Finding the Cracks: Improving LLMs Reasoning with Paraphrastic Probing and Consistency Verification

Anonymous ACL submission

Abstract

Large language models have demonstrated impressive performance across a variety of reasoning tasks. However, their problem-solving ability often declines on more complex tasks due to hallucinations and the accumulation of errors within these intermediate steps. Recent work has introduced the notion of *critical tokens*– tokens in the reasoning process that exert significant influence on subsequent steps. Prior studies suggest that replacing critical tokens can refine reasoning trajectories. Nonetheless, reliably identifying and exploiting critical tokens remains challenging. To address this, we propose the **P**araphrastic **P**robing and **C**onsistency **V**erification (**PPCV**) framework. PPCV operates in two stages. In the first stage, we roll out an initial reasoning path from the original question and then concatenate paraphrased versions of the question with this reasoning path. And we identify critical tokens based on mismatches between the predicted top-1 token and the expected token in the reasoning path. A criterion is employed to confirm the final critical token. In the second stage, we substitute critical tokens with candidate alternatives and roll out new reasoning paths for both the original and paraphrased questions. The final answer is determined by checking the consistency of outputs across these parallel reasoning processes. We evaluate PPCV on mainstream LLMs across multiple benchmarks. Extensive experiments demonstrate PPCV substantially enhances the reasoning performance of LLMs compared to baselines.

1 Introduction

The emergence of large language models (LLMs) (Brown et al., 2020; Grattafiori et al., 2024; Achiam et al., 2023; Yang et al., 2025a) has astonished the AI community with their remarkable capabilities across a wide range of reasoning tasks, including mathematical problem solving, programming, and planning.

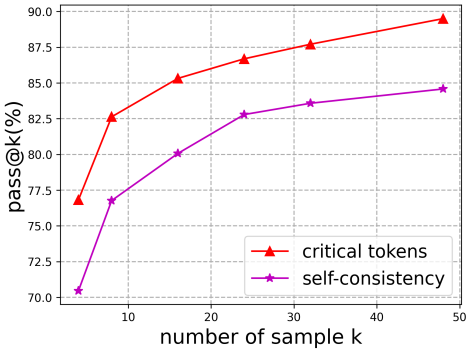


Figure 1: Comparison of the effects of critical tokens and Self-Consistency on the reasoning performance of LLMs, evaluated on samples from GSM8K .

By generating intermediate reasoning steps through techniques such as chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022; Zhang et al., 2022; Sprague et al., 2025), LLMs can emulate human-like reasoning processes and achieve strong performance on diverse benchmarks. Despite their success, the problem-solving ability of LLMs often declines on complex reasoning tasks due to hallucinations and the accumulation of errors in intermediate steps (Lightman et al., 2023; Ling et al., 2023; Bubeck et al., 2023). To mitigate this issue, prior works (Wang et al., 2024; Yuan et al., 2025; Chen et al., 2024a; Chen and Li, 2024) have proposed the inference-time optimization paradigm, which evaluates and refines reasoning trajectories through self-correction by the LLM itself or with feedback from external verifiers such as process reward models (PRMs) (Uesato et al., 2022). However, existing studies (Yang et al., 2025b; Tyen et al., 2024; Stechly et al., 2025) suggest that LLMs struggle to reliably identify errors in their reasoning and often fail to correct previous outputs without external guidance. Moreover, obtaining high-quality, stepwise process supervision for training external verifiers is challenging and limits the practicality of these approaches (Feng et al., 2025).

Case study on the effects of question paraphrasing

Original question: In a card game, you are awarded 10 points if you win one round. While playing, Jane ends up with 60 points. If she lost 20 points, then how many rounds did she play? (ground truth answer: 8)

Response: To solve this problem, we need to follow these steps:

Step 1: Determine the total points Jane earned by subtracting the points she lost from her final score. Total points earned = Final score - Points lost. Total points earned = 60 – 20. Total points earned = 40.

Step 2: Since Jane earns 10 points for each round she wins, we can find the number of rounds she played by dividing the total points earned by the points earned per round. Number of rounds = Total points earned / Points earned per round. Number of rounds = 40 / 10. Number of rounds = 4.

Therefore, Jane played 4 rounds. ✗

Paraphrased question: At a bridge competition, every game won grants the player 10 points. Unfortunately, Jane started with an advantage but ended up losing 20 points and scoring only 60 points. If she had won x rounds and lost none, what would be the value of x? (ground truth answer: 8)

Response: To solve this problem, we need to follow these steps:

Step 1: Understand the given information. Jane started with an advantage but ended up losing 20 points. She scored a total of 60 points. Every game won grants the player 10 points.

Step 2: Determine the total points Jane would have had if she hadn't lost any points. Since she lost 20 points, we need to add 20 to her total score to find out how many points she would have had if she hadn't lost any points. Total points without loss = 60 + 20 = 80 points.

Step 3: Calculate the number of games Jane won. Since every game won grants the player 10 points, we can divide the total points without loss by 10 to find the number of games she won. Number of games won = Total points without loss / 10 = 8. The number of games she won is equal to 8. ✓

Figure 2: Case study illustrating that LLMs can refine intermediate incorrect reasoning segments.

Recent work has introduced the concept of *critical tokens* (Lin et al., 2025), which play pivotal roles in intermediate reasoning steps and exert strong influence on subsequent reasoning and final outcomes. We evaluate the criticality of a token based on the average accuracy of sampling rollouts generated from the partial reasoning trajectories truncated at the token's position. Suppose the rollout correctness score of token r_i as $C(i) = \frac{1}{M}\sum_{m=1}^{M} \mathbf{1}(\Phi_m(r_i) = y)$, where M is the number of sampling rollouts, $\Phi(\cdot)$ denotes the function that extracts the final answer, y is the ground-truth answer, and r_i is considered critical if it satisfies the following two conditions. First, near-zero local correctness: $C(i) < \epsilon_1$, ϵ_1 is close to zero (e.g., below 5%). Second, persistent downstream failure: $C_j < \epsilon_2$, $\forall j \in i+1, \ldots, i+L$. ϵ_2 denotes a threshold (e.g., 10%).

