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 incorrect 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 demonstrated exceptional capabilities across diverse tasks, such as text generation, question answering, and code synthesis (Achiam et al., 2023; Touvron et al., 2023; Guo et al., 2025). However, a persistent challenge lies in their tendency to produce factually incorrect information or flawed logical reasoning—a critical limitation widely documented in LLM inference research (Yao et al., 2023; Liu et al., 2024a). To address this, self-correction mechanisms have emerged as a promising solution. These mechanisms are typically implemented through carefully crafted prompting strategies (Ji et al., 2023; Madaan et al., 2024; Kim et al., 2024; Li et al., 2024; Chen et al., 2024; Huang et al., 2024), which guide models to introspect on their initial responses. The self-correction workflow generally follows a two-step process: first, generating an initial answer via standard LLM inference; and second, prompting the model to critically evaluate the initial output for errors, inconsistencies, or logical gaps, and subsequently revise it.

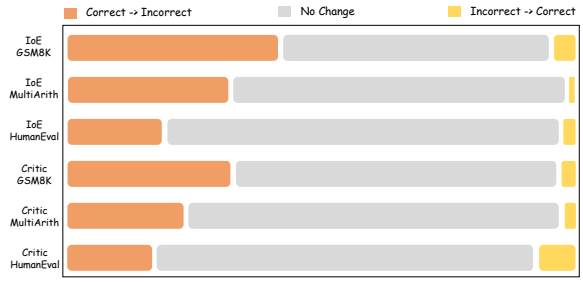

Figure 1: The percentage distribution of answer changes induced by self-correction using If or Else(IoE) Prompts (Li et al., 2024) and Critic Prompts (Huang et al., 2024), with experiments conducted on the Llama3-Instruct-3B.

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 Feedback-Triggered Regeneration (FTR), a self-correction framework that leverages user feedback (e.g., explicit thumbs-up/down signals) to guide LLMs' reasoning refinement. In human-AI interactions, users naturally provide feedback—such as dissatisfaction cues—when engaging with LLM outputs, which acts as a critical trigger for identifying responses requiring deeper revision. By treating feedback as a revision trigger, FTR avoids unnecessary reprocessing of outputs that have received positive user signals, focusing computational resources on cases requiring improvement. In the FTR framework, negative feedback guides the LLM to regenerate responses by reprocessing the original input, with feedback solely determining whether regeneration is initiated. This design circumvents limitations of prompt-based methods—such as self-critical prompting that relies on flawed introspection or labor-intensive prompt tuning for each task—thereby enhancing both correction efficiency and generalizability.

To address **C2**, we enhance FTR by integrating a Long-Term Multipath (LTM) decoding strategy, which enables LLMs to perform global reasoning through adaptively multi-step path exploration. Unlike standard autoregressive decoding methods (e.g., greedy decoding) that optimize solely for next-token probability, LTM evaluates multiple decoding paths at each step, prioritizing long-term sequence coherence and semantic consistency. This approach mitigates the limitation of short-term optimization—such as local optimum traps and context drift—that plague traditional decoding frameworks. Critically, LTM is dynamically activated during the regeneration stage—triggered by negative user feedback—to replace the standard single-path decoding, thereby enabling deep reasoning for error correction without incurring excessive computational overhead.

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.

2 Preliminaries

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:

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

which quantifies how confidently a language model predicts the sequence by measuring the inverse geometric mean of token probabilities, where lower values indicate better predictions. In this work, PPL is employed as a metric to assess the quality of a sequence during the decoding process (Jelinek et al., 1977; Brown et al., 2020).

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

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).

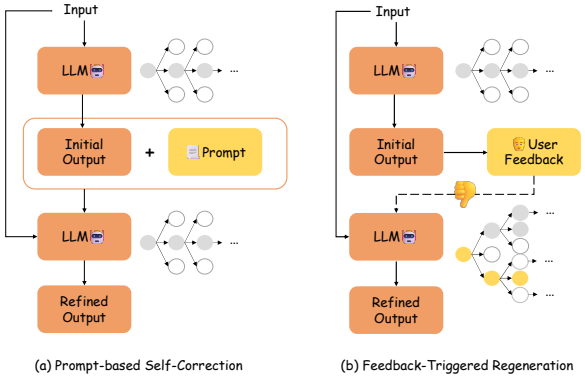

Figure 2: (a) Framework of the prompt-based self-correction approach. (b) Framework of our feedback-triggered self-correction approach.

