Unleashing the True Potential of LLMs: A Feedback-Triggered Self-Correction with Long-Term Multipath Decoding

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have achieved remarkable performance across diverse tasks, yet their susceptibility to generating hallucinated content during inference remains a critical unsolved challenge. While self-correction methods offer potential solutions, their effectiveness is hindered by two inherent limitations: (1) the absence of reliable guidance signals for error localization, and (2) the restricted reasoning depth imposed by conventional next- token decoding paradigms. To address these issues, we propose Feedback-Triggered Regeneration (FTR), a novel framework that synergizes user feedback with enhanced decoding dynamics. Specifically, FTR activates response regeneration only upon receiving negative user feedback, thereby circumventing error propagation from faulty self-assessment while preserving originally correct outputs. Furthermore, we introduce Long-Term Multipath (LTM) decoding, which enables systematic exploration of multiple reasoning trajectories through delayed sequence evaluation, effectively overcoming the myopic decision-making characteristic of standard next-token prediction. Extensive experiments on mathematical reasoning and code generation benchmarks demonstrate that our framework achieves consistent and significant improvements over state-of-the-art prompt-based self-correction methods.

1 Introduction

Large Language Models (LLMs) have achieved remarkable performance across a variety of tasks, including text generation, question answering, and code synthesis (Achiam et al., 2023; Touvron et al., 2023; Guo et al., 2025). Despite these achievements, LLMs face significant challenges, particularly the issue of hallucinations. Hallucinations refer to the generation of plausible but factually incorrect information, which remains a widely acknowledged problem in LLM inference (Yao et al., 2023; Liu et al., 2024a).

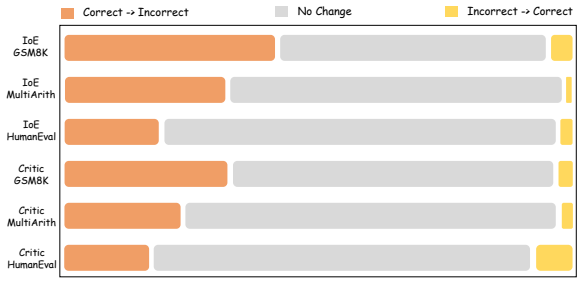

Figure 1: The percentage distribution of answer changes induced by self-correction using IoE Prompts (Li et al., 2024) and Critic Prompts (Huang et al., 2024), with experiments conducted on the Llama3-Instruct-3B.

To address this issue, self-correction mechanisms have emerged as a promising research direction, enabling LLMs to improve their outputs based on previous responses (Ji et al., 2023; Madaan et al., 2024). These approaches empower LLMs to refine their outputs through introspective reasoning, typically facilitated via carefully designed prompts (Kim et al., 2024; Li et al., 2024; Chen et al., 2024a; Huang et al., 2024). The self-correction pipeline generally consists of two phases: first, generating an initial answer using standard LLM inference; and second, prompting the LLM to assess and revise the initial output.

Nevertheless, the effectiveness of prompt-based methods remains contentious. As illustrated in Figure 1, experiments on three typical reasoning datasets demonstrate that two state-of-the-art prompt-based self-correction methods (Huang et al., 2024; Li et al., 2024) not only frequently convert correct answers into incorrect ones but also struggle to revise incorrect answers into correct responses. In this study, we argue that this phenomenon stems from two key challenges inherent in the prompt-based self-correction process:

- **C1. Lack of Effective Guidance Signals**: The prompt-based self-correction process relies on the LLM itself to evaluate the cor-

rectness of its previous answers. However, due to the absence of explicit guidance signals, the LLM may fail to accurately determine which parts require revision, leading to unnecessary self-corrections (i.e., modifying correct answers to incorrect ones). Additionally, given the sensitivity of LLMs to input, biased prompts may cause incorrect alignment and mislead the LLM into making inaccurate judgments (Huang et al., 2024).

- **C2. Shallow Decoding Limits Deep Reasoning**: Self-correction of erroneous results by LLMs requires deeper thinking and reasoning. However, most current methods follow the next-token prediction paradigm, which focuses only on single-step predictions during the decoding process. The correctness of the answer depends on a comprehensive evaluation of the entire output answer sequence. This short-term, step-oriented decoding process limits the LLM's ability to engage in deeper reasoning, thereby hindering its capacity to generate improved responses during the self-correction process.