Prior studies suggest that critical tokens often diverge from human-annotated error tokens, yet they induce more sustained degradation in reasoning quality than other tokens. Moreover, as illustrated in Figure 3, replacing critical tokens in an incorrect reasoning trajectory with suitable candidate tokens can correct subsequent steps and lead to the right answer in new rollouts. To quantitatively assess the effectiveness of critical tokens, we conduct an empirical study using LLMs such as Llama-3.1-8B-Instruct (Grattafiori et al., 2024) on reasoning tasks. Specifically, we randomly sample 100 instances with incorrect reasoning steps from the GSM8K (Cobbe et al., 2021) training data. Following the criterion, we locate critical tokens through exhaustive search. We then truncate the reasoning path at the critical token, substitute it with alternative tokens, and roll out new reasoning paths. For example, as shown in Figure 3, the token "woman" is replaced with "remaining". We evaluate performance using Pass@k and compare against Self-

Case study on critical tokens

Question: There were 10000 spectators at a basketball match. 7000 of them were men. Of the remaining spectators, there were 5 times as many children as women. How many children were there? (ground truth answer: 2500)

Response:
To find the number of children, we need to first find the number of **women** and then multiply it by 5.
Step 1: Find the number of women. Total spectators = 10000. Men = 7000. Women = Total spectators – Men. Women = 10000 – 7000. Women = 3000.
Step 2: Find the number of children. There are 5 times as many children as women. Children = 5 * Women. Children = 5 * 3000. Children = 15000.
So, the number of children is 15000. ✗

Response(replacement of critical tokens):
To find the number of children, we need to first find the number of **remaining** spectators (excluding men) and then calculate the number of women and children.
Step 1: Find the number of remaining spectators (excluding men). Total spectators = 10000. Men = 7000. Remaining spectators = Total spectators – Men = 10000 - 7000= 3000
Step 2: Let's assume the number of women is 'x'. Since there are 5 times as many children as women, the number of children will be 5x.
Step 3: The total number of remaining spectators is the sum of women and children. We can write this as an equation: x + 5x = 3000. Combine like terms:\n6x = 3000. Divide both sides by 6: x = 3000 / 6. x = 500.
Step 4: Now that we have the number of women, we can find the number of children. Number of children = 5x= 5 * 500= 2500.
Therefore, the number of children is 2500. ✓

Figure 3: An example demonstrating how substitution of a critical token (red) with a candidate token (purple) modifies the reasoning and produces the correct answer.

Consistency (Wang et al., 2023), which also samples multiple reasoning paths. As shown in Figure 1, critical token replacement provides a clear advantage in correcting errors compared to plain sampling. Nonetheless, reliably identifying and leveraging critical tokens for reasoning remains a nontrivial challenge.

Recent studies (Zhou et al., 2024; Chen et al., 2024b), on *surface form*, the way questions, assumptions, and constraints are phrased, have revealed its subtle influence on the trajectory of intermediate reasoning steps, As shown in Figure 2. LLMs could adjust the intermediate steps under the paraphrased form of the question. This motivates us to explore the role of paraphrasing in the extraction and utilization of critical tokens for reasoning tasks. To this end, we propose the **P**araphrastic **P**robing and **C**onsistency **V**erification (**PPCV**) framework , a two-stage approach designed to leverage critical tokens to enhance the reasoning ability of LLMs. In the first stage, we probe critical tokens using paraphrased questions. Specifically, we first roll out the initial reasoning path from the original question, then concatenate paraphrased questions with this reasoning path. The resulting synthetic inputs are fed into the LLM to obtain token-level logits for each position in the reasoning path. Positions where the predicted top-1 token diverges from the expected token are flagged as potential pivotal points, as these positions are sensitive to paraphrased inputs and can trigger a pivot in the reasoning trajectory. Next, an empirical criterion is applied to determine the final critical token. In contrast to prior work (Lin et al., 2025), which depends on external models for identifying critical

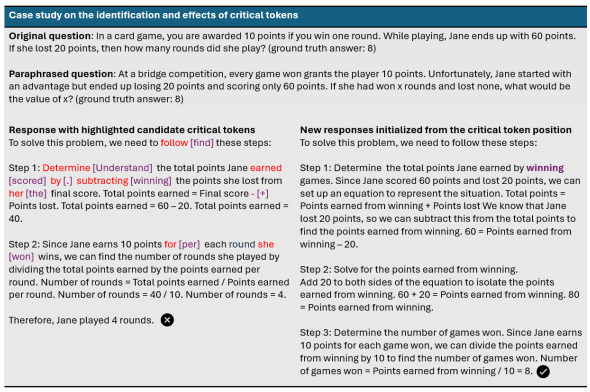


Figure 4: Case study illustrating the identification and effects of critical tokens identified by our method. Tokens highlighted in red indicate candidate critical tokens, whereas tokens highlighted in purple correspond to alternative tokens generated when conditioning on paraphrased questions.

tokens with ambiguous criteria, our method introduces a self-contained mechanism that pinpoints critical tokens.

In the second stage, we leverage the extracted critical tokens to refine the initial reasoning path. Specifically, we select the top-K tokens (include critical token itself) at the critical token position and roll out new reasoning paths for both the original and paraphrased questions. Based on the empirical observation that trajectories leading to correct answers are robust to paraphrastic perturbations, we propose a paraphrase consistency mechanism. In contrast to Self-Consistency (Wang et al., 2023), which relies on majority voting, our method selects the final answer by comparing outputs from paraphrased and original questions and choosing the one with the most consistent matches. To address potential ties across multiple answers, we further introduce similarity-weighted paraphrase consistency, which incorporates similarity scores between paraphrased and original questions when computing consistency.

Compared with self-correction (Wu et al., 2024; Miao et al., 2024) and PRM-based methods (Wang et al., 2024; Yuan et al., 2025), our framework exploits critical tokens to refine reasoning trajectories without requiring step-level error detection by the LLM itself or auxiliary models. We evaluate our method on mainstream LLMs across mathematical and commonsense reasoning benchmarks, demonstrating consistent improvements in reasoning performance. The contributions of the paper is summarized as follows:

- We propose a novel two-stage reasoning framework, **P**araphrastic **P**robing and **C**onsistency **V**erification (**PPCV**) designed to extract and leverage critical tokens to enhance the reasoning performance of LLMs.
- We show that critical tokens can more effectively correct erroneous reasoning trajectories than traditional sampling methods like Self-Consistency. Furthermore, our approach successfully extracts these tokens through paraphrastic probing, achieving improved final results via paraphrase consistency.
- We evaluate our method on mainstream LLMs across various reasoning tasks, including math and logic reasoning. Experimental results show significant performance improvements over baseline methods.

2 Related Work

Inference-Time Optimization for LLM Reasoning. Recent advances in chain-of-thought (CoT) prompting have demonstrated strong reasoning capabilities in LLMs by generating intermediate reasoning steps. This has inspired extensive research on inference-time optimization for reasoning trajectories, including search-based methods (Yao et al., 2023; Besta et al., 2024) and sampling-based methods (Wang et al., 2023; Xu et al., 2025). To improve reasoning reliability, prior works further explore self-correction (Madaan et al., 2023; Chen et al., 2024a) and external verifier-based approaches (Wang et al., 2024; Snell et al., 2024). In contrast, our method improves reasoning by directly leveraging critical tokens without relying on external verifiers or iterative refinement.

Paraphrasing for LLMs. Prior studies (Zhou et al., 2024; Chen et al., 2024b) show that LLM reasoning is highly sensitive to prompt phrasing, where even minor paraphrasing can substantially affect reasoning performance. Motivated by this observation, recent methods employ paraphrasing to improve reasoning quality (Yadav et al., 2024; Chen et al., 2024b) and estimate model uncertainty (Gao et al., 2024). In this work, we utilize paraphrasing as a principled mechanism to identify critical tokens and aggregate reasoning outcomes.