While properly trained LLMs generally produce optimal initial responses to well-formed queries, our investigation identifies two key limitations in self-correction frameworks: (1) Prompt-induced solution drift, where externally introduced guidance inadvertently distorts the model's reasoning trajectory, causing systematic deviations from the original optimal solution (Huang et al., 2024); and (2) Correction capability constraints, as evidenced in Figure 3(b) showing persistent refinement failures even when explicit correctness feedback (p_{cor}) is provided. This dual challenge of prompt fragility and inherent correction limitations fundamentally restricts the practical effectiveness of current self-correction paradigms (Xu et al., 2024; Liu et al., 2024b).

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 equipped with the LTM decoding strategy, which will be 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

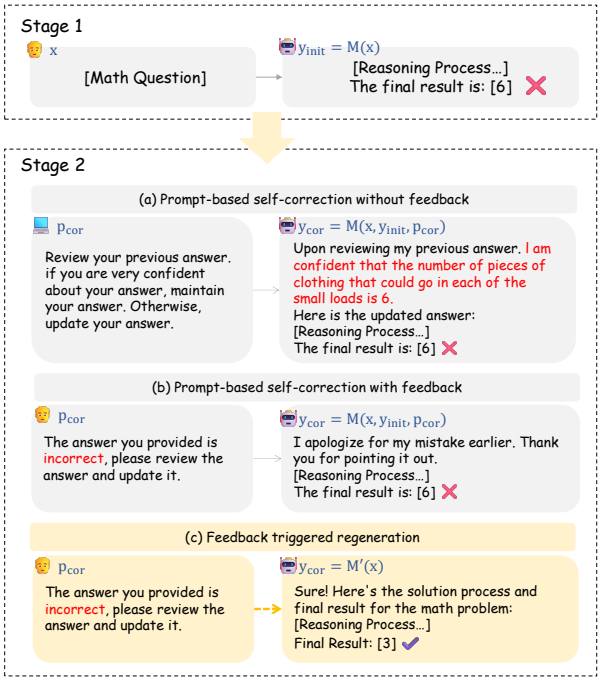

Figure 3: Comparison of different self-correction methods. (a) Self-assessment and update; (b) Revision with user feedback prompt; (c) Regeneration triggered by user feedback.

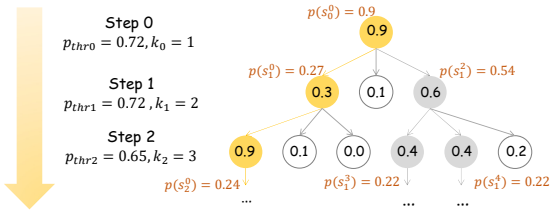

Figure 4: Illustration of LTM decoding strategies ($V = 3$), where black numbers in circles are token likelihood and red ones indicate sequence likelihood.

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

When users provide negative feedback on LLM responses, it indicates the need for deeper reasoning beyond standard next-token prediction, as this conventional paradigm lacks comprehensive evaluation of complete answer quality. Therefore, we enhance the decoding process through two key mechanisms: (1) **Multipath Exploration**: Instead of exploring a single path, token selection is performed using a "tree" structure rather than the traditional "chain" structure. This allows the LLM to explore multiple potential sequences simultaneously, as illustrated in Figure 4. (2) **Sequence Evaluation**: Using PPL as our quality metric, we dynamically retain the top k_i sequences at each step i, with k_i adjusted according to the step's PPL distribution. Unlike Beam Search's constant beam width, our method dynamically adjusts the candidate count k_i during decoding.

LTM's core mechanism dynamically determines k_i through token distributions, with mathematical formalization provided below and demonstration in Figure 4. **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. When all candidate sequences have equal length, PPL can be computed directly from the sequence probability. For computational convenience, we use $P(s_i^j)$ as our evaluation metric. 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. During decoding, LTM selects the most probable candidates subject to dual constraints p_{thr_i} and k^*, ultimately selecting the optimal answer through minimum PPL evaluation from multiple generated candidates.

