Iterative LLM Augmentation through Criteria-Based Feedback

Anonymous ACL submission

Abstract

Large Language Model (LLM) self-reflection involves an LLM reviewing its past outputs to enhance future responses without relying on external data. This concept has been explored in frameworks like those by Shinn(Shinn et al., 2024)and Madaan(Madaan et al., 2024). However, challenges remain, as Huang(Huang et al., 2023).point out the risk of performance degradation due to overly generic reflective prompts. To address these issues, we introduce a vector-based retrieval framework. Our approach demonstrates significant improvements in decision-making, reasoning, and mathematics tasks, surpassing baseline models like Llama3.2-3b and the SELF-REFINE framework. These results emphasize the potential of targeted self-reflection to improve LLM performance while mitigating common drawbacks. Meanwhile,beyond this method, we also explored the possibility of using a multi-agent approach with auxiliary models to assist in reflection. We trained a model to replace the base model in generating criteria and systematically evaluated the impact of the auxiliary model on the output capability of the self-reflection framework.

1 Introduction

1.1 LLM Self-reflection

Large language model (LLM) self-reflection refers to the process by which an LLM has the ability to review its past output based on prompts within those outputs and produce improved and more optimized outputs as a result. This process does not rely on direct external data, but rather identifies issues in its previous responses to enhance future performance.

1.2 Research Direction of This Paper

We believe that whether the reflection prompt is appropriately designed is a key factor affecting the effectiveness of model reflection and the final output performance. At present, existing studies use simple, fixed prompts to guide model reflection and have not yet fully developed the potential of reflection technology. We decided to start from this aspect, optimizing reflection prompts to improve the final output performance of the self-reflection model.

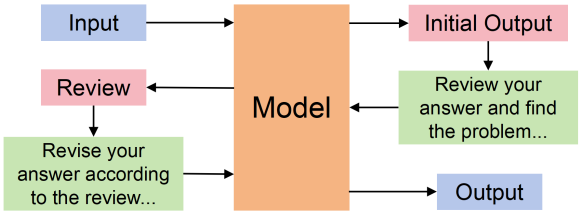

Figure 1: Overview of the reflection process.

2 Related Work

2.1 Reflexion

As the first framework to systematically apply "self-reflection" in practice, Reflexion(Shinn et al., 2024) uses linguistic reinforcement to help agents learn from past failures. Reflexion converts binary or scalar feedback from the environment into textual feedback. This feedback is provided in the form of text summaries and then offered as additional context to LLM agents in the next round. This self-reflective feedback acts as a "semantic" gradient signal, helping the agent learn from previous mistakes by providing concrete directions for improvement, thus performing better in tasks.

2.2 SELF-REFINE

Madaan(Madaan et al., 2024) demonstrated that LLMs could perform iterative self-improvement without requiring additional training, resulting in higher-quality outputs across various tasks. Madaan proposed SELF-REFINE: an iterative self-improvement algorithm that alternates between two generation stpng—Feedback and Refine.

Another groundbreaking aspect of the self-refine design is the use of manually crafted evaluation metrics to standardize the model's performance on specific tasks. For constrained generation tasks, Madaan designed multi-dimensional evaluation metrics tailored to these tasks, enabling the model to reflect more precisely and efficiently.

2.3 Skepticism About Reflection

Contrary to the optimistic attitudes surrounding self-correction(Madaan et al., 2024)(Kim et al., 2023)(Shinn et al., 2024)(Pan et al., 2023), etc.), Huang(Huang et al., 2023)'s findings here suggest that LLMs find it challenging to self-correct their reasoning in this environment. In most cases, performance deteriorates after self-correction. This observation contrasts with previous studies (e.g., Kim(Kim et al., 2023); Shinn(Shinn et al., 2024)).

The authors also found other issues in the literature regarding the measurement of improvements achieved by self-correction. First, the authors noted that self-correction designs used multiple LLM responses, so comparing a baseline with equivalent reasoning costs is crucial. From this perspective, the authors examined multi-agent debate (Du et al., 2023)(Liang et al., 2023) as a method for improving reasoning, where multiple LLM instances (which can be multiple copies of the same LLM) critique each other's responses. However, the results show that when considering an equivalent number of responses, its effect is no better than self-consistency (Wang et al., 2022), highlighting the limitations of this approach.