To address **C1**, we propose a feedback-triggered self-correction framework, named Feedback-Triggered Regeneration (FTR), that leverages user feedback to guide the LLM's reasoning and self-correction process. In real-world scenarios, users naturally provide feedback on the responses generated by LLMs, especially when they are dissatisfied with the LLMs' answers. This feedback serves as a direct indicator of whether the LLM's output requires deeper reasoning and revision. Specifically, when user feedback is negative, the LLM regenerates the output based solely on the initial input, thereby bypassing the issues associated with prompt-based methods. This approach eliminates the need for the LLM to self-assess its previous answers, thereby preventing the unnecessary alteration of correct answers. Moreover, since user feedback is readily available in human-LLM interactions, FTR is highly adaptable to real-world applications and can be generalized across various tasks.

To address **C2**, we strengthen FTR by integrating a novel Long-Term Multipath (LTM) decoding strategy, which is designed to promote deeper reasoning in LLMs. Specifically, LTM explores multiple decoding paths at each step and evaluates their long-term performance, thereby expanding the search space available to the LLM. This approach contrasts with the conventional next-token prediction paradigm, which focuses only on single-step predictions and may fail to identify sequences with higher long-term quality. By considering multiple paths and their long-term impact, LTM enables the LLM to recompose its responses more effectively, leading to more accurate and coherent outputs. This strategy is applied during the regeneration stage of self-correction, allowing the LLM to generate improved responses by leveraging deeper reasoning.

Overall, we propose a novel user feedback-triggered self-correction framework that integrates user feedback with the advanced LTM decoding strategy. To validate the effectiveness of our framework, we conduct a series of experiments comparing it with SOTA prompt-based methods using open-source backend LLMs. These experiments focused on challenging mathematical and coding datasets, where our framework consistently achieves superior performance. In summary, our main contributions are as follows:

1. We introduce a novel self-correction framework that leverages user feedback as a regeneration signal, thereby preventing unnecessary self-corrections and improving the quality of LLM outputs.

2. We propose a novel decoding method that evaluates the long-term performance of multiple reasoning paths, thereby enhancing the accuracy and coherence of generated responses.

3. We demonstrate the superiority of our method through extensive experiments on various datasets and backend LLMs, showing consistent improvements over existing prompt-based approaches.

2 Preliminaries

In this section, we introduce the notation used throughout this paper and provide an overview of the commonly employed two-stage self-correction framework.

2.1 Notation Definition

Let $x = (x_0, x_1, \ldots, x_n)$ denote an input sequence, and $y = (y_0, y_1, \ldots, y_m)$ represent the corresponding LLM response, where $y = \mathcal{M}(x)$. Here, $\mathcal{M}$ denotes a typical autoregressive language model.

In this context, the response is generated sequentially during the decoding process. At the i-th step of the inference process, we define the probability of the current output sequence $s_i = (y_0, y_1, \ldots, y_i)$ as $P(s_i)$, which is calculated as the product of the likelihoods of the first i tokens:

$$P(s_i) = P(y_0|x) \prod_{k=1}^{i} P(y_k|y_{0:k-1}, x) \quad (1)$$

The perplexity (PPL) value of the sequence at step i is then defined as:

$$\text{PPL}_i = P(s_i)^{-\frac{1}{i}} \quad (2)$$

In this work, PPL is employed as a metric to assess the quality of a sequence during the decoding process.

2.2 Two-Stage Framework for Self-Correction

The framework of most prompt-based self-correction methods can be divided into two stages, as depicted in Figure 2 (a):

- **Stage 1**: An initial input x is provided to the LLM to generate an initial response $y_{init} = \mathcal{M}(x)$.
- **Stage 2**: An independent correction prompt p_{cor} is then given, prompting the LLM to reflect on its generated response. This enables the LLM to refine its answer, regenerating the refined output $y_{cor} = \mathcal{M}(x, y_{init}, p_{cor})$.

In Figure 2 (b), we also present our proposed FTR self-correction framework for intuitive comparison. Specifically, we have made improvements from two perspectives: 1) incorporating user feedback as a guiding signal to prompt LLM to regenerate responses based on the initial input when necessary; 2) adopting LTM decoding to enhance the LLM's ability for deeper reasoning in order to address more complex error response scenarios.

3 Methodology

In this section, we first introduce our proposed feedback-triggered regeneration framework, followed by a detailed exploration of LTM decoding.

3.1 Feedback-Triggered Regeneration

In general, the correction prompt p_{cor} provides no information about the correctness of the initial response y_{init}. Additionally, LLMs often lack the ability to independently assess the correctness of their own responses, as highlighted in previous works (Huang et al., 2024; Madaan et al., 2024). These limitations may lead to erroneous decisions by the LLM, as demonstrated in Figure 3 (a).