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

4.1 Datasets

To evaluate the effectiveness of our method, we select mathematical and coding datasets that require model reasoning capabilities while allowing objective verification of outputs. The detailed dataset specifications are as follows: (1) **GSM8K** (Cobbe et al., 2021): This dataset comprises 1,319 grade-school mathematics problems with standard solutions, utilizing the zero-shot prompt: "*The following is a math problem from elementary school. Please provide the solution process and the final result. Write the final result within square brackets. Only one pair of square brackets should be used, and it should contain only a number. The question is:...*" (2) **MultiArith** (Roy and Roth, 2015): This dataset contains 180 mathematical problems evaluated using identical prompting methodology to 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).

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

4.3 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.

4.4 Implementation Details

To verify the universality of our method, we test various open-source LLMs ranging from 1B to 13B parameters, including 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.

All methods employ nucleus sampling (Holtzman et al., 2020) with hyperparameters $p = 0.95, k = 15$ during initial stage. In contrast, baseline approaches retain this strategy in the second stage, whereas our FTR framework employs the proposed LTM decoding strategy—featuring multipath exploration and long-term context evaluation—during the regeneration phase.

5 Experimental Results

5.1 Overall Comparison

Given the challenge of directly obtaining real user feedback in controlled experimental settings—where subjective human preferences and contextual interactions are difficult to replicate—we design two complementary evaluation protocols to rigorously assess FTR against state-of-the-art self-correction methods: (1) **Supervised Ground-Truth Evaluation (Protocol 1)**: In a supervised setting, we compare FTR with

Method	PROTOCOL 1 GSM8K	PROTOCOL 1 MultiArith	PROTOCOL 1 HumanEval	PROTOCOL 2 GSM8K	PROTOCOL 2 MultiArith	PROTOCOL 2 HumanEval
# Llama2-7B #						
Initial Input	0.206	0.539	0.104	0.206	0.539	0.104
+ Critic Prompt	0.171	0.522	0.043	0.202	0.572	0.104
+ IoE Prompt	0.136	0.339	0.091	0.189	0.406	0.116
+ FTR (Ours)	**0.360**	**0.878**	**0.165**	**0.269**	**0.744**	**0.140**
# Llama2-13B #						
Initial Input	0.303	0.656	0.207	0.303	0.656	0.207
+ Critic Prompt	0.122	0.322	0.049	0.276	0.611	0.207
+ IoE Prompt	0.281	0.656	0.122	0.308	0.700	0.195
+ FTR (Ours)	**0.463**	**0.917**	**0.250**	**0.419**	**0.822**	**0.226**
# Llama3-1B #						
Initial Input	0.245	0.406	0.287	0.245	0.406	0.287
+ Critic Prompt	0.167	0.289	0.165	0.246	0.439	0.287
+ IoE Prompt	0.173	0.383	0.281	0.244	0.439	0.317
+ FTR (Ours)	**0.399**	**0.622**	**0.409**	**0.310**	**0.473**	**0.384**
# Llama3-3B#						
Initial Input	0.774	0.961	0.488	0.774	0.961	0.488
+ Critic Prompt	0.485	0.750	0.390	0.753	0.972	0.506
+ IoE Prompt	0.394	0.650	0.323	0.744	0.878	0.482
+ FTR (Ours)	**0.875**	**0.994**	**0.604**	**0.837**	**0.983**	**0.585**
# Qwen-1.5B#						
Initial Input	0.422	0.678	0.409	0.422	0.678	0.409
+ Critic Prompt	0.334	0.506	0.220	0.446	0.667	0.390
+ IoE Prompt	0.328	0.567	0.085	0.434	0.656	0.323
+ FTR (Ours)	**0.594**	**0.839**	**0.732**	**0.459**	**0.689**	**0.585**
# Qwen-3B#						
Initial Input	0.837	0.994	0.677	0.837	0.994	0.677
+ Critic Prompt	0.789	0.939	0.524	0.843	0.994	0.720
+ IoE Prompt	0.823	0.989	0.628	0.846	0.994	0.687
+ FTR (Ours)	**0.902**	**1.000**	**0.768**	**0.879**	**1.000**	**0.750**