3 Methodology

In this section, we present the two components of our framework in detail: paraphrastic probing and consistency verification. An detailed illustration of

our framework is shown in Figure 6. We then discuss the proposed method and provide the complete algorithm.

3.1 Paraphrastic Probing

Previous findings (Zhou et al., 2024; Chen et al., 2024b; Huang et al., 2025) on the impact of a problem's surface form suggest that the quality of intermediate reasoning steps is influenced not only by the underlying mathematical relationships and logic, but also by how the problem is expressed. Notably, LLMs are sometimes able to solve a paraphrased version of a problem that they fail to solve in its original form, highlighting the potential of paraphrasing to uncover pivotal tokens that are critical for successful reasoning. Motivated by this observation, we introduce paraphrastic probing to efficiently identify the critical token. Given the original question q_0, we first prompt the LLM to generate multiple paraphrased forms, denoted as $q_1, q_2, ..., q_N$, where N is the number of the paraphrased questions. We adopt Automatic Prompt Engineering (APE) (Zhou et al., 2022) to derive paraphrasing instructions that preserve semantic integrity and all numerical values, mathematical relationships, and core logical structures of the problem, while maximizing linguistic and contextual diversity. Additional details can be found in Appendix C. We then obtain the initial reasoning path $r_0^{q_0}$ for the original question using greedy decoding. This reasoning path is subsequently concatenated with each paraphrased question, and the resulting synthetic inputs are fed into the LLM to compute the token probability distribution at each position in $r_0^{q_0}$. Specifically, the token probability distribution at ith position conditioned on the paraphrased question q_n is expressed as

$$P_i^{q_n} = \text{LLM}(\tilde{a}_i | \mathcal{I}, q_n, r_{0,<i}^{q_0}), \tag{1}$$

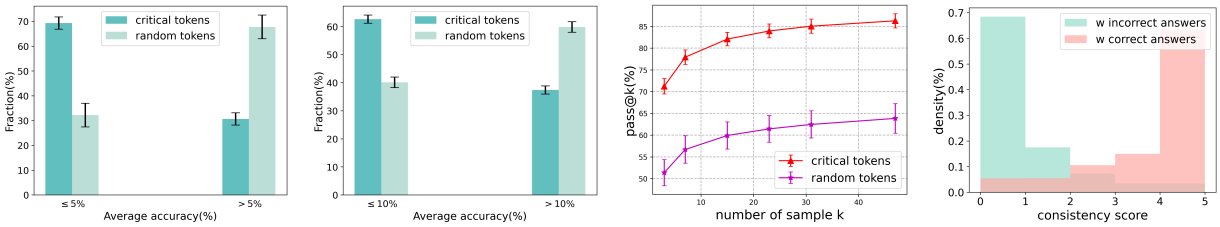
where $\mathcal{I}$ denotes the instruction prefix and $\tilde{a}_i$ represents the sampled token at ith position. The token $\tilde{a}_i$ is regarded as a candidate critical token if predicted top-1 token does not match the expected token at the same position in $r_0^{q_0}$, i.e.,

$$P_i^{q_n} \neq a_i, \tag{2}$$

where a_i denotes the token at the ith position in $r_0^{q_0}$.

To validate the effectiveness of our method in identifying critical tokens and to demonstrate the pivotal role these tokens play when extracted via paraphrastic probing, we conduct a case study illustrated in Figure 4. Because large language models can adjust their reasoning trajectories under the paraphrased form of the question, certain tokens in the original reasoning sequence produce different subsequent tokens when conditioned on paraphrased inputs. In this example, most candidate critical tokens are located within incorrect reasoning segments. Notably, when we identify "subtracting" as a critical token and replace it with an alternative token (i.e., "winning"), the resulting rollout corrects the erroneous reasoning steps and yields the correct final answer. This example highlights the impact of critical tokens and underscores the effectiveness of our method in identifying tokens that are pivotal to reasoning outcomes.

In addition, we conduct a quantitative analysis to examine the authenticity and impact of tokens extracted via paraphrastic probing, comparing them against randomly selected tokens. Specifically, we sample 100 instances with incorrect reasoning trajectories from the GSM8K (Cobbe et al., 2021) training sets. Following the paraphrastic probing pipeline, we identify candidate critical tokens in the early reasoning steps. In each run, we randomly sample 40% of the candidate critical tokens for evaluation and repeat the experiment independently 10 times. For comparison, we apply the same procedure to randomly selected tokens. All evaluations are conducted using Llama-3.1-8B-Instruct (Grattafiori et al., 2024). We first compute the average accuracy of rollouts generated from partial trajectories truncated at the position of the extracted tokens, and compare these results with those obtained from random-token truncation. As shown in Figure 5a, a large proportion of the extracted tokens exhibit near-zero correctness, consistent with the first criterion of critical tokens. We further evaluate the average accuracy of rollouts initiated from subsequent tokens (5–10 tokens). The results in Figure 5b indicate that errors persist beyond the identified positions, supporting the second criterion that critical tokens induce sustained degradation in downstream reasoning. Finally, we substitute the extracted critical tokens with alternative tokens and roll out multiple new reasoning trajectories. As shown in Figure 5c, replacing critical tokens leads to a significantly larger improvement in reasoning accuracy compared to substituting random tokens. These results further validate both the pivotal role of the identified critical tokens and the effectiveness of our paraphrastic probing method.



(a) (b) (c) (d)

Figure 5: Analysis of extracted critical tokens on GSM8K. (a) Fraction of tokens whose rollout correctness falls below or above 5%. (b) Fraction of subsequent tokens whose rollout correctness falls below or above 10%. (c) Impact of extracted versus random tokens on reasoning performance. (d) Density distributions of consistency scores for correct and incorrect rollouts.

Additional results can be found in Appendix A.

We introduce a heuristic verifier to select the final critical token from multiple candidates. For a candidate token a_i and paraphrased question q_n, the verification score is defined as

$$\Delta_{q_n}(a_i) = \max P_i^{q_n} - P_i^{q_n}(\tilde{a}_i = a_i). \quad (3)$$

where $P_i^{q_n}$ denotes the predictive distribution at position i on question q_n. Intuitively, Δ measures how much the predicted top-1 token deviates from the expected token, indicating the token's potential impact on the reasoning trajectory. For each extracted token a_i,we take the maximum score across paraphrases,