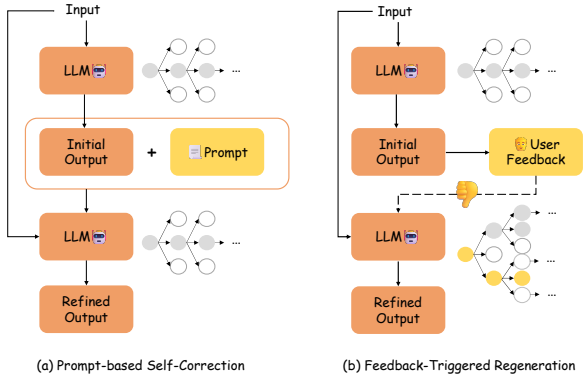

Figure 2: (a) Framework of the prompt-based self-correction approach. (b) Framework of our feedback-triggered self-correction approach.

Moreover, when the LLM is well-aligned and receives a carefully crafted initial input, the first response should ideally be optimal, given the selected decoding algorithm. However, introducing an additional prompt may cause the LLM to generate a response that is more aligned with the combined input, rather than improving accuracy (Huang et al., 2024). This highlights another issue: the LLM's sensitivity to prompt design, which may ultimately degrade performance (Xu et al., 2024; Liu et al., 2024b). For instance, even when correctness feedback is included in p_{cor} , as shown in Figure 3 (b), the LLM may still fail to refine the previous answer accurately.

To avoid the negative impact of self-correction prompts, we propose an enhanced two-stage FTR self-correction framework:

- **Stage 1**: Similarly, provide the initial input x to the LLM to generate the initial response $y_{init} = \mathcal{M}(x)$.
- **Stage 2**: If user feedback indicates that the LLM's output y_{init} is problematic, the original prompt x and an advanced decoding strategy, LTM, are employed to regenerate the output. Otherwise, no further action is taken.

$$y_{cor} = \mathcal{M}'(x) \quad (3)$$

Here, $\mathcal{M}'$ denotes the LLM with an alternative LTM decoding strategy, wequipped with the LTM

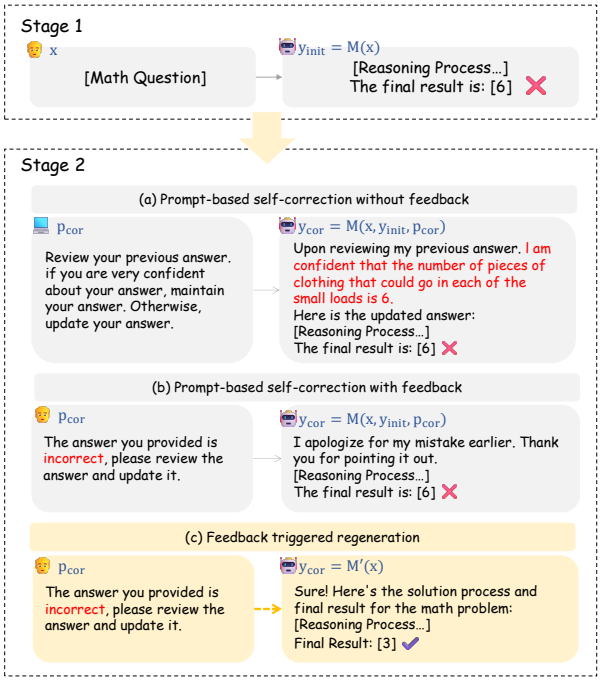

Figure 3: Comparison of different self-correction methods. (a) Self-assessment and update; (b) Revision with user feedback prompt; (c) Regeneration triggered by user feedback.

decoding strategy, which is described in the following section. Note that the second stage of FTR uses only the original input x without introducing additional prompts. Human feedback serves solely as an indicator to trigger regeneration, as illustrated in Figure 2 (b) and Figure 3 (c). This approach prevents the LLM output from being degraded by potentially biased prompts.

3.2 Long-Term Multipath Decoding

After receiving negative feedback from users, the FTR framework initiates the second stage of regeneration. Given the complexity of error scenarios, the LLM requires more in-depth reasoning to generate a higher quality response than the initial incorrect one. To address this, we propose LTM decoding strategy for the second stage of self-correction. Unlike traditional paradigm that focus solely on the score of the next token, LTM considers the performance from the sequence perspective, thereby alleviating the short sightedness of conventional decoding approaches and enabling more in-depth reasoning for the LLM. This approach improves decoding from two aspects: (1) **Multipath Exploration**: Instead of exploring a single path, token selection is performed using a "tree" structure rather than the traditional "chain" structure.