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

prompt-based baselines that rely solely on internal self-assessment (i.e., no user or external signal). Here, baselines perform self-correction via default prompt configurations, while FTR simulates supervised feedback by leveraging ground-truth labels to identify initially incorrect answers and trigger regeneration. This protocol isolates the impact of objective error signals, ideal for tasks with verifiable ground truths (e.g., math problem solving). (2) **Human-Mimicking Proxy Evaluation (Protocol 2)**: To bridge the gap between controlled experiments and real-world human-AI interactions, we employ GPT-4o (Hurst et al., 2024) as an automated proxy for human judgment. This unified framework applies GPT-4o's error classification (e.g., factual errors, logical flaws) to first-stage outputs, mimicking how users might subjectively evaluate responses. Critically, regeneration is triggered only for outputs deemed incorrect by the proxy, ensuring fair comparison across methods. [1]

Table 1 presents a comprehensive performance comparison between FTR and baseline approaches across datasets, LLMs, and both evaluation protocols. Under Protocol 1, FTR achieves substantial performance gains (10%–20%) consistently across all evaluated scenarios, highlighting its effectiveness in integrating user feedback with the LTM decoding strategy to drive targeted corrections. This consistency demonstrates FTR's adaptability to diverse model scales and task types, from factual answer to logical reasoning. In contrast, quantitative results show that baseline methods like Critic and IoE prompt techniques often exhibit performance degradation compared to their own initial outputs across most datasets and LLMs. These findings align with prior research by Huang et al. (2024), which attributes such issues to prompts potentially misleading LLMs into altering correct responses during self-assessment. Together, these results underscore the limitations of prompt-based self-correction in open-source models, where introspective capabilities remain constrained. Under Protocol 2, FTR demonstrates consistent superiority over baselines across model architectures and task domains, even when using noisy feedback proxies. This robustness validates FTR's effectiveness in realistic settings and confirms that improve-

[1]The rationality of this approach is supported by GPT-4o's demonstrated human-like competence: 76.6% accuracy on the MATH dataset (Hendrycks et al., 2021) and comparable code generation performance to humans (Suh et al., 2025). Quantitative analysis of the proxy's evaluation errors (false positives/negatives) is provided in Appendices Table 4.

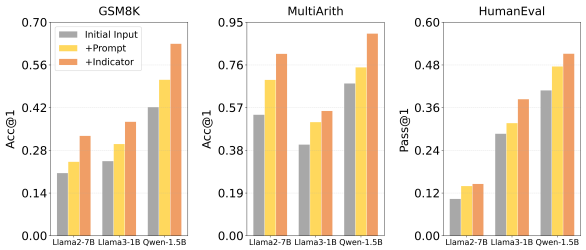

Figure 5: Performance comparison of different user feedback utilization approaches.

ments stem from its architectural design—rather than reliance on ground-truth labels—as shown by comparable gains in both protocols. Such cross-protocol analysis further reveals that explicit accuracy feedback (Protocol 1) and simulated human feedback (Protocol 2) both contribute to FTR's performance, albeit through different mechanisms, reinforcing its generalizability as a self-correction framework.

5.2 Feedback Experiment

Here, we analyze the design rationale of the proposed FTR framework, which leverages user feedback as a guiding signal to refine LLM outputs. A critical limitation of prompt-based self-correction methods is their propensity to erroneously revise correct responses during self-assessment—a phenomenon often attributed to flawed introspective mechanisms (Huang et al., 2024). User feedback addresses this by serving as an external validation signal, directly identifying outputs requiring revision without relying on the LLM's internal judgment. To investigate the optimal integration of feedback into self-correction workflows, we design two experimental configurations: (1) **feedback-as-prompt**: The accuracy of the initial answer is encoded as a natural language prompt to guide refinement. For example, when an initial response is incorrect, the LLM receives the prompt: "*The answer you provided contains factual errors. Please review the question and regenerate a correct response.*" (2) **feedback-as-indicator**: The system uses feedback solely as a binary trigger (regenerate if feedback is negative) and reprocesses the original input without additional prompts. Both configurations employ identical nucleus sampling strategy to ensure comparability.