$$\Delta(a_i) = \max_{q_n} \Delta_{q_n}(a_i), \quad (4)$$

and select the final critical token as

$$a_c =_i \Delta(a_i). \quad (5)$$

3.2 Consistency Verification

After identifying the final critical token a_c, we aim to refine the original reasoning path with alternative tokens and achieve final answer with paraphrase consistency mechanism. Specifically, we generate a set of alternative tokens $a_c^0, a_c^1, a_c^2, ..., a_c^{K-1}$ using the LLM conditioned on original question q_0, where a_c^0 is the original token in $r_0^{q_0}$ and the remaining tokens are sampled via top-K sampling. The initial reasoning path is truncated at the position of critical token, and each alternative token is concatenated to form synthetic inputs $\tilde{r}_c^0, \tilde{r}_c^1, \tilde{r}_c^2, ..., \tilde{r}_c^{K-1}$. We then roll out new reasoning trajectories for each synthetic input with respect to both the original and paraphrased questions using greedy decoding, denoted as $r_k^{q_0}, r_k^{q_1}, ..., r_k^{q_N}$ for $k = 0, 1, 2, ..., K-1$. Next, for the rollout with the kth alternative token, we compare the answers obtained from the paraphrased forms with that from the original form and compute a consistency score $c_k = \sum_{n-1}^{N} \mathbb{I}(\Phi(r_k^{q_0}) = \Phi(r_k^{q_n}))$, where $\Phi(\cdot)$ and $\mathbb{I}(\cdot)$ denotes the function that extracts the final answer from a reasoning trajectory and the indicator function, respectively. The answer associated with the highest consistency score is then selected as the final prediction

$$\text{ans}_f = \Phi(r_k^{q_0}), \text{where } k =_k c_k. \quad (6)$$

To justify our paraphrase consistency mechanism, we investigate the impact of paraphrased forms on LLM reasoning. We sample instances from the GSM8K (Cobbe et al., 2021) and follow our pipeline to extract critical tokens. From each truncated reasoning trajectory, we roll out multiple reasoning paths by concatenating alternative tokens. For each original question, we generate five paraphrased variants and compute the consistency score for resulting rollouts. The evaluation is conducted on Llama-3.1-8B-Instruct (Grattafiori et al., 2024). We then analyze the distribution of consistency scores for rollouts that yield correct versus incorrect answers. As shown in Figure 5d, more than 90% of rollouts with correct answers achieve a consistency score of at least 1, whereas this proportion drops to around 30% for rollouts with incorrect answers. This sharp contrast indicates that correct rollouts are more robust across paraphrased variants, motivating the design of our paraphrase consistency mechanism to exploit this property for improved final predictions.

To address potential collisions when multiple answers obtain the same maximum consistency score, we introduce similarity-weighted consistency verification. Inspired by weighted majority voting (Dogan and Birant, 2019), this approach adjusts the influence of each paraphrased form on the consistency score according to its similarity to the original form. Intuitively, paraphrased forms with lower

Table 1: Comparison of our method with baseline approaches on Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.2.

Model	Method	GSM8K	GSM-Hard	Math500	SVAMP	ARC
Llama-3.1	Chain-of-Thought	77.40	28.00	31.00	83.00	58.91
	Self-Consistency	80.60	31.80	37.80	85.10	60.75
	Tree-of-Thought	75.74	33.28	31.60	81.20	80.72
	Guided Decoding	75.51	32.45	31.20	81.70	81.74
	Predictive Decoding	81.43	40.26	34.00	85.90	84.56
	Phi-Decoding	86.58	39.88	38.20	84.50	85.41
	PPCV (Ours)	**88.24**	**49.73**	**50.00**	**89.60**	**88.31**
Mistral-7B	Chain-of-Thought	46.45	26.91	12.20	62.40	41.42
	Self-Consistency	50.38	28.65	14.20	66.70	44.54
	Tree-of-Thought	50.49	25.78	11.40	60.60	41.04
	Guided Decoding	50.79	27.07	14.00	62.90	39.51
	Predictive Decoding	55.67	27.07	14.40	62.10	47.87
	Phi-Decoding	**56.60**	28.43	13.40	63.20	60.24
	PPCV (Ours)	56.58	**31.08**	**14.60**	**69.30**	**69.88**

Table 2: Comparison of our method with baseline approaches on Qwen3-32B (non-thinking mode).

Model	Method	AIME2024	AIME2025	BRUMO2025	HMMT2025
Qwen3-32B	Chain-of-Thought	30.00	23.67	30.00	9.67
	Guided Decoding	26.67	22.67	28.67	7.33
	Predictive Decoding	32.67	24.00	33.33	10.33
	Phi-Decoding	33.60	24.33	36.67	10.67
	PPCV (Ours)	**40.00**	**26.00**	**43.33**	**13.33**

similarity should exert greater weight, as they provide stronger evidence of robustness, whereas those closely resembling the original form contribute less. Concretely, we first extract embeddings for both the original and paraphrased questions and compute their similarity scores as $s_n = \text{sim}(q_0, q_n)$, where $\text{sim}(\cdot)$ denotes a similarity measure such as cosine similarity. We derive weights via a softmax function $w_n = \text{softmax}(s_n) = \frac{\exp(-\lambda s_n)}{\sum_n \exp(-\lambda n)}$, where λ is the temperature scaling coefficient. Finally, the similarity-weighted consistency score is defined as $\tilde{c}_k = \sum_{n-1}^{N} w_n \mathbb{I}(\Phi(r_k^{q_0}) = \Phi(r_k^{q_n}))$. This ensures agreement with more diverse paraphrases contributes more strongly to the final decision.

3.3 Discussion

Notably, our framework builds upon existing concepts such as critical tokens and paraphrasing. However, the key contribution of our work lies in establishing a new perspective that connects paraphrastic sensitivity with token-level reasoning instability. This perspective enables a unified framework for both diagnosing and correcting reasoning failures in LLMs. Rather than relying on self-reflection or auxiliary supervision, our method is motivated by the observation that LLM reasoning is highly sensitive to surface-form variations, raising the question of whether such perturbations can serve as diagnostic signals for reasoning instability. From this perspective, we find that incorrect reasoning trajectories tend to exhibit lower stability under paraphrastic perturbations. Building on this insight, our framework leverages paraphrasing not merely for sample generation, but as a principled mechanism for diagnosing and refining reasoning trajectories. More discussions can be found in Appendix E.

Table 3: Comparison of model performance when using critical tokens versus random tokens.

Method	GSM8K	GSM-Hard	Math500	SVAMP	ARC
Chain-of-Thought	77.40	28.00	31.00	83.00	58.91
random tokens	82.08	40.29	42.12	84.77	75.68
critical tokens (Ours)	**88.24**	**49.73**	**50.00**	**89.60**	**88.31**

Table 4: Comparison of our proposed paraphrase consistency against the majority voting.

Method	GSM8K	GSM-Hard	Math500	SVAMP	ARC
Chain-of-Thought	77.40	28.00	31.00	83.00	58.91
majority voting	87.20	47.36	48.19	88.80	86.16
paraphrase consistency (Ours)	**88.24**	**49.73**	**50.00**	**89.60**	**88.31**

In our framework, we select a single token as the critical token. However, this does not imply that only one critical token exists in a reasoning trajectory. Rather, the design choice of using a single critical token is motivated primarily by computational efficiency and methodological simplicity, instead of a strict assumption about the reasoning process. Our empirical results demonstrate that identifying and refining even a single influential token can already yield substantial improvements in reasoning performance. Nevertheless, complex reasoning problems may involve multiple failure points distributed across different reasoning steps. Our PPCV framework is naturally extensible to multiple critical tokens. For example, the verifier can be adapted to select the top-K candidate tokens for diagnosis and refinement. At last, The details of the algorithm can be found in Appendix D.