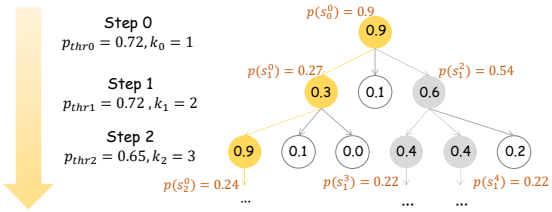

Figure 4: Illustration of LTM decoding strategies ($V = 3$), where black numbers in circles are token likelihood and red ones indicate sequence likelihood.

This allows the LLM to explore multiple potential sequences simultaneously, as illustrated in Figure 4. (2) **Sequence Evaluation**: We use PPL as a metric to evaluate the quality of all potential sequences, retaining the top k_i sequences at step i. This selection is dynamically adjusted according to the PPL distribution at each decoding step.

The detailed implementation of LTM is described below, including the method for determining k_i at each step. An illustrative example is provided in Figure 4 for clarity. **Firstly**, the probabilities of all possible sequences are computed. Let k_{i-1} denote the number of candidates retained at the $(i-1)$-th step, and let V represent the size of the LLM's vocabulary. Accordingly, there are $k_{i-1} \times V$ candidate sequences. **Secondly**, the top-k_i sequences are selected from the $k_{i-1} \times V$ candidates. These candidates are sorted based on their probabilities $P(s_i^j)$, where $j \in [0, k_{i-1} \times V - 1]$ denotes the index of each candidate. The cumulative probability is computed until it exceeds the threshold value p_{thr_i}:

$$\sum_{j=0}^{k_i} P(s_i^j) \geq p_{thr_i}. \tag{4}$$

The number of retained sequences, denoted as k_i, is the minimal set that satisfies this condition, with all other sequences pruned. Since PPL can be directly calculated from the sequence probability, we use $P(s_i^j)$ as the metric prior to the completion of the sequence. The threshold p_{thr_i} is defined as:

$$p_{thr_i} = p^* \times \sum_{j=0}^{k_{i-1} \times V - 1} P(s_i^j). \tag{5}$$

Here, $p^* \in [0, 1]$ controls the number of sequences to be pruned, where a lower p^* results in more sequences being discarded. **Finally**, to control computational overhead, an additional hyperparameter

k^* is introduced. When k_i exceeds k^*, only the first k^* sequences are retained. At each step, LTM selects the most probable candidates while adhering to the constraints of p_{thr_i} and k^*. The pseudocode for the algorithm is presented in Algorithm 1.

Algorithm 1 Pseudocode of Dynamic Decoding

Input: A text sequence l;
Hyperparameters: p^*, k^*, maximum sequence length N;
Output: Beams B

Initialize $B = [\,]$
Initialize $P(S_0) = 1$
for i in range(1,N) **do**
  // Calculate probabilities of all possible sentences
  $P(Y_i) \leftarrow \mathcal{M}(l + B)$
  $P(S_i) \leftarrow P(Y_i) \times P(S_{i-1})$
  // Sort sentence probabilities
  $[P_{sort}(S_i), Token_{sort}] \leftarrow \text{sort}(P(S_i))$
  // Compute probability threshold p_{thr_i}
  $p_{thr_i} \leftarrow p^* \times \sum P(S_i)$
  // Select candidates T based on threshold
  $T \leftarrow \text{top}(Token_{sort}, P_{sort}(S_i), p_{thr_i})$
  // Update beams
  $B.\text{append}(Token_{sort}[: \min(\text{len}(T), k^*)])$
end for

In traditional sampling methods that select one token at a time, errors introduced early in the process can propagate through subsequent stages. Moreover, a token initially selected as the optimal choice may lose its advantage as the context evolves, leading to suboptimal sequence selection and a decline in overall performance. For instance, as illustrated in Figure 4, the left branch at step 1 has a lower probability than the right branch but achieves better performance at step 2. This highlights the limitations of methods that focus solely on immediate token probabilities without considering long-term sequence quality. In contrast, LTM explores multiple sequences and evaluates the overall long-term performance of each. This approach enables the LLM to look ahead at future tokens and retrospectively correct errors, thereby enhancing its ability to generate higher quality outputs.