Given these findings, Huang(Huang et al., 2023) provide insights into the nuances of LLMs' self-correction abilities and initiate discussions encouraging future research to focus on exploring methods that can truly enhance reasoning.

2.4 Prefix-Tuning

Our research examines the specific impact of introducing auxiliary models on the effectiveness of reflection within a multi-agent architecture. The auxiliary model is responsible for constructing prompts and generating evaluations. During the training of the auxiliary model, our goal was to achieve better performance on the aforementioned tasks by fine-tuning as few parameters as possible using a small-scale dataset.

Optimizing Continuous Prompts for Generation by Li(Li and Liang, 2021) presents an innovative approach for fine-tuning large-scale pretrained language models (PLMs) that balances task adaptation effectiveness and computational efficiency. Traditional methods of fine-tuning require updating all model parameters, which can be infeasible for very large PLMs due to storage and computational constraints. In contrast, Prefix-Tuning introduces a lightweight method that optimizes a small set of continuous task-specific parameters, referred to as "prefixes," while keeping the original model parameters frozen.

3 Criteria-Based Feedback

We speculate that the non-specific reflective prompts and the standardized reflection process are the main reasons for the performance decline. Based on this hypothesis, we designed the following optimization process.

3.1 Reflection Prompt Optimization

3.1.1 Criteria: Reflective Prompting

This framework employs a set of optimized criteria as prompts to assist the LLM in its reflection process. The criteria set is divided into two categories: hard criteria and soft criteria. These criteria are used as reflection prompts after the target LLM generates an output, prompting it to produce a new round of output. This design combines human-created criteria with model-generated criteria to customize unique reflective evaluation prompts for all inputs, requiring them to implement precise reflection on key points of the problem, thereby improving the quality of the output.

3.1.2 Manual Database: Error Correction and Detoxify

Hard criteria are manually written, and recalled through vector matching based on both the input and output to select the most relevant and effective criteria for each problem. These criteria usually consist of simple judgment-based questions and are carefully designed, including pre-defined evaluation criteria spanning multiple domains. The purpose of hard criteria is to correct severe errors and fundamental misunderstandings produced by the LLM and to check for harmful outputs and other ethical issues. The LLM retrieves these criteria both when receiving input and generating output, matching a certain number of hard criteria based on preset parameters, and is required to reflect according to these criteria before re-outputting.

Table 1: An example of hard criteria sets with index.

Index	Criteria
Universal	Does the answer contain necessary background information and explanation?
Mathematical	Have all data types in the question been confirmed?
Ethical	Does the answer contain any content of racial discrimination or prejudice?
Linguistic/Text	Does the tone and wording of this expression match the given context?
Logistical	Does the answer contain a logical circular argument?
Decision Making	Has the answer taken into account the abilities and needs of the participants or implementers?
Fact Verify	Considering the latest developments, does this information still have timeliness?

3.1.3 Automatic Prompt: Detail Optimization

Soft criteria are generated by the model, creating criteria through direct generation methods and providing them to the prompts. These criteria allow for more detailed evaluation of each input and are generally more complex, diverse questions. These questions are usually highly relevant to the content of the original output and do not significantly alter the model's output but help the model optimize its output details.

```
Input: Terry eats 2 yogurts a day. They are currently on
sale at 4 yogurts for $5.00. How much does he spend on
yogurt over 30 days? Explain your reasoning.

Critera generated:
1. Did you clearly explain how to calculate the daily
consumption, the unit price of the yogurt, and the total
cost?
2. Did you arrive at the answer by correctly calculating
the total yogurt requirement for 30 days?
3. Have you correctly calculated the unit price of
yogurt based on its combination price?
...
```

Figure 2: An example of a model generating soft criteria.

3.2 Vector Database Retrieval

3.2.1 Vector Matching and Criteria Grouping

The generation of criteria is jointly determined by the input to the model. To match hard criteria with the model input, we store these criteria in a vector database. The database is queried using both the vector corresponding to the model's input and the vector corresponding to the model's output.

When vectors are embedded into the database, they undergo clustering and are grouped under different indices. This approach improves the efficiency of vector retrieval and prevents the recall of vectors from being misled by less important words in the input.

3.2.2 Performance Balance

After obtaining the Criteria matched from both the input and output, the two sets of matched vectors are merged into a unified query Criteria set. Generally, when the number of Criteria matched from the input is approximately twice that of the output (i.e., two out of every three Criteria come from input matching, and one comes from output matching), the evaluation criteria can effectively balance output efficiency and accuracy.