4 Experiments

In this section, we describe the experimental setup, followed by the main results of our method compared to the baselines. We also perform ablation study and computational cost analysis.

4.1 Setup

Datasets. To comprehensively assess our method, we evaluate it on seven benchmarks. Six focus on mathematical reasoning, including GSM8K (Cobbe et al., 2021), GSM-Hard (Gao et al., 2023), SVAMP (Patel et al., 2021), Math500 (Hendrycks et al., 2021), and the more challenging competition-level datasets AIME2024, AIME2025, BRUMO2025, and HMMT2025 (Balunović et al., 2025). In addition, we use ARC-Challenge (Clark et al., 2018) to evaluate knowledge reasoning ability of LLMs.

Baselines. In our experiments, we use

Table 5: Comparison of the average number of rollouts, average output tokens per question, and performance on GSM-Hard and Math500 using Llama-3.1.

Dataset	Method	Avg. Rollouts	Avg. Output Tokens per Question	Acc.
GSM-Hard	Self-Consistency	48	17,403	31.80
	Predictive Decoding	78	19,600	40.26
	Phi-Decoding	36	15,731	39.88
	PPCV-Top5 (Ours)	25	12,173	47.05
	PPCV-Top10 (Ours)	50	22,010	**49.73**
Math500	Self-Consistency	48	41,122	37.80
	Predictive Decoding	97	46,214	34.00
	Phi-Decoding	45	40,518	38.20
	PPCV-Top5 (Ours)	25	28,301	46.40
	PPCV-Top10 (Ours)	50	50,454	**50.00**

Chain-of-Thought (CoT) (Wei et al., 2022), Self-Consistency (Wang et al., 2023), Tree-of-Thought (ToT) (Yao et al., 2023), Guided Decoding (Xie et al., 2023), Predictive Decoding (Ma et al., 2025), and Phi-Decoding (Xu et al., 2025) as baseline methods.

Metric. Following prior work, we adopt pass@k (k=1,4) as the primary evaluation metric.

Implementation Details. In our experiments, we adopt Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Mistral-7B-Instruct-v0.2 (Jiang et al., 2023), Qwen-3-32B (Yang et al., 2025a) and DeepSeek-R1-Distill-Llama-70B as the target models. we employ the non-thinking mode for Qwen-3-32B. Throughout our method, we employ the same model for generating paraphrased problems, identifying critical tokens, and producing new rollouts. In the first stage, we generate 4 paraphrased variants for each problem in the math benchmarks and 3 variants for each problem in the ARC dataset. In the second stage, we select the top 10 tokens for new rollouts, with the temperature scaling coefficient λ set to 2. For fair comparison, we ensure a comparable inference budget across methods. Specifically, we rollout 48 samples for Self-Consistency (Wang et al., 2023). For Predictive Decoding (Ma et al., 2025) and Phi-Decoding (Xu et al., 2025), we rollout 4-8 samples per foresight step, and each problem typically involves 5–8 foresight steps. We also adopt a zero-shot CoT prompt to elicit the new rollouts. For the baselines, we strictly follow their original settings, including temperature values, sampling strategies, and the number of few-shot examples. All experiments are conducted on NVIDIA A100 GPUs.

4.2 Main Results

The main results are summarized in Table 1 and Table 2. The results indicate that Self-Consistency (Wang et al., 2023) effectively improves the reasoning performance of LLMs compared to vanilla Chain-of-Thought prompting. For example, Llama-3.1-8B-Instruct (Grattafiori et al., 2024) achieves about 3% higher accuracy with Self-Consistency (Wang et al., 2023) than with CoT. These findings suggest that augmenting reasoning during inference through sampling is an effective way to refine reasoning trajectories. Recent decoding-based methods, such as Predictive Decoding (Ma et al., 2025) and Phi-Decoding (Xu et al., 2025), also achieve strong results.Unlike prior works that rely on carefully designed prompts to self-correct errors in intermediate steps, these two methods modify the current step by probing future steps with pre-defined reward signals. Furthermore, our experimental results demonstrate that the proposed method consistently outperforms the baselines across most tasks, spanning both mathematical and knowledge reasoning, thereby highlighting its generalization ability across different reasoning settings. Notably, our method even surpasses the latest approaches such as Predictive Decoding (Ma et al., 2025) and Phi-Decoding (Xu et al., 2025). In particular, it achieves approximately 50.00% accuracy on the Math500 dataset (Hendrycks et al., 2021), exceeding these baselines considerably. The results on competition-level datasets further demonstrate the effectiveness of our method in enhancing the reasoning ability of LLMs. These results indicate that our method can effectively extract critical tokens that play a pivotal role in the final outcome and correct the reasoning trajectory by leveraging alternative tokens. Additional results can be found in Appendix F.

4.3 Ablation Study

In this section, we analyze the contribution of each stage individually. All the evaluations are conducted on Llama-3.1-8B-Instruct.

Effectiveness of extracted critical tokens. To demonstrate the effectiveness of our extracted critical tokens, we conduct an evaluation in which the critical tokens are replaced with random tokens in the first stage, while keeping the second stage unchanged. This evaluation is performed across multiple benchmark datasets, with pass@1 as the metric. The results, shown in Table 3, reveal a substantial decline in performance. These findings highlight the pivotal role of critical tokens and indicate that our method can effectively identify and extract them. More ablation study on comparison with Paraphrased Majority Voting (PMV) can be found in Appendix G.

Effectiveness of paraphrase consistency. We also evaluate the effectiveness of our proposed paraphrase consistency and compare it with traditional majority voting. While keeping the first stage unchanged, instead of using paraphrased forms to generate new reasoning steps, we simply sample multiple new steps from alternative tokens conditioned on the original question and use majority voting to determine the final answer. The results, shown in Table 4, reveal a noticeable decline in performance, highlighting the importance of paraphrased forms in improving the intermediate reasoning steps.

Impact of number of sampled alternative tokens.We investigate the influence of the number of sampled alternative tokens in the second stage by selecting values of 3, 5, 7, and 10. The results, shown in Figure 10a, demonstrate that performance improves as the number of alternative tokens increases. This suggests that exploring more reasoning steps with additional alternative tokens during inference can be beneficial for reasoning tasks.

5 Computational Cost Analysis

In this section we examine the composition of the latency in our method. The latency arises from four components: Paraphrased question generation (T_1); initial answer generation (T_2), equivalent to vanilla CoT; a forward pass for identifying critical tokens (T_3), which does not generate new tokens and is computationally lightweight; rollouts of truncated trajectories using alternative tokens under both the original and paraphrased questions (T_4), which constitutes the main source of overhead.