4 Experimental Setup

In this section, we detail the experimental setup of our study, encompassing the LLMs employed, datasets utilized, baseline methods compared, evaluation metrics applied, and implementation details.

4.1 Backend LLMs

To verify the universality of our method, we test various open-source LLMs ranging from 1B to 13B parameters. These include Llama2-Chat-7B and Llama2-Chat-13B (Touvron et al., 2023), Llama3-Instruct-1B and Llama3-Instruct-3B (Dubey et al., 2024), and Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct (Yang et al., 2024). For brevity, we refer to these LLMs as Llama2-7B, Llama2-13B, Llama3-1B, Llama3-3B, Qwen-1.5B, and Qwen-3B in the subsequent sections.

4.2 Datasets

To simulate user feedback, we conduct experiments using mathematical and coding datasets, where the correctness of LLM outputs can be assessed by comparing them with ground-truth solutions. The datasets used are as follows: (1) **GSM8K** (Cobbe et al., 2021): This dataset contains 1,319 mathematical problems with standardized answers. We adopt a zero-shot approach, prompting the LLMs to generate both reasoning processes and final results. (2) **MultiArith** (Roy and Roth, 2015): Comprising 180 mathematical problems, this dataset uses the same initial prompts as GSM8K. (3) **HumanEval** (Chen et al., 2021): This dataset includes 164 programming questions. Prompts are formulated based on implementations from the DeCLaRe Lab (Chia et al., 2023).

4.3 Baselines

To evaluate the effectiveness of our method, we compare it with two general-purpose, two-stage prompt-based self-correction approaches: (1) **Critic Prompt** (Huang et al., 2024): To ensure experimental fairness, we adopt the experimental setting from Li et al. (2024), which instructs the LLM to identify errors in its previous responses and generate refined results. (2) **If or Else (IoE) Prompt** (Li et al., 2024): This method prompts the LLM to assess its confidence in the initial answer and generate a refined response if necessary.

For the Critic Prompt method, the LLM is instructed using the following prompt: "Review your previous answer and find problems with your answer. Based on the problems you found, improve your answer. Please reiterate your answer". For the IoE Prompt method, the prompt employed is "Review your previous answer. If you are very confident about your answer, maintain your answer. Otherwise, update your answer". The key differ-

ence between these methods aligns with variations in p_{cor} within the discussed framework.

4.4 Evaluation metrics

We evaluate model performance using top-1 accuracy ($acc@1$) for mathematical tasks and top-1 pass rate ($pass@1$) for coding tasks. $acc@1$ measures the percentage of test cases where the model's highest-ranked prediction matches the ground truth, while $pass@1$ quantifies the proportion of instances where the top prediction successfully passes a predefined coding test.

4.5 Implementation Details

Users typically expect a single output, but LTM may generate multiple responses. In the FTR self-correction method, we first evaluate the PPL metric for all generated responses and select the one with the lowest PPL as the final output. For other self-correction methods, only one response is generated per attempt using nucleus sampling (Holtzman et al., 2020; Fan et al., 2018; Holtzman et al., 2018) with $p = 0.95$ and $k = 15$.

5 Experimental Results

In this section, we present the empirical findings that demonstrate the efficacy of our approach.

5.1 Overall Comparison

Table 1 illustrates the comparative performance of our FTR method relative to baseline approaches across multiple datasets and LLMs. Our FTR method achieves significant improvements (10%–20%) across various datasets and LLMs, highlighting its effectiveness and adaptability in integrating user feedback with the advanced LTM decoding strategy. In contrast, evaluations indicate that the Critic and IoE prompt methods often result in performance degradation when compared to the initial input across most datasets and LLMs. These findings suggest that prompt-based techniques may not be robust in enhancing LLM performance.

This observation aligns with prior research by Huang et al. (2024), which indicates that certain prompts may mislead LLMs, resulting in a higher likelihood of correct responses being altered to incorrect ones. However, recent studies on advanced OpenAI GPT models, such as the research conducted by Li et al. (2024), present contrasting results. They propose that the effectiveness of confidence prompts improves when utilized with larger LLMs. This discrepancy may be attributed to the superior instruction-following and reasoning capabilities of these advanced models, which are less developed in smaller LLMs. Our experimental results underscore the limited effectiveness of prompt-based methods within the state-of-the-art open-source model under 13B parameters.

