overfitting on downstream tasks can reduce transferability. Core evaluation metrics include accuracy, consistency, and generalisation, though ethical and fairness metrics are gaining traction in recent evaluation literature (Hu and Zhou, 2024).

A common approach to adapting LMs is **Supervised Fine-Tuning (SFT)**, where a pre-trained model is further tuned on task-specific labeled data. Early examples include GPT-1 (Radford et al., 2018) and BERT (Devlin et al., 2018), which demonstrated the effectiveness of pretraining followed by fine-tuning for diverse NLP tasks. More recent work (Pareja et al., 2024) highlights strategies for efficiently fine-tuning smaller models (3B–7B) under resource constraints, which is particularly relevant to our small-model setup.

Low-Rank Adaptation (LoRA). LoRA (Hu et al., 2021) is a parameter-efficient fine-tuning technique that injects low-rank residuals into the weights of frozen pre-trained models. It replaces full weight updates with trainable matrices $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$ such that:

$$W = W_0 + \alpha \cdot BA$$

where W_0 is the frozen base weight, and α is a scaling factor. LoRA enables efficient adaptation across tasks without introducing inference overhead or increasing storage significantly.

LLM-as-a-Judge Paradigm. The use of language models as evaluators—known as the *LLM-as-a-Judge* paradigm—has become increasingly popular for tasks such as summarisation, translation, and dialogue (Chiang and yi Lee, 2023). LLM judges often outperform automated metrics like ROUGE or BLEU in aligning with human preferences (Chiang and yi Lee, 2023; Liu et al., 2023), especially when guided via structured prompting. However, concerns around judge bias and inconsistency remain (Zheng et al., 2023), motivating further work on improving prompt design and calibration.

Prompting Strategies. Prompting remains central to guiding LLM behaviour across zero-shot, few-shot, and instruction settings. In **zero-shot prompting**, models are provided only the task description; in **few-shot prompting**, curated input-output pairs are included as demonstrations; and in **instruction prompting**, structured commands are used to elicit desired behaviours. Prompt design directly affects the output fidelity of LMs, particularly in low-resource or weak supervision regimes (Wei et al., 2022; Sanh et al., 2022; Brown et al., 2020).

Evaluation Metric: ROUGE-1 F1. We evaluate summarisation quality using ROUGE-1 (Lin, 2004), which measures unigram overlap between candidate and reference summaries. The F1 variant balances precision and recall, making it suitable for both extractive and abstractive summarisation tasks. Although simple, ROUGE remains widely adopted in benchmark evaluations and is used alongside model-based metrics in our experiments.

3 Related Work

Multi-Agent Fine-Tuning and Self-Improvement. Recent work explores how collaboration between agents can drive self-improvement in language models. Subramaniam et al. (2025) propose a multi-agent debate framework where multiple generator and critic agents engage in iterative feedback loops, enabling a wider exploration of reasoning paths and improves performance on structured tasks like GSM8K and MATH.

Chen et al. (2024) introduces **Self-Play Fine-Tuning (SPIN)**, where a model iteratively generates, critiques, and revises its own outputs to improve over time, without human supervision. Inspired by reinforcement learning and game-theoretic self-play, SPIN enables weak models to converge to stronger ones through feedback from comparisons with gold references. It outperforms supervised and instruction-tuned baselines in multiple reasoning tasks.

Memory Augmentation. The concept of external memory in neural models was first introduced by Neural Turing Machines (Graves et al., 2014), where a neural controller (typically an RNN) interfaces with a differentiable memory matrix. This architecture allows the model to learn to read from and write to memory using attention mechanisms, enabling more flexible sequence modelling. Through combining Retrieval-Augmented Generation (RAG) and critique within a self-reflective loop, Self-RAG (Asai et al., 2023) builds on this foundation. The model learns to iteratively retrieve relevant information, generate responses, and evaluate its own outputs to refine future predictions. The ReAct framework (Yao et al., 2023) similarly introduces a reasoning-action-feedback loop allowing LLMs to reflect on their own outputs. Likewise, Honovich et al. (2022) explore feedback collection for model-generated instructions, which are reused in downstream fine-tuning.

Our method draws inspiration from these paradigms, but focuses on scalable improvements in smaller models using modular adapters, memory accumulation, and shared feedback, without additional human annotation.

4 Methodology

The key component of our proposed methodology is the *expert-judge system*, where multiple expert models generate responses, and a judge module evaluates their performance and provides feedback. This feedback is stored in a *memory* along with the inputs and original responses. This memory is later used to guide additional fine-tuning of experts. The complete algorithm is described in Algorithm 1. The detailed execution pipeline is shown in appendix figure 3.

4.1 Expert Fine-Tuning

The training set is divided into multiple subsets (*Train Set 1, Train Set 2, ..., Train Set n*), where n represents the number of expert models. Each training subset is used to fine-tune a base model using LoRA. The fine-tuned models serve as individual **experts**. Together, these models form a team of experts that collaborate during the critique and refinement stages. We evaluate this team at the current stage to record the test set performance before training ($\mathcal{R}_{\textbf{pre}}$).

4.2 Critique and Feedback Mechanism

Once the experts are fine-tuned, a critique process is initiated, which proceeds through the following steps:

1. Expert response aggregation: A batch of tasks is selected from the evaluation dataset, and responses are collected from all experts for the selected batch.
2. Evaluation and feedback: A Judge module assesses the responses and provides feedback for each expert based on their performance in the current batch.