We evaluate all components on Llama-3.1-8B-Instruct using vLLM on NVIDIA A100 GPUs, with a maximum output length of 4096 tokens for each question. For our method, we use 4 paraphrased questions on math datasets and 3 on ARC, and select the top-10 candidate tokens as alternatives. The updated average latency results are reported in Figure 10b. As expected, T_1 scales with the number of paraphrases, T_3 remains minimal, and T_4 dominates the total cost. Specifically, T_4 depends on the number of top-k alternative tokens, the number of paraphrased questions and the position of the critical token in the trajectory. Since the new rollouts from truncated trajectories for different alternative tokens and paraphrased questions are independent, T_4 can take advantage of vLLM's parallelism. These rollouts can therefore be processed concurrently, improving overall efficiency. This is reflected in the higher throughput (tokens/sec) shown in Figure 10c. And results of our method in latency comparable to baseline methods, even on challenging benchmarks such as Math500 (Hendrycks et al., 2021) and GSM-Hard (Gao et al., 2023) where the critical token tends to occur in later reasoning steps. On the GSM8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021) benchmarks, our method, as well as baselines such as Predictive Decoding, would incur a approximately 6–9$\times$ latency overhead compared to vanilla Chain-of-Thought (Wei et al., 2022).

To ensure fair comparison, we also quantify the inference budget using the average number of rollouts per question and the average number of generated output tokens per question. We evaluate on GSM-Hard (Gao et al., 2023) and Math500 (Hendrycks et al., 2021) to compare the compute budgets of baselines such as Self-Consistency (Wang et al., 2023) and our method with different compute budgets by varying the number of alternative tokens K. All experiments are conducted on NVIDIA A100 GPUs with a maximum output length of 4096 tokens. As shown in Table 5, our method achieves stronger performance under comparable inference budgets. Furthermore, strong performance can still be maintained under reduced compute budgets, indicating that the observed gains are not simply due to increased computational cost, but stem from more effective reasoning refinement. More analysis on the trade-off between the latency and performance can be found in Appendix H.

6 Conclusion

In this study, inspired by beneficial impact of paraphrase forms on reasoning, we investigate the pivotal role of critical tokens in shaping the reasoning trajectory. To leverage these two factors, we propose the Paraphrastic Probing and Consistency Verification framework. Our framework consists of two stages: Paraphrastic Probing, which identifies and extracts critical tokens, and Consistency Verification, which adopts paraphrase forms to generate new reasoning trajectories with alternative tokens to reach the final answer. We evaluate our proposed framework with different large language models and extensive evaluations across multiple benchmarks demonstrate the promising performance of our method.

Limitations

Despite the effectiveness of PPCV, several limitations remain. First, our framework introduces additional inference-time computational overhead compared to standard reasoning methods, since it requires paraphrase generation, critical token probing, and multiple rollouts from alternative tokens. Although our experiments show that better reasoning performance can generally be achieved with more sampled alternative tokens and paraphrased variants, this also increases latency and inference cost. In practice, there exists a trade-off between computational efficiency and reasoning performance, and future work could explore more efficient critical token selection and adaptive rollout strategies.

Second, our method relies on access to token-level probabilities and intermediate decoding states for identifying and manipulating critical tokens. Consequently, PPCV is currently applicable mainly to open-source LLMs or APIs that expose token logits and decoding information. This limits its applicability to many closed-source commercial models where such internal information is inaccessible. Extending the framework to black-box settings remains an important direction for future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint*.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. Matharena: Evaluating llms on uncontaminated math competitions.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2025. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. In *Forty-second International Conference on Machine Learning*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, and 1 others. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint*.

Sijia Chen and Baochun Li. 2024. Toward adaptive reasoning in large language models with thought rollback. In *Forty-first International Conference on Machine Learning*.

Sijia Chen, Baochun Li, and Di Niu. 2024a. Boosting of thoughts: Trial-and-error problem solving with large language models. In *The Twelfth International Conference on Learning Representations*.

Wenqing Chen, Weicheng Wang, Zhixuan Chu, Kui Ren, Zibin Zheng, and Zhichao Lu. 2024b. Self-para-consistency: Improving reasoning tasks at low cost for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 14162–14167.

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*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint*.

Alican Dogan and Derya Birant. 2019. A weighted majority voting ensemble approach for classification. In *2019 4th international conference on computer science and engineering (UBMK)*, pages 1–6. IEEE.

Shengyu Feng, Xiang Kong, Shuang Ma, Aonan Zhang, Dong Yin, Chong Wang, Ruoming Pang, and Yiming Yang. 2025. Step-by-step reasoning for math problems via twisted sequential monte carlo. In *The Thirteenth International Conference on Learning Representations*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Xiang Gao, Jiaxin Zhang, Lalla Mouatadid, and Kamalika Das. 2024. Spuq: Perturbation-based uncertainty quantification for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2336–2346.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint*.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. In *Forty-second International Conference on Machine Learning*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *NeurIPS 2023 Workshop on Generalization in Planning*.

Alexander Havrilla, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 2024. Glore: When, where, and how to improve llm reasoning via global and local refinements. In *Forty-first International Conference on Machine Learning*.

Mingqian He, Yongliang Shen, Wenqi Zhang, Zeqi Tan, and Weiming Lu. 2024. Advancing process verification for large language models via tree-based preference learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2086–2099.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, and 1 others. 2025. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations. In *Forty-second International Conference on Machine Learning*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Yang Li. 2025. Policy guided tree search for enhanced llm reasoning. In *Forty-second International Conference on Machine Learning*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. 2025. Critical tokens matter: Token-level contrastive estimation enhances llm's reasoning capability. In *Forty-second International Conference on Machine Learning*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 36:36407–36433.

Chang Ma, Haiteng Zhao, Junlei Zhang, Junxian He, and Lingpeng Kong. 2025. Non-myopic generation of language models for reasoning and planning. In *The Thirteenth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2024. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. In *The Twelfth International Conference on Learning Representations*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

Zhenting Qi, MA Mingyuan, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2025. Mutual reasoning makes smaller llms stronger problem-solver. In *The Thirteenth International Conference on Learning Representations*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint*.

Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations*.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2025. On the self-verification limitations of large language models on reasoning and planning

tasks. In *The Thirteenth International Conference on Learning Representations*.

Sree Harsha Tanneru, Chirag Agarwal, and Himabindu Lakkaraju. 2024. Quantifying uncertainty in natural language explanations of large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 1072–1080. PMLR.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Yuanzhu Peter Chen, and Tony Mak. 2024. Llms cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13894–13908.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint*.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025. Reasoning aware self-consistency: Leveraging reasoning paths for efficient llm sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3613–3635.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference on Learning Representations*.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2024. Large language models can self-correct with key condition verification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12846–12867.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650.

Fangzhi Xu, Hang Yan, Chang Ma, Haiteng Zhao, Jun Liu, Qika Lin, and Zhiyong Wu. 2025. Phi-decoding: Adaptive foresight sampling for balanced inference-time exploration and exploitation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13214–13227.