Method	GSM8K	MultiArith	HumanEval
# Llama2-7B #			
Initial Input	0.206	0.539	0.104
+ Critic Prompt	0.171	0.522	0.043
+ IoE Prompt	0.136	0.339	0.091
+ FTR (Ours)	**0.360**	**0.878**	**0.165**
# Llama2-13B #			
Initial Input	0.303	0.656	0.207
+ Critic Prompt	0.122	0.322	0.049
+ IoE Prompt	0.281	0.656	0.122
+ FTR (Ours)	**0.463**	**0.917**	**0.250**
# Llama3-1B #			
Initial Input	0.245	0.406	0.287
+ Critic Prompt	0.167	0.289	0.165
+ IoE Prompt	0.173	0.383	0.281
+ FTR (Ours)	**0.399**	**0.622**	**0.409**
# Llama3-3B#			
Initial Input	0.774	0.961	0.488
+ Critic Prompt	0.485	0.750	0.390
+ IoE Prompt	0.394	0.650	0.323
+ FTR (Ours)	**0.875**	**0.994**	**0.604**
# Qwen-1.5B#			
Initial Input	0.422	0.678	0.409
+ Critic Prompt	0.334	0.506	0.220
+ IoE Prompt	0.328	0.567	0.085
+ FTR (Ours)	**0.594**	**0.839**	**0.732**
# Qwen-3B#			
Initial Input	0.837	0.994	0.677
+ Critic Prompt	0.789	0.939	0.524
+ IoE Prompt	0.823	0.989	0.628
+ FTR (Ours)	**0.902**	**1.000**	**0.768**

Table 1: Performance comparison of different self-correction methods across different datasets and LLMs.

5.2 Feedback Experiment

Here, we examine the rationale behind our proposed FTR framework, which uses user feedback as a guiding signal to refine LLM outputs. One reason for the performance decline of prompt-based self-correction methods is their tendency to convert correct responses into incorrect ones.

User feedback serves as a crucial mechanism to mitigate the performance decline observed in prompt-based self-correction methods by identifying incorrect outputs. This approach also could be seamless integrated to prompt-based techniques within self-correction framework. Therefore, we conduct experiments using different configurations: (1) **feedback as prompt**. In this method, the accuracy of the initial answer is incorporated as an additional prompt to guide the LLM in refining its output. The prompt is "The answer you provided is incorrect, please review the answer and update it".

Method	GSM8K	MultiArith	HumanEval
# Llama2-7B #			
Initial Input	0.206	0.539	0.104
+ Prompt	0.243	0.694	0.140
+ Indicator (Ours)	0.328	0.810	0.146
# Llama2-13B #			
Initial Input	0.303	0.656	0.207
+ Prompt	0.359	0.806	0.220
+ Indicator (Ours)	0.455	0.872	0.243
# Llama3-1B #			
Initial Input	0.245	0.406	0.287
+ Prompt	0.301	0.506	0.317
+ Indicator (Ours)	0.374	0.556	0.384
# Llama3-3B#			
Initial Input	0.774	0.961	0.488
+ Prompt	0.822	0.972	0.534
+ Indicator (Ours)	0.859	0.983	0.567
# Qwen-1.5B#			
Initial Input	0.422	0.678	0.409
+ Prompt	0.512	0.750	0.476
+ Indicator (Ours)	0.630	0.900	0.512
# Qwen-3B#			
Initial Input	0.837	0.994	0.677
+ Prompt	0.867	0.994	0.732
+ Indicator (Ours)	0.892	1.000	0.768

Table 2: Effectiveness of self-correction methods under different user feedback utilization approaches.

(2) **feedback as indicator**: In this approach, the system directly generates the output based on the original input without an additional prompt. To ensure fairness, we apply the same nucleus sampling method to both configurations.

The results are presented in Table 2. Evidently, explicit feedback regarding the accuracy of the model's responses, used in both feedback prompts and FTR methods, leads to significant performance improvements by targeting refinements specifically to incorrect outputs. Notably, the feedback-as-indicator method outperforms the feedback-as-prompt method, suggesting that prompt-based approaches may lack robustness and potentially degrade model performance through additional prompts. Therefore, we recommend using user feedback as an indicator to trigger regeneration, rather than integrating it into the input prompt.