After each batch, the judge evaluates expert responses against the ground truth and provides a general feedback to each expert based on what can be done to improve it's response. The batch of tasks, feedback, ground truth, and original responses are stored and accumulated in a custom **memory module**.

4.3 Memory-Based Refinement

The feedback memory is converted into an instruction-tuning prompt format, which includes a batch of tasks, historical responses, corresponding ground truths, and the associated feedback. This prompt data is then processed and embedded as *inputs* and *labels*, forming a shared memory accessible to all expert models. By sharing a common memory, experts benefit from each other's feedback and insights. This shared memory facilitates consistent learning across the expert team, promoting collaboration and convergence toward improved performance. Using the stored instructions, a **Refined Team** is formed, in which each expert undergoes further adaptation via LoRA, incorporating memory-driven refinements.

4.4 LoRA-Based Fine-Tuning and Refinement

We apply LoRA in two stages: (1) expert-specific fine-tuning and (2) memory-based refinement using shared feedback. In each stage, a separate LoRA adapter is added to the frozen attention projections (query and value) of the transformer.

Given a frozen weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA introduces a low-rank residual:

$$\Delta W = \alpha \cdot BA, \quad A \in \mathbb{R}^{r \times d_{\text{in}}}, \quad B \in \mathbb{R}^{d_{\text{out}} \times r}$$

Each expert maintains two such adapters per projection:

- $\Delta W^{(1)}$: learned during initial fine-tuning
- $\Delta W^{(2)}$: learned during feedback-based refinement

At inference time, the final effective weight is computed as:

$$W_{\text{eff}} = W_0 + \Delta W^{(1)} + \Delta W^{(2)}$$

Algorithm 1 Expert Training and Iterative Refinement Pipeline

```
1: Input: Dataset $\mathcal{D}$; number of experts $n$; batch size $B$; critique rounds $R$
2: Output: Initial scores $\mathcal{R}_{\text{pre}}$, final scores $\mathcal{R}_{\text{post}}$, refined experts
3: Phase 1: Data Preparation
4: $\mathcal{D} \leftarrow \mathcal{D}^{\text{train}} \cup \mathcal{D}^{\text{val}} \cup \mathcal{D}^{\text{test}}$
5: $\mathcal{D}^{\text{train}} \leftarrow \bigcup_{i=1}^{n} \mathcal{D}^{(i)}_{\text{train}}$,    disjoint partitions for each expert
6: Phase 2: Expert Initialization (LoRA Phase 1)
7: for $i = 1$ to $n$ do
8:     Initialize expert $\text{E}_i$ from base model $M$
9:     Apply LoRA: $W \leftarrow W_0 + \alpha_1 B_1^{(i)} A_1^{(i)}$
10:    Train $\text{E}_i$ on $\mathcal{D}^{(i)}_{\text{train}}$ using LoRA adapters $(A_1^{(i)}, B_1^{(i)})$
11: end for
12: $\mathcal{R}_{\text{pre}} \leftarrow$ Evaluate $\text{E}_1, \dots, \text{E}_n$ on $\mathcal{D}^{\text{test}}$
13: Phase 3: Critique Loop
14: Initialize memory buffer $\mathcal{M} \leftarrow \emptyset$
15: for $r = 1$ to $R$ do
16:     Sample batch $(x_b, y_b) \sim \mathcal{D}^{\text{val}}$
17:     $\hat{y}_i \leftarrow \text{E}_i(x_b)$ for all $i$
18:     $\mathcal{F}_b \leftarrow \text{Judge}(\{\hat{y}_i\}, y_b)$
19:     $\mathcal{M} \leftarrow \mathcal{M} \cup \{(x_b, y_b, \{\hat{y}_i\}, \mathcal{F}_b)\}$
20: end for
21: Phase 4: Expert Refinement (LoRA Phase 2)
22: Construct instruction-tuning set $\mathcal{U}$ from $\mathcal{M}$
23: for $i = 1$ to $n$ do
24:     Add second adapter: $W \leftarrow W_0 + \alpha_1 B_1^{(i)} A_1^{(i)} + \alpha_2 B_2^{(i)} A_2^{(i)}$
25:     Train $\text{E}_i$ on $\mathcal{U}$ with $(A_2^{(i)}, B_2^{(i)})$
26: end for
27: Phase 5: Evaluation
28: $\mathcal{R}_{\text{post}} \leftarrow$ Evaluate $\text{E}_1, \dots, \text{E}_n$ on $\mathcal{D}^{\text{test}}$
29: return $\mathcal{R}_{\text{pre}}$, $\mathcal{R}_{\text{post}}$, $\{\text{E}_i\}_{i=1}^{n}$
```

This design allows for modular adaptation and supports ablation by disabling either adapter at inference time. The model is able to retain task-specific behaviour while incorporating cross-expert feedback. The total parameter overhead remains $O(r(d_{\text{in}} + d_{\text{out}}))$ per adapter. Full adapter configurations, dimensions, and projection equations are provided in Appendix B.1.

4.5 Evaluation

After refinement, the Test Set is used to evaluate the refined team to get the ($\mathcal{R}_{\textbf{post}}$). These are then compared with ($\mathcal{R}_{\textbf{pre}}$) from 4.1. Performance is measured using the **ROUGE-1** metric.

4.6 Statistical Evaluation Methodology

To assess the significance of performance improvements resulting from our framework, we employ standard statistical tests commonly used in NLP and machine learning evaluation (Dror et al., 2018; Demšar, 2006).