Vikas Yadav, Zheng Tang, and Vijay Srinivasan. 2024. Pag-llm: Paraphrase and aggregate with large language models for minimizing intent classification errors. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 2569–2573.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint*.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E Gonzalez, Bin CUI, and Shuicheng YAN. 2025b. Supercorrect: Advancing small llm reasoning with thought template distillation and self-correction. In *The Thirteenth International Conference on Learning Representations*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Zhiyuan Zeng, Xiaonan Li, Junqi Dai, Qinyuan Cheng, Xuan-Jing Huang, and Xipeng Qiu. 2024. Reasoning in flux: Enhancing large language models reasoning through uncertainty-aware adaptive guidance. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2401–2416.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. 2025. Free process rewards without process labels. In *Forty-second International Conference on Machine Learning*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*.

Yue Zhou, Yada Zhu, Diego Antognini, Yoon Kim, and Yang Zhang. 2024. Paraphrase and solve: Exploring and exploiting the impact of surface form on mathematical reasoning in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2793–2804.

A Empirical Study

we follow the previous pipeline to conduct a quantitative analysis to examine the authenticity and impact of tokens extracted via paraphrastic probing, comparing them against randomly selected tokens. Specifically, we sample 100 instances with incorrect reasoning trajectories from the Math500 (Hendrycks et al., 2021) training sets. Following the paraphrastic probing pipeline, we identify candidate critical tokens in the early reasoning steps. In each run, we randomly sample 40% of the candidate critical tokens for evaluation and repeat the experiment independently 10 times. For comparison, we apply the same procedure to randomly selected tokens. All evaluations are conducted using Llama-3.1-8B-Instruct (Grattafiori et al., 2024). We first compute the average accuracy of rollouts generated from partial trajectories truncated at the position of the extracted tokens, and compare these results with those obtained from random-token truncation. As shown in Figure 7a, we also obverse a large proportion of the extracted tokens exhibit near-zero correctness. We further evaluate the average accuracy of rollouts initiated from subsequent tokens (5–10 tokens). The results in Figure 7b also indicate that errors persist beyond the identified positions. Finally, we substitute the extracted critical tokens with alternative tokens and roll out multiple new reasoning trajectories. As shown in Figure 7c, replacing critical tokens leads to a significantly larger improvement in reasoning accuracy compared to substituting random tokens.

To justify our paraphrase consistency mechanism on Math500 dataset (Hendrycks et al., 2021), we follow the same procedure. The evaluation is conducted on Llama-3.1-8B-Instruct (Grattafiori et al., 2024). We then analyze the distribution of consistency scores for rollouts that yield correct versus incorrect answers. As shown in Figure 7d, majority of rollouts with correct answers achieve a consistency score of at least 1, whereas this proportion drops sharply for rollouts with incorrect answers.

B Related Work

Inference-Time Optimization for LLM reasoning. With the advent of chain-of-thought (CoT) prompting, LLMs have demonstrated strong reasoning capabilities by producing intermediate steps during inference. This success has motivated a growing body of work (Wu et al., 2025; Snell et al.,

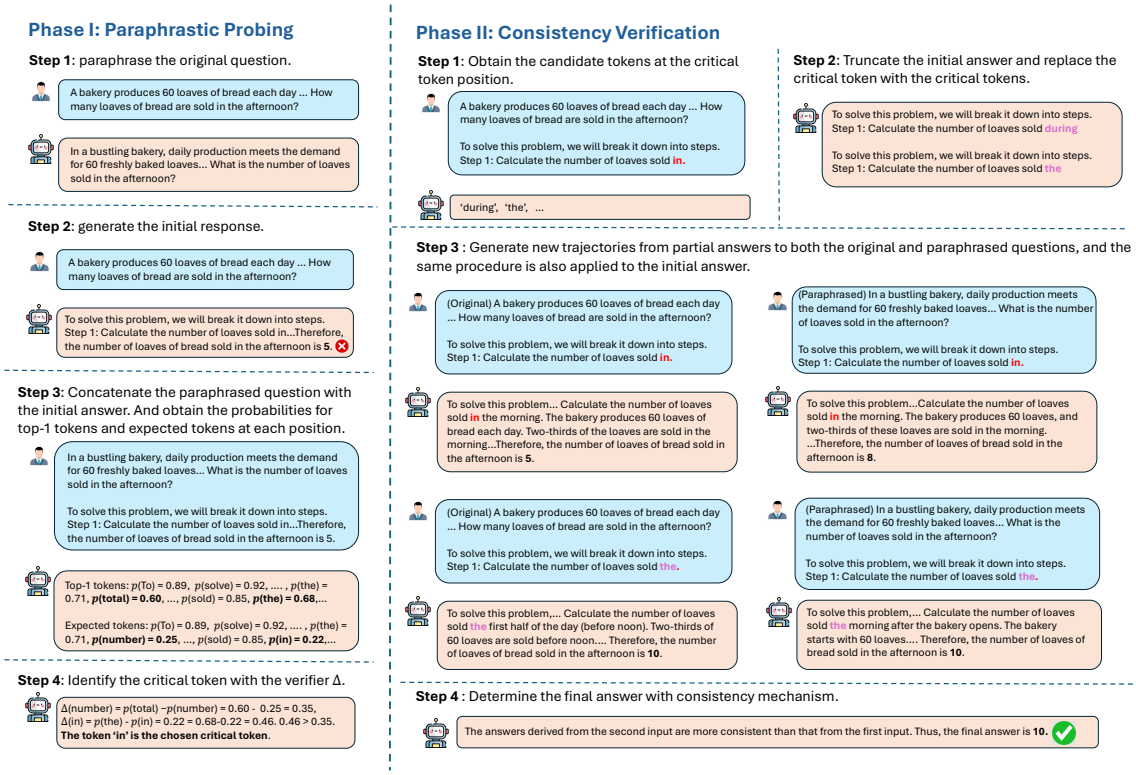
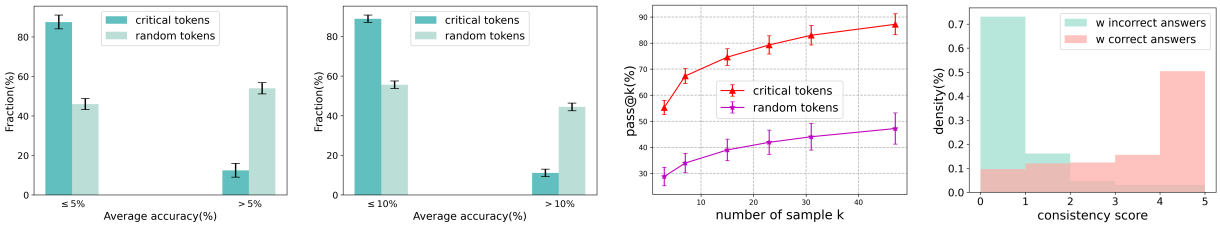


Figure 6: Illustration of the proposed paraphrastic probing and consistency verification (PPCV) framework. The framework comprises two stages: (i) probing critical tokens through paraphrased forms, and (ii) rolling out new reasoning steps with alternative tokens and selecting the final answer using the paraphrase consistency verification mechanism.