5.3 Decoding Comparison

To evaluate the LTM method, we assessed its performance as a standalone decoding strategy for LLMs without integrating any self-correction techniques, focusing on single-turn tasks. The models and datasets used are consistent with those described in the preceding sections. The baseline methods comprise: (1) **Beam Search**: This method selects the top-k most probable beams at each decoding step and is a classical decoding technique in the field of NLP. (2) **Combined Sampling**

	GSM8K	MultiArith	HumanEval
# Llama2-7B #			
Beam Search	0.261	0.739	0.134
Combined Sampling	0.253	0.733	0.134
Adaptive Decoding	0.257	0.722	0.128
LTM Decoding	**0.276**	**0.750**	**0.146**
# Llama2-13B #			
Beam Search	0.366	0.800	0.220
Combined Sampling	0.345	0.756	0.213
Adaptive Decoding	0.366	0.722	0.213
LTM Decoding	**0.378**	**0.833**	**0.232**
# Llama3-1B #			
Beam Search	0.268	0.478	**0.354**
Combined Sampling	0.231	0.411	0.287
Adaptive Decoding	0.257	0.322	0.348
LTM Decoding	**0.289**	**0.494**	**0.354**
# Llama3-3B #			
Beam Search	0.796	0.983	0.555
Combined Sampling	0.761	0.950	0.506
Adaptive Decoding	0.747	0.972	0.512
LTM Decoding	**0.804**	**0.994**	**0.561**
# Qwen-1.5B #			
Beam Search	**0.456**	0.489	0.610
Combined Sampling	0.418	0.439	0.390
Adaptive Decoding	0.439	0.506	0.476
LTM Decoding	**0.456**	**0.522**	**0.622**
# Qwen-3B #			
Beam Search	0.851	**1.000**	0.756
Combined Sampling	0.817	0.994	0.762
Adaptive Decoding	0.823	0.994	0.713
LTM Decoding	**0.852**	**1.000**	**0.768**

Table 3: Performance comparison of different decoding methods for LLMs.

(nucleus sampling): This approach combines top-p sampling and top-k sampling, with parameters $p = 095$ and $k = 15$ used consistently across all experiments. It is a widely adopted decoding strategy in LLMs. (3) **Adaptive Decoding**: This method enhances top-k sampling by dynamically adjusting the candidate set size at each generation step based on an entropy-based confidence score. Adaptive Decoding is a relatively novel technique compared to the aforementioned methods. For a fair comparison, we ensure that the computing budgets (the number of tokens during the decoding process) are approximately equal by adjusting the hyperparameters of the different decoding methods.

The results are presented in Table 3. It is evident that the LTM surpasses all baseline methods. This highlights the efficacy of multi-path exploration and dynamically adjusted candidate sets, which not only extend the exploration of decoding space but also ensure that computational resources are con-

centrated on the most crucial steps. Conversely, we observe that the combined decoding method exhibits performance lower than expected in terms of accuracy. Nonetheless, it remains prevalent in the current landscape of LLMs owing to the diversity it offers in its outputs. Therefore, to enhance the output diversity of LTM while maintaining effectiveness, it is crucial to explore a balance between these two factors in future work.

6 Related Work

In this section, we introduce some related work, including self-correction and decoding methods.

6.1 Self-Correction Methods

Numerous studies have advanced self-correction methods in the domain of LLMs. For instance, Madaan et al. (2024) proposed a three-stage framework that enhances LLM outputs by integrating feedback from previous iterations. Li et al. (2024) designed prompts to guide the LLM in assessing its confidence and deciding whether to generate a revised response. Huang et al. (2024) further explored the use of critic prompts to evaluate the self-correction capabilities of LLMs. Additionally, task-specific prompts have been developed for translation tasks to facilitate iterative refinement (Chen et al., 2024a), and methods like Self-Debug have enabled LLMs to generate feedback based on their own code and execution results (Chen et al., 2024b). External tools like search engines and compilers have also been integrated to support error correction (Gou et al., 2024). However, existing self-correction methods for LLMs rely on prompts while neglecting real-time user feedback. This can lead to redundant refinements and degrade LLM performance. Unlike previous work, our prompt-free approach incorporates useful user feedback to enhance efficiency and performance.

6.2 Decoding Strategies

Current decoding strategies in LLMs primarily use next-token prediction mechanisms such as top-k sampling(Fan et al., 2018; Holtzman et al., 2018) and nucleus sampling (top-p sampling)(Holtzman et al., 2020). In top-k sampling, the LLM selects the next token from the top k most probable tokens, while in nucleus sampling, it samples from the smallest set of tokens whose cumulative probability exceeds a threshold p. Basu et al. (2021) proposed a modified version of top-k sampling that incorporates a feedback mechanism to control the perplexity of the generated text. Zhu et al. (2024b) introduced adaptive decoding, a variant of top-k sampling that dynamically adjusts the size of k based on the information entropy of the token probability distribution.