Figure 5 presents the comparative performance of three representative LLMs across different reasoning benchmarks. The feedback-as-indicator approach significantly outperforms feedback-as-prompt methods in reasoning accuracy, providing empirical evidence that conventional prompt-based correction mechanisms introduce decision boundary instability and lead to performance degradation. These results empirically support the recommendation to employ user feedback primarily as a regeneration trigger—rather than embedding it in corrective prompts—to maintain decoding stability and optimize LLM self-correction efficiency. [2]

Decoding Strategy	GSM8K	MultiArith	HumanEval
# Llama2-7B #			
Greedy Decoding	0.243	0.683	0.122
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 #			
Greedy Decoding	0.330	0.750	0.220
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 #			
Greedy Decoding	0.286	0.428	0.274
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 #			
Greedy Decoding	0.782	0.967	0.500
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 #			
Greedy Decoding	0.441	0.456	0.457
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 #			
Greedy Decoding	0.842	1.000	0.756
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 2: Performance comparison of different decoding strategies for backend LLMs.

5.3 Decoding Comparison

To evaluate the LTM method, we assess 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) **Greedy Decoding**: This method always selects the word with the highest probability at each decoding step. (2) **Beam Search**: This

[2] Complete quantitative results for all backend LLMs and datasets are documented in Appendices Table 5.

method selects the top-k most probable beams at each decoding step and is a classical decoding technique in the field of NLP. (3) **Combined Sampling (nucleus sampling)**: This approach combines top-p and top-k sampling, with parameters $p = 0.95$ and $k = 15$ used consistently across all experiments. It is a widely adopted decoding strategy in LLMs. (4) **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.

To ensure fair comparative evaluation, we standardize computational budgets—quantified by generated token counts—across methods through hyperparameter adjustments.[3] Results in Table 2 demonstrate that LTM consistently outperforms baseline approaches. This superiority highlights the effectiveness of its multipath exploration and dynamically adjusted candidate set mechanism, which expand the decoding space exploration beyond single-path methods. Compared to Beam Search, LTM strategically allocates computational resources to critical decoding steps. For example, in scenarios with flattened token probability distributions, LTM adaptively selects diverse tokens, whereas Beam Search's fixed-width strategy may overlook high-potential paths, illustrating LTM's flexibility in optimizing resource utilization for improved performance. [4]

6 Related Work

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., 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. Other recent approaches boost LLM performance by exploring multiple decoding paths. For instance, Wang and Zhou (2024) combines sampling methods to select optimal outputs, while Zhu et al. (2024a) scores reasoning steps to determine path expansion. However, these methods are constrained by fixed templates and limited reasoning steps. Our LTM decoding overcomes these limitations through flexible, template-free reasoning for more adaptable human-AI 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-AI interaction scenarios where intuitive feedback is readily available.

[3] Detailed hyperparameter settings for different decoding strategies are presented in Appendices Table 6.

[4] Appendices A.1 presents a real illustrative case.

Limitations

Despite its advantages, LTM's multipath decoding mechanism exhibits a tendency to produce repetitive or redundant outputs in text generation tasks. Future research may focus on developing mechanisms to detect and mitigate such redundancy during decoding—such as dynamic path pruning based on semantic similarity—while incorporating advanced sampling techniques (e.g., diverse beam search) to enhance output diversity. These enhancements could not only reduce computational overhead and inference latency but also improve real-time interaction quality in human-AI dialogue systems. Meanwhile, our current experimental scope is limited to LLMs with fewer than 13 billion parameters, which serve as a cost-efficient platform for method validation. While prior studies (Li et al., 2024) have shown that prompt-based approaches excel in larger models (e.g., GPT series) due to their superior instruction-following capabilities, our controlled experiments on smaller architectures provide essential baselines for low-resource scenarios. Future work will systematically evaluate LTM's scalability across LLM model scales to characterize its impacts on computational efficiency, accuracy, and latency in diverse computational environments.

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*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

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. 2024. 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*.

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*.

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*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

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*.

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*.

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*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

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*.

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*.

Hyunjae Suh, Mahan Tafreshipour, Sam Malek, and Iftekhar Ahmed. 2025. Human or llm? A comparative study on accessible code generation capability. *CoRR*, abs/2503.15885.