Specifically, we apply both the paired t-test (assuming normally distributed differences) and the Wilcoxon signed-rank test (a non-parametric alternative), to evaluate changes in ROUGE-1 F1 scores before and after self-improvement. This dual testing strategy ensures robustness.

We report p-values, *Cohen's* d (Cohen, 1988, pp. 20–27) (effect size), and 90% *confidence intervals* to quantify the magnitude and uncertainty of the observed improvements, encouraging interpretation beyond mere statistical significance. Since evaluations are conducted on the same samples before and after refinement, we treat them as *paired observations* (Devore, 2011, pp. 34–37, 325–326).

Significance level α is set to 0.05.

5 Implementation Details

5.1 Model and Datasets

For this study, we employ **google/flan-t5-small** (*77M parameters*) as the base model for expert models. Each expert is further fine-tuned using the SAMSum dataset for the summarisation. The dataset consists of human-written 16k dialogues and corresponding summaries, making it a valuable resource for training models in abstractive summarisation. We divide the dataset into train (14732 tasks), validation (818 tasks), and test (818 tasks) sets. In the critique phase, a separate model, **google/gemma-3-4b-it** (*4B parameters*), is designated as the Judge.

5.2 Prompt Designing

We experimented with various prompting strategies to enable the critic model to provide effective feedback. Initial attempts using zero-shot, few-shot, and standard instruction prompting resulted in vague or generic feedback. Additionally, providing detailed context often caused the model to copy specific details from the ground truth rather than offering general improvement instructions.

To overcome these limitations, we designed a hybrid prompting strategy combining instruction prompting, role-playing prompting, and feedback prompting with controlled output constraints. Specifically, the model was instructed to act as a teacher and provide one-line, generic instructions to each expert to help improve their answers, without mentioning the ground truth content.

5.3 Task Selection

We experimented with different tasks to find those best suited for our pipeline. Initially, we used the GSM8K dataset (Cobbe et al., 2021), containing 8.5K grade school math problems. Although our pipeline should have handled this dataset, performance was limited by the small expert models, which struggled with multi-step reasoning and basic arithmetic despite fine-tuning. The critic model also failed to effectively guide or evaluate their outputs. As shown in Appendix Figure 4, even with valid feedback, experts often failed to compute correct answers (e.g. 6×6), making it difficult to assess the pipeline's true effectiveness. Tasks requiring explicit reasoning and numerical accuracy, like math word problems, demand larger models for satisfactory performance. We therefore shift to tasks better suited to smaller models, such as dialogue summarisation.

For the summarisation task, using the SAMSum dataset, the combination of expert and critic models produced generally coherent and logical summaries. The expert models were able to generate reasonable summaries, while the critic provided feedback to help them refine their outputs and get closer to the ground truth.

5.4 Metric Selection

ROUGE-1 is used to evaluate performance, measuring unigram overlap between generated and reference summaries. It is well-suited for the SAMSum dataset, which contains short, extractive summaries, as it effectively captures key content while preserving coherence. ROUGE-1 is also widely adopted in summarisation tasks, particularly when retaining factual and essential information is critical.

Baselines. To contextualise our results, we report two baselines. First, naive LoRA fine-tuning on the full dataset results in a performance drop from 0.433 to 0.396, indicating that indiscriminate fine-tuning can harm performance in extremely small models. Second, we report the performance of InstructDS (Wang et al., 2023), a recent instruction-tuning method on a 3B params model, which achieves a ROUGE-1 F1 score of 0.553 on the same task, serving as a strong upper bound under full supervision.

6 Experiments

6.1 In-Context Feedback Update

We initially explored in-context feedback updates for the STLMs. In this setup, the judge LLM evaluates the expert's response against the ground truth and generates feedback, which is then appended to the expert's system prompt for subsequent queries. However, during experiments on the summarisation task, the critic model frequently produced contradictory or inconsistent feedback, resulting in unstable behaviour and a degradation in overall pipeline performance. An example of such an instance is illustrated in Figure 5.

6.2 Fine-tuning with Feedback Memory

All experiments in this section are conducted over 10 independent runs, with each run initialized from a fresh model state and a separately sampled training subset, i.e., a clean start. Final performance improvements are reported as the mean and standard deviation of the ROUGE-1 F1 score gains across these 10 runs, providing a measure of both central tendency and variability.

6.2.1 Initial Setup

For our second approach we used the memory module from section 4.3 to store feedbacks for multiple validation tasks across all experts. We performed this experiment with **3 experts** for **5 rounds** of discussions with **10 validation tasks** in each round. We then further refined the experts with LoRA fine-tuning on the feedbacks stored in memory module.

6.2.2 Varying number of Experts

To further evaluate the effectiveness of the setup, we varied the number of experts to observe its impact on performance. In addition to the original 3-expert configuration, we conducted experiments with **2 experts** and **4 experts**. Increasing the number of experts beyond four was not feasible due to computational constraints; specifically, the memory overhead from storing multiple expert-specific LoRA parameters and shared memory inputs exceeded available GPU capacity.

6.2.3 Varying the memory size

Our next ablation study investigates the relationship between the size of the feedback memory and its impact on model performance. To isolate this effect, we limit our experiments to configurations with 2 and 3 experts and analyse performance after fine-tuning using memory-based refinement across varying memory sizes.