In an effort to improve LLM performance, several approaches have been developed that explore multiple inference paths. For example, the Chain-of-Thought (CoT) reasoning algorithm (Wang and Zhou, 2024) improved reasoning by combining top-k sampling and greedy sampling, selecting the longest output from the generated candidates. Likewise, Zhu et al. (2024a) enhanced reasoning by decomposing model outputs into discrete steps and assigning deductive scores to determine how many branches should be processed. However, these approaches are often limited to fixed output templates and constrained reasoning steps, reducing their flexibility and practical applicability. Our proposed LTM decoding addresses these limitations by focusing on long-term reasoning paths and eliminating reliance on predefined templates, enhancing adaptability and flexibility in human-LLM interactions.

7 Conclusion

This work introduces the FTR self-correction framework, which significantly enhances the performance of LLMs by leveraging user feedback as a guiding signal. Specifically, when user feedback indicates dissatisfaction with the LLMs' output, the framework employs LTM—an advanced decoding strategy—to refine the response. By integrating LTM with feedback-triggered regeneration, the framework notably improves the overall quality of the LLMs' responses. Unlike existing methods that rely heavily on prompts and the LLMs' internal assessment capabilities, the FTR framework is more flexible and adaptive. This makes it particularly well-suited for real-world human-LLM interaction scenarios where intuitive feedback is readily available.

We propose two directions for future work: (1) Although LTM demonstrates superior performance, it lacks diversity in its outputs. Exploring methods to enhance the diversity of LTM would be a valuable area of research; (2) Recently, reinforcement learning (RL) has gained significant importance in the LLM field. One key direction is improving response generation for possible candidate solution. It would be interesting to investigate whether LTM can aid in generating data that benefits RL.

Limitations

Despite its advantages, LTM, as a greedy decoding method, is prone to higher levels of repetition and redundancy in text generation. Potential future directions include detecting and eliminating redundant outputs during the decoding process and integrating sampling mechanisms into the algorithm to improve output diversity. These enhancements could further reduce computational load and latency, thereby improving real-time user interaction.

In addition, LTM increases computational load to enhance the quality of LLM outputs, making it less suitable for scenarios that require both high output quality and low latency. Therefore, exploring a decoding approach that improves output quality while maintaining low latency is of significant importance.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Sourya Basu, Govardana Sachitanandam Ramachandran, Nitish Shirish Keskar, and Lav R. Varshney. 2021. Mirostat: a neural text decoding algorithm that directly controls perplexity. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. 2024a. Iterative translation refinement with large language models. In *Proceedings of the 25th Annual Conference of the European Association for Machine Translation (Volume 1), EAMT 2024, Sheffield, UK, June 24-27, 2024*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024b. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2023. Instructeval: Towards holistic evaluation of instruction-tuned large language models. *arXiv preprint arXiv:2306.04757*.

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*.

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

Angela Fan, Mike Lewis, and Yann N. Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating LLM hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2024. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*.

Loka Li, Zhenhao Chen, Guangyi Chen, Yixuan Zhang, Yusheng Su, Eric Xing, and Kun Zhang. 2024. Confidence matters: Revisiting intrinsic self-correction capabilities of large language models. *arXiv preprint arXiv:2402.12563*.

Fang Liu, Yang Liu, Lin Shi, Houkun Huang, Ruifeng Wang, Zhen Yang, Li Zhang, Zhongqi Li, and Yuchi Ma. 2024a. Exploring and evaluating hallucinations in llm-powered code generation. *arXiv preprint arXiv:2404.00971*.

Fengyuan Liu, Nouar AlDahoul, Gregory Eady, Yasir Zaki, Bedoor AlShebli, and Talal Rahwan. 2024b. Self-reflection outcome is sensitive to prompt construction. *arXiv preprint arXiv:2406.10400*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Ziyang Xu, Keqin Peng, Liang Ding, Dacheng Tao, and Xiliang Lu. 2024. Take care of your prompt bias! investigating and mitigating prompt bias in factual knowledge extraction. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, Yu-Yang Liu, and Li Yuan. 2023. Llm lies: Hallucinations are not bugs, but features as adversarial examples. *arXiv preprint arXiv:2310.01469*.

Tinghui Zhu, Kai Zhang, Jian Xie, and Yu Su. 2024a. Deductive beam search: Decoding deducible rationale for chain-of-thought reasoning. *arXiv preprint arXiv:2401.17686*.

Wenhong Zhu, Hongkun Hao, Zhiwei He, Yiming Ai, and Rui Wang. 2024b. Improving open-ended text generation via adaptive decoding. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.