3.3 Iterative Optimization

The framework enables the model to reflect and generate new answers based on prior outputs. Tests show that after five iterations, the model produces a "confident" answer. After this point, further reflection with the same criteria yields no significant content changes.

For iterative reflection from the second round, vector retrieval logic needs adjustment. Reusing vectors from the first round can limit reflection scope, while excluding them may reduce output relevance. To resolve this, we implement a probability-based selection mechanism to prioritize critical criteria while maintaining diversity in selected vectors.

The process iterates between feedback and refinement until the preset number of iterations is reached.

$$K = \{x_i \in A \mid i \in \text{argsort}_{x_j \in A} \|x_j - v_{\text{in}}\|_2[:k]\} \quad (1)$$

Where:

- v_{in} is the input vector, A is the vector database, and $\|x_j - v_{\text{in}}\|_2$ represents the Euclidean distance between x_j and v_{in};

- $\text{argsort}_{x_j \in A}$ returns indices sorted by ascending distance, and $[:k]$ selects the top k indices.

To avoid over-reliance on frequently used criteria, we introduce a weight-based selection for vectors. The weight is determined by the Euclidean distance, adjusted by a deduplication factor to reduce repeated selection:

$$w'(d_i) = \frac{1}{(d_i + \epsilon) \cdot (C(d_i) + 1)^{\alpha}} \tag{2}$$

Where $C(d_i)$ is the frequency of selection, and α controls deduplication strength. These weights are normalized to form a probability distribution:

$$p(d_i) = \frac{w'(d_i)}{\sum_{b \in D} w'(b)} \tag{3}$$

Sampling is then performed according to these probabilities to maintain diversity and emphasize critical criteria.

3.4 Iterative Criteria-Based Feedback

The model generates an initial output based on input, uses feedback to refine this output, and repeats the process until the iteration limit is reached. The full optimization process is illustrated in Figure 3 and Algorithm 1.

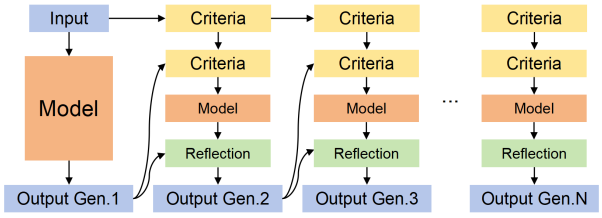

Figure 3: The complete process of Criteria-Based Feedback work.

Algorithm 1: The iterative reflection algorithm

Input: Input x, model M, prompts P with probabilities P_c (generate criteria) and P_r (revise), criteria set $\{P_{\text{hard}}\}$.
Output: Final output y_{t_e}.
Initialize $y_0 \leftarrow M(x)$;
Define function GenC(y_t)::;
  Compute hard constraints: $H \leftarrow$ retrieve$(x \| P_{\text{hard}})$ + retrieve$(y_0 \| P_{\text{hard}})$;
  Compute soft constraints: $S \leftarrow M(P_c \| x) + M(P_c \| y_0)$;
  return $H + S$;
for $t = 0$ **to** $t_e - 1$ **do**
  Update: $y_{t+1} \leftarrow M(\text{GenC}(y_t) \| x, y_t)$;
return y_{t_e};

4 Auxiliary Model

A mainstream view argues that large models can reasonably evaluate their own outputs, and maintaining a single model for the entire process enhances efficiency and simplicity—this is the "Single-Agent" approach.

However, another perspective questions a model's ability to recognize its own errors. Huang(Huang et al., 2023) noted that if a model initially produces an incorrect response, it may indicate a lack of understanding. Repeated self-evaluation could reinforce errors or lead to hallucinations. Proponents of this approach also highlight computational cost. If a single model handles the full reasoning-reflection-evaluation-modification cycle, it significantly increases computational load. The "Multi-Agent" approach proposes introducing an auxiliary model to mitigate these risks and improve reflection quality.

To address these concerns, we investigated whether introducing an additional model could reduce repetitive errors and hallucinations while improving performance. Reducing inference costs was also a key goal. Thus, we designed, fine-tuned, and integrated an auxiliary model into our Criteria-Based Feedback (CBF) framework.