6.2.4 Only memory fine-tuning

We also evaluated the effectiveness of memory-based fine-tuning in isolation, without any task-specific fine-tuning prior to the discussion phase with the judge. In this setting, the base models were initialized but not fine-tuned on the training dataset before entering the critique stage. The evaluation was conducted using a 3-expert configuration.

7 Results & Discussion

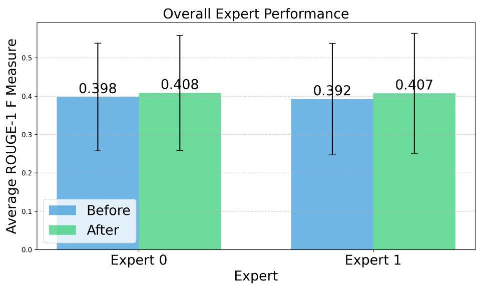
7.1 In-Context Learning for STLMs

With Fixed Feedback Size: Post fine-tuning, experts exhibit inconsistent behaviour as in-context feedback is added after each critique round. Performance is highly sensitive to the initial fine-tuning subset. For instance, ROUGE-1 F1 improves from 0 to 0.38 in one case, but drops from 0.5 to 0 in another (Appendix Figure 6). This instability was observed across **20 random subsets**.

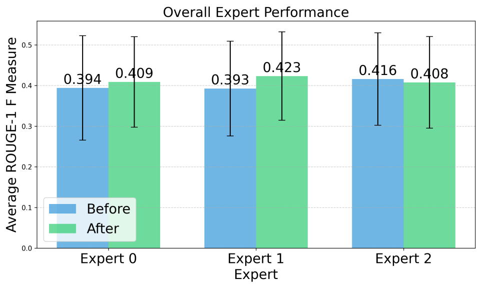
Effect of Varying Feedback Size: Using the same fine-tuning dataset, we varied the feedback retained in context. Performance dropped from 0.7 to 0.2 after five feedback rounds, then became erratic without a clear trend (Figure 7).

7.2 Finetuning with Feedback Memory

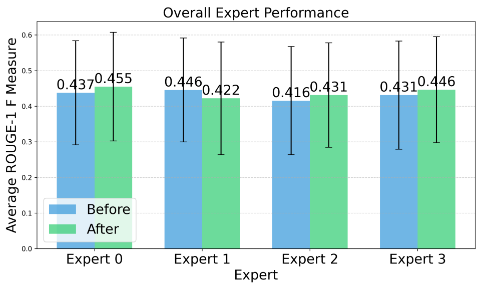
7.2.1 Base Result



(a) 2 Experts



(b) 3 Experts



(c) 4 Experts

Figure 1: Performance improvements after fine tuning with the cross-expert feedbacks

Figure 1b illustrates the performance of each of the 3 expert models before and after iterative refinement in 10 runs. Experts 0 and 1 demonstrate consistent improvement after feedback integration, while Expert 2 exhibits performance deterioration. The observed variance suggests sensitivity to both the initial fine-tuning data subset and the critique refinement process.

7.2.2 Varying number of Experts

In the 2-expert configuration, we observe performance improvements from 0.398 to 0.408 and from 0.392 to 0.407 for the two experts, respectively (Figure 1a). However, as the number of experts increases beyond two, at least one expert consistently experiences a performance drop. This is evident in Expert 2 in the 3-expert setup (Figure 1b) and Expert 1 in the 4-expert configuration (Figure 1c). Despite this, the remaining experts generally show performance gains, increasing from approximately 0.38 to 0.41 on average.

7.2.3 Varying the memory size

We evaluated how expert performance varies with the size of the feedback memory, using 2- and 3-expert setups with fine-tuning applied at fixed intervals. Each run consists of 75 tasks for 2 experts (5 rounds × 15 tasks) and 150 tasks for 3 experts (10 rounds × 15 tasks). As shown in Figure 2, performance improves steadily with memory size, peaking around 700–750 tasks (ROUGE1 F1≈0.49). Beyond this, performance degrades: one expert declines in the 2-expert setup (Figure 2a), while all decline in the 3-expert case (Figure 2b). Since the evaluation set contains 818 samples, results suggest that memory sizes exceeding 90% of the evaluation set lead to over-fitting or reduced generalisation.

7.2.4 Only memory fine-tuning

Figure 8 presents the performance of each expert before and after second-stage adapter tuning. The ROUGE-1 F1 scores show minimal overall change across all experts. Specifically, Expert 0 and Expert 2 experience slight declines (–0.016 and –0.004, respectively), while Expert 1 exhibits a modest improvement (+0.018).

These results suggest that instruction tuning alone, applied via a second adapter on top of an unadapted base model, is insufficient to yield consistent performance gains. This underscores the importance of task-specific fine-tuning during the initial phase (first adapter), which likely provides the necessary specialization for the critic's feedback to be meaningfully integrated during second-stage refinement.

7.3 Statistical Significance

Our analysis of ROUGE-1 F1 scores provides strong evidence for the effectiveness of the proposed memory-based refinement approach. We evaluate the 2-expert and 3-expert setup for 30 runs each and perform statistical tests on the results.