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*.

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*.

An Yang, Baosong Yang, and Beichen Zhang et al. 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*.

A Appendices

A.1 Case Study

In this section, we present a case study comparing LTM with Beam Search to demonstrate LTM's distinctive advantages. As illustrated in Figure 6, we analyze the reasoning processes of both methods on a mathematical problem from the MultiArith dataset. For controlled comparison, Beam Search uses a fixed beam width of 3, while LTM dynamically adjusts its beam width with an average value of 3 across decoding steps.

The key observation reveals that at a critical decoding step with a relatively flat candidate distribution, Beam Search fails to retain all promising candidates due to its fixed beam width strategy, ultimately leading to an incorrect solution. In contrast, LTM dynamically adjusts its computational resources at this step and increases the beam width to 5, successfully preserving all potentially correct candidates and consequently arriving at the accurate final answer. This case clearly demonstrates LTM's adaptive advantage over static beam search methods in handling complicated decoding scenarios.

A.2 Inference Time Analysis

Although LTM decoding introduces computational overhead compared to single-beam decoding, FTR's design mitigates this cost by triggering regeneration only for samples with negative feedback, unlike standard self-correction methods that perform mandatory double-pass generation for all

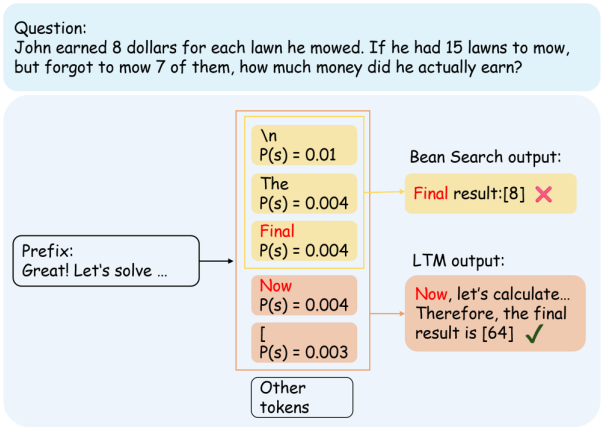

Figure 6: Comparison of decoding processes between Beam Search (fixed beam width=3) and LTM on a MultiArith dataset question. The yellow rectangle denotes Beam Search's candidate set, while the orange rectangle represents LTM's candidate set.

Method	GSM8K	MultiArith	HumanEval
# Llama2-7B #			
Initial Input	1×	1×	1×
+ Critic/IoE Prompt	2×	2×	2×
+ FTR (Ours)	2.47×	2.15×	3.87×
# Llama2-13B #			
Initial Input	1×	1×	1×
+ Critic/IoE Prompt	2×	2×	2×
+ FTR (Ours)	2.48×	1.64×	2.98×
# Llama3-1B #			
Initial Input	1×	1×	1×
+ Critic/IoE Prompt	2×	2×	2×
+ FTR (Ours)	2.40×	2.49×	2.32×
# Llama3-3B#			
Initial Input	1×	1×	1×
+ Critic/IoE Prompt	2×	2×	2×
+ FTR (Ours)	1.42×	1.12×	2.64×
# Qwen-1.5B#			
Initial Input	1×	1×	1×
+ Critic/IoE Prompt	2×	2×	2×
+ FTR (Ours)	2.07×	1.81×	2.89×
# Qwen-3B#			
Initial Input	1×	1×	1×
+ Critic/IoE Prompt	2×	2×	2×
+ FTR (Ours)	1.30×	1.02×	2.03×

Table 3: Inference time comparison of different self-correction methods across different datasets and LLMs.

inputs and thus lead to net efficiency gains in low error rate scenarios. Baseline methods such as Critic and IoE Prompt require $2 \times N \times t$ computation time (with N as the number of samples and t as the average inference time per sample), a linear complexity arising from re-generating responses for all inputs. In contrast, FTR's total computation time is $N \times t \times (1 + p \times n)$, where p is the proportion of samples requiring regeneration and n is the average LTM beam width, with efficiency emerging when $p \times n < 1$. It is worth noting that the impact of beam size on performance can be reduced through parallel computing. For example, with parallel LTM decoding at an average beam width $n = 3$, the per-sample regeneration cost is $1.85\times$ single-beam decoding time, leading to the efficiency condition $p < \frac{1}{1.85} \approx 0.54$. Detailed test results across backend LLMs and datasets are presented in Table 3, confirming that FTR's adaptive regeneration strategy balances reasoning depth with computational efficiency for real-world deployments. [5]