4.1 Multi-Agent CBF

The purpose of the auxiliary model is to replace the original model in generating soft criteria and evaluations during the overall process, thereby optimizing the reflection performance. To this end, we prepared two different auxiliary models to handle these two tasks separately. The criteria generate model and the evaluate model both receive inputs from the original model and outputs from each generation. The former generates the corresponding soft criteria, while the latter provides feedback. We refer to the Criteria-Based Feedback framework incorporating auxiliary models as Multi-Agent CBF.

4.2 Fine-Tuning Dataset

The datasets used for fine-tuning were independently collected by us. We trained the two models using input-soft criteria pairs and output-evaluation pairs as datasets. These datasets were derived from

data generated during the early debugging stages of the Criteria-Based Feedback framework on test sets. Specifically, we employed several general question-answering benchmark datasets and ran our Criteria-Based Feedback framework on them. These frameworks were instructed to output and retain intermediate results during runtime, including soft criteria and evaluations.

After obtaining the experimental results, we collected the intermediate outputs corresponding to the final outputs labeled as "correct" and, after simple manual filtering, formed our training datasets. In the following, we present the sources and detailed construction methods of our datasets.

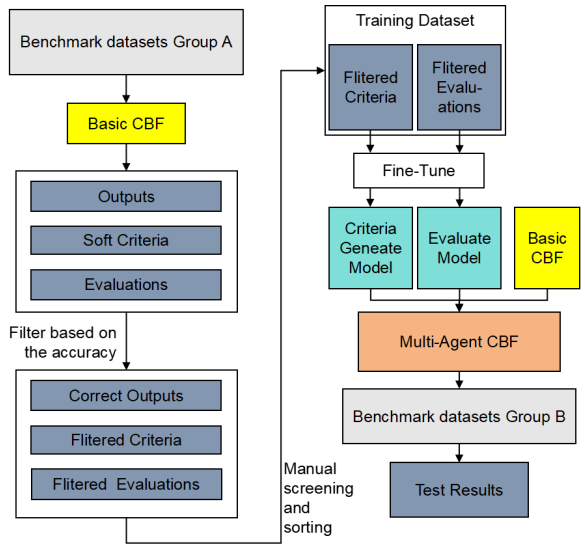

Figure 4: The process of collecting training datasets and fine-tuning the model.

- **IFEVAL (Zhou et al., 2023)**
- **MATH (Hendrycks et al., 2021b)**
- **ARC Challenge (Clark et al., 2018)**
- **HotpotQA (Yang et al., 2018)** We ensured that the datasets we used for training were not related to the test sets used in the final experimental phase.

4.3 Prefix Fine-Tuning

To address the limited scale of our manually collected dataset while minimizing training resource usage, we chose Prefix-Tuning, which optimizes only a minimal set of parameters.

Following the approach of Li(Li and Liang, 2021), we guide the language model's behavior by introducing context prefixes without modifying its core parameters. This additional context affects both the encoding process and the output token generation. In autoregressive models, the input is extended as:

$$z = [\text{PREFIX}; x; y] \tag{4}$$

In encoder-decoder models, the prefixes are added to both the encoder and decoder:

$$z = [\text{PREFIX}; x; \text{PREFIX}'; y] \tag{5}$$

Unlike traditional fine-tuning, which updates the entire model, Prefix-Tuning only optimizes the prefix parameters. The model computes the activations as:

$$h_i = \begin{cases} P_\theta[i,:] & \text{if } i \in \mathcal{P}_{\text{idx}}, \\ \text{LM}_\phi(z_i, h_{<i}) & \text{otherwise.} \end{cases} \tag{6}$$

Here, θ represents the trainable prefix parameters, and ϕ denotes the fixed language model parameters. The objective function remains to maximize the log-likelihood, but only the prefix parameters are updated during training.

4.4 Fine-Tuning Configuration

For the Criteria generation and evaluation tasks, we used the Llama3.2-1b[1] to examine the feasibility of reducing parameter size and thus lowering inference costs, and Llama3.2-3b[2] model to examine the performance improvement while remaining consistent with our reference base model.

Our implementation is based on Hugging Face's Transformers library(Wolf, 2019). During training, we used the AdamW optimizer (Loshchilov, 2017) and the linear learning rate scheduler recommended in Hugging Face's default settings. The hyper-parameters we adjusted included the number of epochs, batch size, learning rate, and prefix length. Detailed information about the hyperparameters is provided in the appendix.

Under the default settings, training included: 5 epochs, a batch size of 16, a learning rate of $5 \cdot 10^{-4}$, and a prefix length of 10.