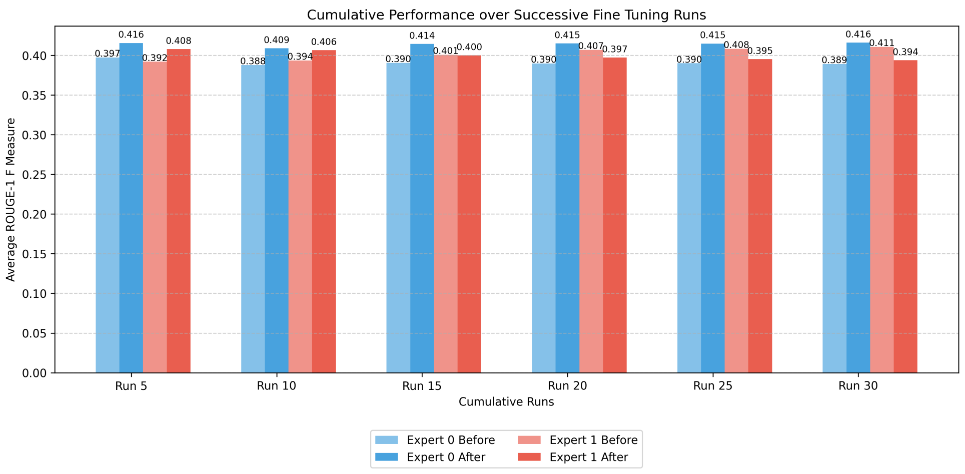
Three-Expert Configuration: In the 3-expert setup, we observed a statistically significant improvement in overall performance, with scores increasing from 0.393 ± 0.138 to 0.406 ± 0.136 (mean difference: $+0.013$, 90% CI: $[0.006, 0.021]$). This improvement was highly significant across all statistical tests ($p = 0.005$), and the effect size was non-negligible ($d = 0.096$). Furthermore, two of the three experts (66.7%) demonstrated statistically significant individual improvements (see Appendix C.1).

Two-Expert Configuration: The 2-expert configuration showed more modest overall improvement, with scores increasing from 0.396 ± 0.129 to 0.413 ± 0.142 (mean difference: $+0.018$, 90% CI: $[0.009, 0.026]$). This improvement was consistently significant across all tests, although the effect size remained small ($d = 0.13$).Notably, one of the 2 experts (50.0%) showed a clear and statistically significant individual gain, while the other expert exhibited only a marginal, non-significant improvement (Ref Appendix C.2)

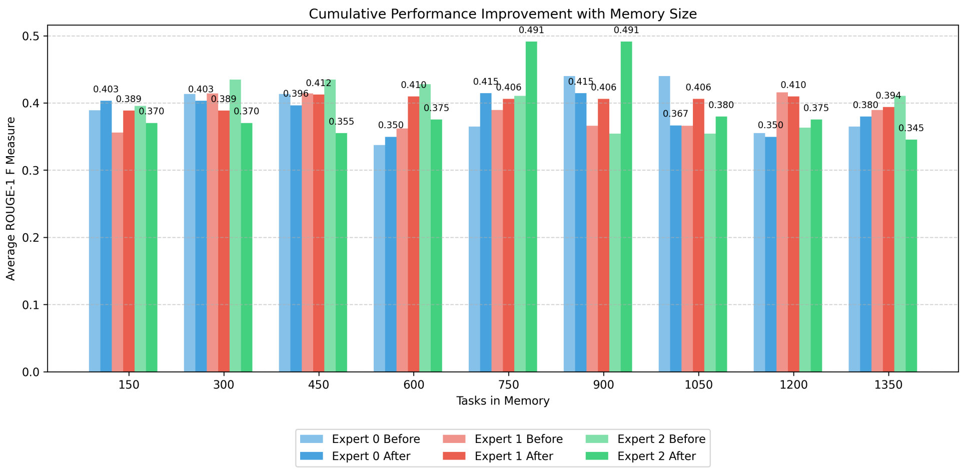
The 3-expert setup consistently outperforms the 2-expert variant, with improvements of 5.1–5.7% in summarisation quality. The results are robust across tests, indicating that memory-based refinement is effective, especially with three experts. Individual responses to refinement vary, pointing to the need for expert-specific adaptation strategies.

7.4 Limitations

Our experiments were constrained by computational resources, particularly GPU memory limitations. All critic models used had fewer than 4 billion parameters, and the expert models were restricted to fewer than 100 million parameters. The iterative nature of the critique process, involving multiple rounds of feedback and refinement, significantly increased computation time and memory usage compared to single-model baselines. This often led to out-of-memory (OOM) errors, especially during joint inference across multiple



(a) 2 Expert system with accumulated memory



(b) 3 Expert system with accumulated memory

Figure 2: Performance Metrics over Successive Fine Tuning runs with accumulated memory from past discussions

experts and judge models. To mitigate these issues, we limited the memory size to a maximum of 5 rounds with 10 tasks per round, aiming to maintain a manageable experiment failure rate ($\approx$33%). Additionally, due to the high computational cost and runtime of each experiment, statistical significance testing was conducted only for the most essential setups.

8 Conclusion

We proposed a memory-augmented critique framework for improving small language models (STLMs) without requiring human supervision. By combining structured critique, modular LoRA adapters, and a shared memory bank, our method enables collaborative refinement across expert agents. Empirical results show that it outperforms naive fine-tuning and in-context prompting, achieving statistically significant gains in ROUGE-1 F1. Performance peaks when memory size approaches, but does not exceed, the evaluation set, suggesting a trade-off between critique diversity and overfitting. These findings point to memory-driven refinement as a promising direction for self-improving STLMs.