A.3 Additional Experiments Details

[5]The experimental evaluations were performed on a computing system featuring an NVIDIA RTX 6000 Ada Generation GPU and Intel Xeon Platinum 8358P CPUs with 32 physical cores (64 threads) running at 2.60GHz base frequency and 48MB L3 cache.

LLM Model		GSM8K	MultiArith	HumanEval
Llama2-7B	FN	34.93	13.83	5.88
	FP	8.40	30.23	1.36
Llama2-13B	FN	22.31	22.03	11.76
	FP	9.57	40.32	1.54
Llama3-1B	FN	16.41	2.74	6.38
	FP	25.71	60.75	5.13
Llama3-3B	FN	13.03	1.73	15.00
	FP	15.77	28.57	17.86
Qwen-1.5B	FN	13.54	4.10	20.90
	FP	35.70	82.76	20.62
Qwen-3B	FN	9.51	0.56	8.11
	FP	18.14	0.00	20.75

Table 4: False Negative (FN) and False Positive (FP) rates of GPT-4o on different datasets (%).

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 5: Detailed model performance comparison under different user feedback utilization approaches.

	GSM8K	MultiArith	HumanEval
# Llama2-7B #			
Beam Search	$n = 3$	$n = 5$	$n = 7$
Combined Sampling	$n = 3$	$n = 5$	$n = 6$
Adaptive Decoding	$n = 3$	$n = 5$	$n = 7$
LTM Decoding	$p^* = 0.8, k^* = 7$	$p^* = 0.9, k^* = 7$	$p^* = 0.85, k^* = 7$
# Llama2-13B #			
Beam Search	$n = 4$	$n = 3$	$n = 5$
Combined Sampling	$n = 4$	$n = 3$	$n = 5$
Adaptive Decoding	$n = 4$	$n = 3$	$n = 5$
LTM Decoding	$p^* = 0.9, k^* = 7$	$p^* = 0.85, k^* = 7$	$p^* = 0.8, k^* = 7$
# Llama3-1B #			
Beam Search	$n = 3$	$n = 5$	$n = 3$
Combined Sampling	$n = 3$	$n = 5$	$n = 3$
Adaptive Decoding	$n = 3$	$n = 5$	$n = 3$
LTM Decoding	$p^* = 0.8, k^* = 7$	$p^* = 0.9, k^* = 7$	$p^* = 0.8, k^* = 7$
# Llama3-3B #			
Beam Search	$n = 3$	$n = 7$	$n = 5$
Combined Sampling	$n = 3$	$n = 6$	$n = 7$
Adaptive Decoding	$n = 3$	$n = 6$	$n = 7$
LTM Decoding	$p^* = 0.8, k^* = 7$	$p^* = 0.95, k^* = 7$	$p^* = 0.9, k^* = 7$
# Qwen-1.5B #			
Beam Search	$n = 3$	$n = 5$	$n = 7$
Combined Sampling	$n = 3$	$n = 5$	$n = 7$
Adaptive Decoding	$n = 3$	$n = 5$	$n = 7$
LTM Decoding	$p^* = 0.8, k^* = 7$	$p^* = 0.9, k^* = 7$	$p^* = 0.9, k^* = 8$
# Qwen-3B #			
Beam Search	$n = 3$	$n = 5$	$n = 7$
Combined Sampling	$n = 3$	$n = 5$	$n = 7$
Adaptive Decoding	$n = 3$	$n = 5$	$n = 7$
LTM Decoding	$p^* = 0.85, k^* = 7$	$p^* = 0.85, k^* = 7$	$p^* = 0.9, k^* = 7$

Table 6: Hyperparameters of different decoding methods used in Tables 1 and 2. For beam search, n denotes the number of beams, while for combined sampling and adaptive decoding, n indicates the number of generated answers.