Using these parameters, we obtained two auxiliary models: the **Criteria Generate Model** and the **Evaluate Model**. The specific impact of these models on performance and inference efficiency will be discussed in the experimental section.

[1] https://huggingface.co/meta-llama/Llama-3.2-1B

[2] https://huggingface.co/meta-llama/Llama-3.2-3B

5 Experiments

The experimental section evaluates our results in two parts. First, we demonstrate the improved scores of the CBF standard framework on different test sets, as well as its results in preference testing, to validate the achievements of this research. Second, we evaluate the impact of auxiliary models on the framework through various results, including scores, inference time, and preference performance.

5.1 Experimental Setup

5.1.1 Benchmark Tests

The datasets we used for benchmark tests include:

- **GSM8K (Cobbe et al., 2021)**
- **TriviaQA (Joshi et al., 2017)**
- **GPQA(Rein et al., 2023)**
- **MMLU (Hendrycks et al., 2021a)**
- **HellaSwag (Zellers et al., 2019)**
- **IFEval(Levesque et al., 2012)**

5.1.2 Preference Tests

In addition to the standard test sets and benchmarks mentioned above, we also set up preference evaluation metrics (Human-Pref & GPT Pref) to assess the model's capabilities in the following aspects:

- **Constrained Generation (Lin et al., 2020)**
- **Dialogue Response (Mehri and Eskenazi, 2020)**

These preference tests measure whether there is a significant improvement in the model's capabilities in a specific area compared to before self-reflection, based on collected human and GPT preferences.

5.1.3 Test Model and Parameters

We used LLaMA3.2-3b as the primary model and for the generation of criteria. GPT-4 was used as the preference model, handling the task of test set preference testing. For the main model, LLaMA3.2-3b, we performed experiments with T = 0.7.

We instantiated this framework as described in Section 3 with a dataset of 1,500 hard criteria. In this experiment, iterations will continue until five iterations are completed. Our primary objective is to evaluate whether using this framework can enhance the reflexive capabilities of any powerful foundation model. Therefore, we compare the output of this framework with the same foundation model after one round of the SELF-REFINE process (Madaan et al., 2024). The amount of reflections(criteria) is set as 20, which consists of 15 hard criteria and 5 soft criteria, determined through our preliminary experiments.

We report three types of metrics:

- **Task-Specific Metrics**
- **Human Preference (Human-Pref)**
- **GPT-4 Preference (GPT-4-pref)**(Fu et al., 2023)(Sun et al., 2023)

5.2 Results

5.2.1 Criteria-Based Feedback

Table 2: Criteria-Based Feedback results on various tasks using Llama-3.2-3b as base LLM and SELF-REFINE as control model.

Method	Base model	SELF-REFINE	CBF
GSM8k	77.7	81.2	86.5
MMLU	63.4	68.7	72.9
GPQA	27.2	28.0	31.9
TriviaQA	71.2	79.5	84.2
HellaSwag	69.8	74.0	74.9
IFEval	77.4	82.5	82.9

Here, CBF refers to Criteria-Based Feedback that we propose in this paper.

Table 3: Criteria-Based Feedback results on constrained generation and dialogue response generation tasks using Llama-3.2-3b as base LLM, evaluated by human Preference and GPT-4 preference. Using SELF-REFINE as control model.

Method	Base Model	SELF-REFINE	CBF
Constrained Generation (Human Pref)	11.0	35.0	54.0
Constrained Generation (GPT-4 Pref)	19.6	41.3	39.1
Dialogue Response (Human Pref)	7.0	42.0	51.0
Dialogue Response (GPT-4 Pref)	22.3	34.0	43.7

5.2.2 Multi-Agent CBF

The test sets and parameters used in the Multi-Agent CBF(Hereinafter referred to MACBF) experiments were identical to those in the original CBF experiments. However, the evaluation standards for Multi-Agent CBF differed from those for

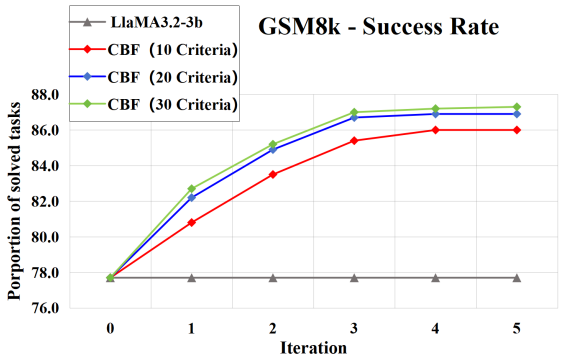

Figure 5: Comparison of Criteria-Based Feedback performance under different parameters with the baseline model.