Future Work. Future directions include scaling the framework to larger expert/judge models and extending it to other tasks such as question answering, reasoning, and code generation. Adaptive critique scheduling and improved prompting strategies could further enhance efficiency. Finally, enabling memory-based generalisation across tasks without retraining remains an open challenge.

9 References

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*.

Cheng-Han Chiang and Hung yi Lee. 2023. Can large language models be an alternative to human evaluations?

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd edition. Routledge, Hillsdale, NJ.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jay L Devore. 2011. *Probability and Statistics for Engineering and the Sciences*, 8th edition. Cengage Learning.

Rotem Dror, Guy Baumer, Ben Bogin, Alon Jacovi, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392. Association for Computational Linguistics.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. pages 70–79.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*, abs/1410.5401.

Dylan Hillier, Leon Guertler, Cheston Tan, Palaash Agrawal, Chen Ruirui, and Bobby Cheng. 2024. Super tiny language models.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Taojun Hu and Xiao-Hua Zhou. 2024. Unveiling llm evaluation focused on metrics: Challenges and solutions. *arXiv preprint arXiv:2404.09135*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment.

Aldo Pareja, Nikhil Shivakumar Nayak, Hao Wang, Krishnateja Killamsetty, Shivchander Sudalairaj, Wenlong Zhao, Seungwook Han, Abhishek Bhandwaldar, Guangxuan Xu, Kai Xu, Ligong Han, Luke Inglis, and Akash Srivastava. 2024. Unveiling the secret recipe: A guide for supervised fine-tuning small llms. *arXiv preprint arXiv:2412.13337*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Victor Sanh, Albert Webson, Colin Raffel, Shaden Smith Bach, Rami Aly, Alex Chaffin, Zaid Hsieh, Minjoon Kim, Mohamed Aly, Kyle Lo, and et al. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations (ICLR)*.

Vighnesh Subramaniam, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. 2025. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*.

Bin Wang, Zhengyuan Liu, and Nancy Chen. 2023. Instructive dialogue summarization with query aggregations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7630–7653, Singapore. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models.

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. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

A Methodology

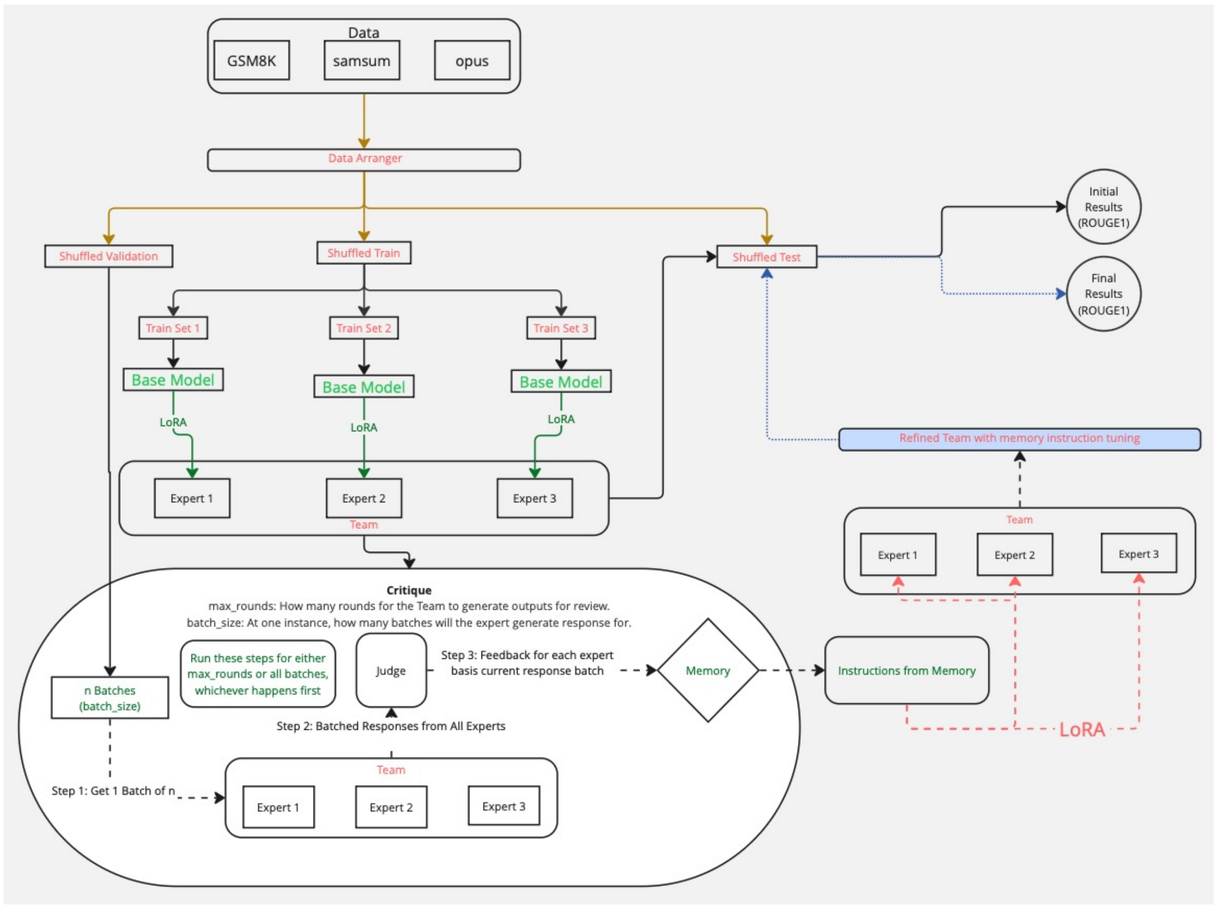

Data
GSM8K
samsum
opus
Data Arranger
Shuffled Validation
Shuffled Train
Shuffled Test
Initial Results (ROUGE1)
Final Results (ROUGE1)
Train Set 1
Train Set 2
Train Set 3
Base Model
Base Model
Base Model
LoRA
LoRA
LoRA
Expert 1
Expert 2
Expert 3
Team
Refined Team with memory instruction tuning
Team
Expert 1
Expert 2
Expert 3
Critique
max_rounds: How many rounds for the Team to generate outputs for review.
batch_size: At one instance, how many batches will the expert generate response for.
Run these steps for either max_rounds or all batches, whichever happens first
Judge
Step 3: Feedback for each expert basis current response batch
Memory
Instructions from Memory
LoRA
n Batches (batch_size)
Step 2: Batched Responses from All Experts
Step 1: Get 1 Batch of n
Team
Expert 1
Expert 2
Expert 3

Figure 3: Detailed Execution Pipeline

Math Task

Task: Molly is catering a birthday party for her sister and invited 16 people. 10 people want the chicken salad which is $6.50 per person and 6 people want the pasta salad at $6 per person. What is the total amount Molly will pay for the catering?

Round 1

Expert Answer: Molly needs 16 * 10 = 160 dollars for the chicken salad. Molly needs 16 - 6 = 16 more dollars for the pasta salad. Molly needs 160 + 16 = 176 dollars for the chicken salad. Molly needs 16 + 6 = 108 dollars for the pasta salad. Molly will pay 108 + 176 = $265 for the catering. Thus, the total amount Molly will pay for the catering is $265 + $265 = $155.
Critic Output: Please calculate the cost of the pasta salad separately and then add it to the cost of the chicken salad.

Round 10

Expert Answer: The cost of the chicken salad is $6.50 x 10 = $11. The cost of the pasta salad is $6 x 6 = $26. The total cost of the chicken salad and pasta salad is $11 + $26 = $33. The total amount Molly will pay is $33 + $26 = $121. The final answer: 121.
Critic Answer: Double check your multiplication and make sure you are adding the correct amounts.

Figure 4: Comparison of the Expert and Critic Outputs on a given Math task between Round 1 and Round 10.

Summarisation Task

Round 1

Expert Answer: Lucy is selling the rollerblades for a small discount. Lucy is not interested in the price list.
Ground Truth: Lucy's buying rollerblades for her niece. They have a few scratches, but are in a good condition and are good for beginners. Hannah won't give Lucy any discount. Lucy will think about buying them.
Critic Output: The answer is too brief and doesn't capture the full context of the situation.

Round 4

Expert Answer: Rosie's pen was not at his place yesterday. Harrison will give him hers tomorrow. Harrison will be there to take care of her. Harrison will be there to pick her up. Harrison will be around Harrison tomorrow. Harrison will be in his place.
Ground Truth: Rosie left her pen at Harrison's place yesterday. Harrison will bring it back to her in a while.
Critic Answer: The response is too verbose and includes unnecessary details about Harrison's actions.

Figure 5: Example of contradicting Critic Feedbacks on different summarisation tasks in the same batch.

B Implementation

B.1 LoRA Adapter Structure

In both training phases, LoRA introduces trainable low-rank updates to frozen transformer weights. For a projection matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA learns:

$$\Delta W = \alpha \cdot BA, \quad A \in \mathbb{R}^{r \times d_{\text{in}}}, \quad B \in \mathbb{R}^{d_{\text{out}} \times r}$$

We maintain two such adapters per projection:

- Phase 1: $\Delta W^{(1)} = \alpha_1 B_1 A_1$ (task-specific)
- Phase 2: $\Delta W^{(2)} = \alpha_2 B_2 A_2$ (feedback-driven)

The final effective weight is:

$$W_{\text{eff}} = W_0 + \Delta W^{(1)} + \Delta W^{(2)}$$

For an input batch $X \in \mathbb{R}^{b \times d_{\text{in}}}$, the final projection is:

$$Y = XW_0^\top + X(\alpha_1 B_1 A_1)^\top + X(\alpha_2 B_2 A_2)^\top$$

B.2 Set-Up

We maintain a configuration file (config) for all environment settings. We define a base class from which both our agents (Judge and Expert) are derived. Technical configurations worked with:

- **LoRA**
  - Rank: 8
  - Scaling Factor: 16
  - Dropout Probability: 0.1
  - Target Modules: q, v
- **Critique**
  - Task Batch Size: 15
  - Discussion Rounds: 5
- **GPU**: NVIDIA GeForce RTX 3090 (24GB Memory; 10,496 cores)
- **CPU**: Intel i9-12900K (24 cores)

C Detailed Statistical Analysis

C.1 3 Experts

Overall Improvement (All Experts Combined)

Metric	**Value**
Sample Size	900
Mean (Before)	0.393 ± 0.138
Mean (After)	0.406 ± 0.136
Mean Difference	$+0.013$ [90% CI: 0.006, 0.021]
Effect Size (Cohen's d)	0.096
Wilcoxon signed-rank test	$p = 0.005$
Paired t-test	$p = 0.005$
Randomization Test	$p = 0.008$

Table 1: Overall improvement in ROUGE-1 F1 scores across all experts after refinement.