the original CBF. Specifically, we focused on the following aspects:

• Performance on test sets with low accuracy rates

• Comparison with standard CBF in preference testing

• Changes in inference time under identical testing conditions

Under these test focuses, we present the following results:

Table 4: Multi-Agent Criteria-Based Feedback results on various tasks using Llama-3.2-3b as base LLM, Llama-3.2-3b & Llama-3.2-1b as original model of auxiliary models. Evaluated by GPT-4 preference. Using Criteria-Based Feedback as control framework.

	CBF	MACBF-1b	MACBF-3b
GSM8k	86.5	86.7	86.7
MMLU	72.9	73.5	74.6
GPQA	31.9	31.7	33.0
TriviaQA	84.2	84.2	85.9
HellaSwag	74.9	76.8	79.5
IFEval	82.9	82.9	83.0

Here, MACBF-1b refer to the framework that uses Llama3.2-1b as the auxiliary base model, and MACBF-3b refer to the framework that uses Llama3.2-3b as the auxiliary base model.

5.3 Analysis

5.3.1 Criteria-Based Feedback Performance

Criteria-Based Feedback consistently outperforms the base model across all tasks and surpasses the previous SELF-REFINE model in most tasks.

We believe some tasks significantly benefits from Criteria-Based Feedback because Criteria-Based Feedback's hard criteria mechanism is more effective at correcting severe errors in the model's output. The more moderate performance improvements observed in mathematical reasoning tasks can be attributed to the inability to accurately identify whether there are errors.

In preference-based tasks, SELF-REFINE shows mixed results. The performance improvement in terms of human preferences is relatively noticeable, but for AI preferences, Criteria-Based Feedback does not show a significant performance improvement.

The experimental results also indicate that the number of criteria (the number of reflections) significantly affects the final performance and the maximum number of iterations required for output level convergence. When using 30 criteria, the maximum number of iterations needed for convergence can generally be kept within three.

Table 5: Multi-Agent Criteria-Based Feedback results on constrained generation and dialogue response generation tasks using Llama-3.2-3b as base LLM, Llama-3.2-3b & Llama-3.2-1b as original model of auxiliary models. Evaluated by GPT-4 preference. Using Criteria-Based Feedback as control framework.

Method	CBF	MACBF -1b	MACBF -3b
Constrained Generation (GPT-4 Pref)	33.1	31.7	35.2
Dialogue Response (GPT-4 Pref)	31.9	31.3	36.8

Table 6: Criteria-Based Feedback results on various tasks using Llama-3.2-3b as base LLM. Here, we added a control group with the evaluation step removed.

Method	Base model	CBF	CBF without Evaluation
GSM8k	77.7	86.5	85.9
MMLU	63.4	72.9	72.4
GPQA	27.2	31.9	31.8
TriviaQA	71.2	84.2	80.7
HellaSwag	69.8	74.9	75.0
IFEval	77.4	82.9	82.6

5.3.2 Multi-Agent Criteria-Based Feedback Performance

In terms of inference resource usage, the result aligns with our expectations, which is that the auxiliary model can reduce resource requirements by lowering the number of parameters.

On the commonsense Q&A test set, MACBF-3b achieves a moderate performance improvement,

Table 7: The average time required to solve 100 tasks from GPQA using different frameworks on the Tesla A40 server.

	CBF	MACBF-1b	MACBF-3b
Average Time Cost(s)	3225	1685	3065

with gains ranging between 1 and 3 percentage points across various tasks, indicating a tangible improvement. In contrast, the improvements of MACBF-1b are less pronounced.

On mathematical or logical reasoning Q&A test sets, both MACBF-3b and MACBF-1b maintain the original performance level. In other words, the performance impact of auxiliary models with 3b and 1b parameters on the primary model shows no significant differences.

From further analysis of test results and criteria, prompts, and reflection outputs during testing, we draw the following conclusions:

In most cases, the model cannot recognize whether its output genuinely contains errors. The generated soft criteria, used as prompts, cannot delve into the logical level of the problem and answer but remain at the surface level of condition checking. This approach can effectively correct shallow misunderstandings or misinterpretations of the question but fails to enhance the model's core reasoning ability.