Per-Expert Breakdown

Expert	**N**	**Before**	**After**	**Mean Diff**	**Cohen's d**	**Wilcoxon p**	**Randomization p**	**t-test p**
Expert 0	300	0.3846	0.4066	+0.0220	0.163	0.001	0.006	0.006
Expert 1	300	0.3900	0.4098	+0.0198	0.147	0.023	0.014	0.013
Expert 2	300	0.4054	0.4031	-0.0023	-0.016	0.90	0.762	0.783

Table 2: ROUGE-1 F1 score improvements and statistical significance for each expert using paired tests. Experts 0 and 1 show statistically significant improvements.

The Wilcoxon signed-rank test results are consistent with the t-test as well as the randomization test outcomes, with Experts 0 and 1 showing significance ($p < 0.05$) and Expert 2 not showing a meaningful change. In total, 2 out of 3 experts (**66.7%**) demonstrate statistically significant improvement after memory-based refinement.

C.2 2 Experts

Overall Improvement (All Experts Combined)

Metric	**Value**
Sample Size	600
Mean (Before)	0.396 ± 0.129
Mean (After)	0.413 ± 0.142
Mean Difference	$+0.018$ [90% CI: 0.009, 0.026]
Effect Size (Cohen's d)	0.13
Wilcoxon signed-rank test	$p = 0.0012$
Paired t-test	$p = 0.0008$
Randomization Test	$p = 0.0010$

Table 3: Overall improvement in ROUGE-1 F1 scores across both experts after refinement.

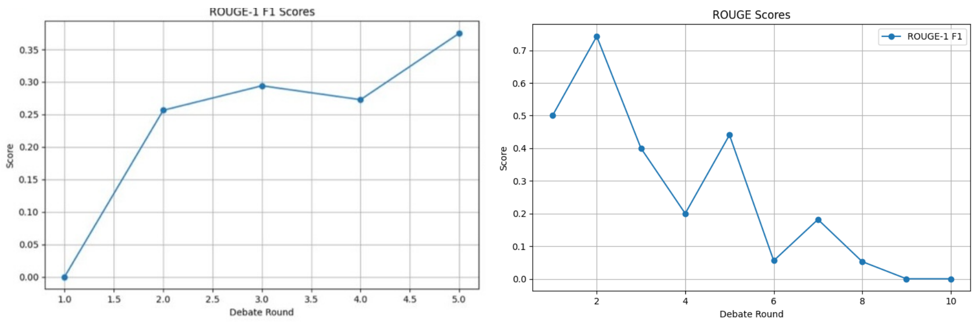
Per-Expert Breakdown

Expert	N	Before	After	Mean Diff	Cohen's d	Wilcoxon p	Randomization p	t-test p
Expert 0	300	0.3952	0.4227	+0.0275	0.206	0.0004	0.0010	0.0003
Expert 1	300	0.3964	0.4042	+0.0079	0.057	0.2847	0.2860	0.2831

Table 4: ROUGE-1 F1 score improvements and statistical significance for each expert using paired tests.

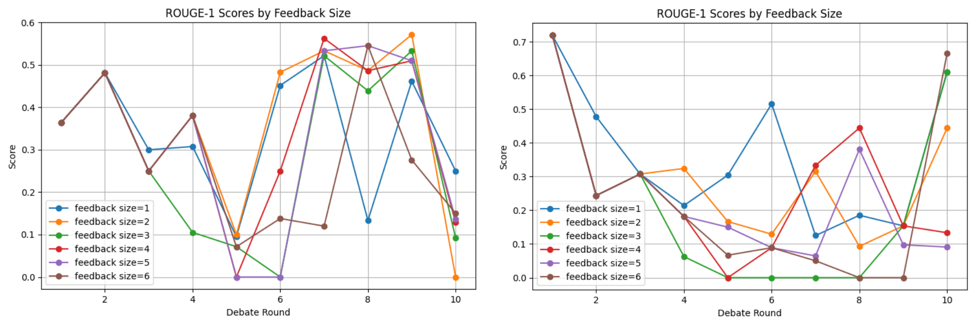
The overall results indicate a statistically significant improvement in ROUGE-1 F1 scores across both experts, with a small effect size ($d = 0.13$). Expert 0 shows a robust and statistically significant gain across all tests ($p < 0.001$), while Expert 1 shows only marginal improvements with no statistically significant change ($p > 0.28$ across tests). Thus, only one of the 2 experts benefits substantially from the refinement, suggesting that gains may be model-dependent.

D Addition to Results



(a) Improvement (b) Deterioration

Figure 6: In-context feedback shows inconsistent behaviour for small models



(a) Random Initialization 1 (b) Random Initialization 2

Figure 7: In-context feedback shows inconsistent improvements for small models

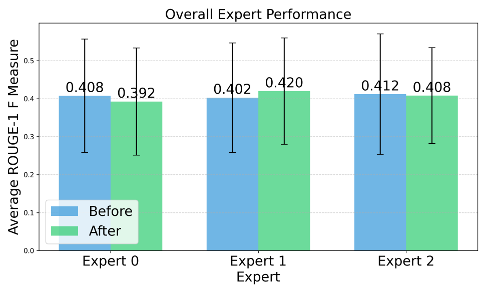


Figure 8: Performance with only memory fine-tuning