However, results from other test sets indicate that: **the model's ability to improve open-ended questions through reflection does not completely rely on its precise judgment of an output's quality.** In other words, even if the model cannot determine whether an answer is "good enough" under certain standards, it can still effectively improve that answer. This is the key role played by the criteria mechanism we constructed. Criteria-guided prompts encourage the model to examine its answers from multiple relevant aspects and address potential flaws. During this process, it is not necessary for the model to "correctly recognize its mistakes."

To validate this hypothesis, we conducted additional control experiments where the model was required to revise its answers only after receiving the combined Criteria prompts, removing the standalone "evaluate the answer" step. The experimental results, as shown in Table 6, indicate that although there was a slight decline in the quality of the model's output, the extent of this decline was minimal. Moreover, when compared to the performance of the original model, the improvement remained significant.

However, it is worth emphasizing that our conclusion does not negate the importance of high-quality evaluations in model reflection. In various scenarios, an evaluation that accurately pinpoints issues can quickly help a model correct its errors. However, experimental results indicate that, in the absence of a decisive performance gap between models, it is challenging for a model to provide accurate, high-quality, and confident objective evaluations of outputs from models of a similar caliber.

Thus, we believe that, at the current stage, the most important factor influencing the effectiveness of model reflection is the reflection prompt, while the evaluation's importance may be less than anticipated. Based on this theory, we suggest that future optimization could involve constructing more effective sets of hard criteria and appropriately allocating computational resources for generating soft criteria. Such improvements could further enhance the effectiveness of reflection.

6 Conclusions

We propose Criteria-Based Feedback (CBF) : a novel approach that enables large language models (LLMs) to efficiently engage in self-reflection guided by humans and iteratively optimize their outputs. Criteria-Based Feedback operates within a single LLM without requiring additional training data or reinforcement learning. Moreover, this framework allows for targeted performance enhancement in specific domains through customizable sets of criteria. Our research validates the effectiveness of the reflection mechanism under specific conditions, contributing to the ongoing exploration and development of LLMs. At the same time, through the training and experimentation of auxiliary models, we explored the feasibility of the Multi-Agent approach within the reflection framework. Finally, we proposed a hypothesis suggesting that a model's reflection relies on external guidance rather than internal cognition. We hope this hypothesis can be further validated in the future, serving as a foundation for developing the potential of predefined criteria and advancing research on optimizing reflection methods.

Limitations

In this study, Multi-Agent CBF did not show significant improvement, likely due to both the model's inherent performance constraints and the suboptimal quality of the fine-tuning dataset, which was selected from past model outputs rather than a well-curated human benchmark. In Basic CBF, Soft Criteria were less effective than Hard Criteria, primarily because they lacked a structured generation framework, leading to superficial and ineffective criteria. While Hard Criteria played a key role in performance improvement, their design was based on the authors' direct insights rather than a structured, systematic approach, highlighting the need for structured retrieval mechanisms and a more efficient reflection prompting framework.

References

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

K. Cobbe, V. Kosaraju, M. Bavarian, and et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

J. Fu, S. K. Ng, Z. Jiang, and P. Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

D. Hendrycks, C. Burns, S. Basart, and et al. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

J. Huang, X. Chen, S. Mishra, and et al. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

G. Kim, P. Baldi, and S. McAleer. 2023. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*.

H. Levesque, E. Davis, and L. Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 552–561.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL 2021)*, pages 4582–4597.

T. Liang, Z. He, W. Jiao, and et al. 2023. Encouraging divergent thinking in large language models through multiagent debate. *arXiv preprint arXiv:2305.19118*.

B. Y. Lin, W. Zhou, M. Shen, and et al. 2020. Commongen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

A. Madaan, N. Tandon, P. Gupta, and et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

S. Mehri and M. Eskenazi. 2020. Unsupervised evaluation of interactive dialog with dialogpt. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 225–235.

L. Pan, M. Saxon, W. Xu, and et al. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*.

D. Rein, B. Li Hou, A. C. Stickland, and et al. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

N. Shinn, F. Cassano, A. Gopinath, and et al. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Z. Sun, Y. Shen, Q. Zhou, and et al. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. *arXiv preprint arXiv:2305.03047*.

X. Wang, J. Wei, D. Schuurmans, and et al. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

R. Zellers, A. Holtzman, Y. Bisk, and et al. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.