2024) on augmenting reasoning trajectories at test time to further improve performance. Existing approaches can be broadly categorized into search-based methods (Bi et al., 2025; Yao et al., 2023; Hao et al., 2023; Xie et al., 2023; Besta et al., 2024), such as Tree-of-Thoughts (Yao et al., 2023), and sampling-based methods (Wang et al., 2023; Xu et al., 2025; Wan et al., 2025; Ma et al., 2025), such as Self-Consistency (Wang et al., 2023). However, due to intrinsic hallucinations (Bubeck et al., 2023), LLMs often generate erroneous intermediate steps, which can ultimately lead to incorrect answers, especially on complex problems. This limitation highlights the need for inference-time optimization of reasoning processes.

To address this issue, one line of research (Yin et al., 2024; Chen et al., 2024a; Ling et al., 2023; Wu et al., 2024; Miao et al., 2024; Madaan et al., 2023) designs instructional prompts that guide LLMs to detect and refine their own mistakes. Despite its appeal, prior work has shown that the effectiveness of self-correction is limited in practice. Another line of work (Wang et al., 2024; Yuan et al., 2025; He et al., 2024; Havrilla et al., 2024) introduces external verifiers, such as process reward models (Snell et al., 2024), to identify and filter out error-prone reasoning steps. These methods typically require high-quality training data for the verifier, with data scarcity often mitigated through strategies such as Monte Carlo Tree Search (Guan et al., 2025; Qi et al., 2025; Li, 2025; Zhang et al., 2024). In addition, a recent line of decoding-based approaches (Xu et al., 2025; Ma et al., 2025) seeks to improve reasoning by dynamically adjusting the next-token prediction based on future trajectory probing. In contrast, our method refines reasoning by directly leveraging critical tokens, without relying on stepwise verification or external verifiers. This design underscores both the utility and universality of our approach.

Paraphrasing for LLMs. A growing number of work (Zhou et al., 2024; Chen et al., 2024b) has examined the impact of a problem's surface form on the reasoning ability of LLMs. Findings (Zhou et al., 2024; Chen et al., 2024b; Huang et al., 2025) suggest that even subtle modifications in phrasing can substantially affect both the reasoning process and the final outcome. Building on this observation, several methods (Yadav et al., 2024; Chen et al., 2024b) leverage beneficial paraphrasing to enhance



(a) (b) (c) (d)

Figure 7: (a) Comparison of the fraction of tokens whose associated rollout correctness falls below or above 5%, for tokens extracted by our method versus randomly selected tokens. (b) Comparison of the fraction of the subsequent tokens whose associated rollout correctness falls below or above 10%, for tokens extracted by our method versus randomly selected tokens. (c) Comparison of the impact of extracted tokens by our method versus random tokens on LLM reasoning performance. (d) Comparison of the density distributions of consistency scores for rollouts with correct and incorrect answers. All the evaluations are conducted on Math500 dataset.

LLM performance in tasks such as reasoning and intent classification. In addition, paraphrasing has been employed to assess model uncertainty (Gao et al., 2024; Tanneru et al., 2024), thereby enhancing the reliability and trustworthiness of LLM applications. In our work, we utilize paraphrasing as a principled tool to extract critical tokens and to aggregate answers.

C Automatic Prompt Engineering for Paraphrasing

The quality of paraphrased questions is crucial for our framework, as it directly affects both the identification of critical tokens and the stability of paraphrase-based consistency verification. To ensure high-quality paraphrasing, we adopt Automatic Prompt Engineering (APE) (Zhou et al., 2022) as an evaluation to optimize the paraphrasing instruction. This allows us to systematically control the quality of paraphrases rather than relying solely on manually written prompts. The APE procedure we employ consists of four steps:

- We create a small set of original questions paired with valid paraphrased variants. Using this exemplar set, the model generates multiple candidate instructions that could produce the paraphrased outputs from the original questions.
- Prompt each candidate instruction to the language model to generate paraphrases for the original problems and compare the mean solve rate change before and after paraphrasing.
- Choose the instruction that maximizes the mean solve rate change.
- Repeat the previous steps multiple times.

In our experiments, we initialize the paraphrasing instruction and iteratively refine it using APE with samples drawn from the GSM8K and Math500 training sets. With this approach, the refined paraphrasing prompt helps us reliably produce high-quality paraphrased questions.

D Algorithm

The complete algorithm of our proposed PPCV framework is illustrated in Algo. 1.

E Discussions

Our technical contributions differ from prior works in three distinct ways. First, prior works (Zhou et al., 2024; Chen et al., 2024b; Yadav et al., 2024) typically use paraphrasing merely to expand the solution space. In contrast, we introduce Paraphrastic Probing, a mechanism that uses paraphrasing to test the model's internal confidence. By analyzing the discrepancy in token-level logits of the initial trajectory between the original and paraphrased questions, we can rigorously pinpoint the critical tokens that may lead to errors in the following steps.This transforms paraphrasing from a generation tool into a precise, token-level diagnostic tool. Second, prior works (Zhou et al., 2024; Chen et al., 2024b) typically rely on simple majority voting across multiple solutions. Our Paraphrase Consistency mechanism is technically distinct. It validates answers based on their robustness across semantic variations of the problem constraint. We further introduce a similarity-weighted consistency metric that weighs answers based on the linguistic diversity of the paraphrase, offering a more nuanced selection criterion than simple frequency counts. At last, a major technical limitation in current reasoning research is the reliance on external models or human-

Algorithm 1 PPCV Framework

Input: LLM; original question q_0; number of paraphrased questions N; number of sampled alternative tokens K; temperature-scaling coefficient λ.

Output: Final answer ans_f.

(Paraphrasing) Generate paraphrased variants $q_1, q_2, \ldots, q_N$ from q_0 using the LLM.

(Initial reasoning) Obtain the initial reasoning path $r_0^{q_0}$ for q_0.

(Token influence analysis) For each paraphrased question q_n, compute the token distribution P^{q_n} by concatenating q_n with $r_0^{q_0}$ (Eq. 1).

(Candidate selection) Identify candidate critical token positions based on the mismatch between the top-1 predicted tokens and the expected tokens in q_0 (Eq. 2).

(Critical token verification) Select the final critical token a_c using the criteria (Eqs. 3–5).

(Alternative token sampling) Sample K alternative tokens $a_c^0, a_c^1, \ldots, a_c^{K-1}$ (including $a_c^0 = a_c$) using top-K sampling on q_0.

(Truncated rollouts) Truncate the initial reasoning path $r_0^{q_0}$ at position a_c and form synthetic inputs $\tilde{r}_c^0, \tilde{r}_c^1, \ldots, \tilde{r}_c^{K-1}$ by appending each alternative token.

for each synthetic input $\tilde{r}_c^k$, $k = 0, \ldots, K-1$ **do**

  Generate rollouts $r_k^{q_0}, r_k^{q_1}, \ldots, r_k^{q_N}$ for the original and paraphrased questions using the LLM.

end for

(Consistency verification) Compute the final answer ans_f using (similarity-weighted) paraphrase consistency (Eq. 6).

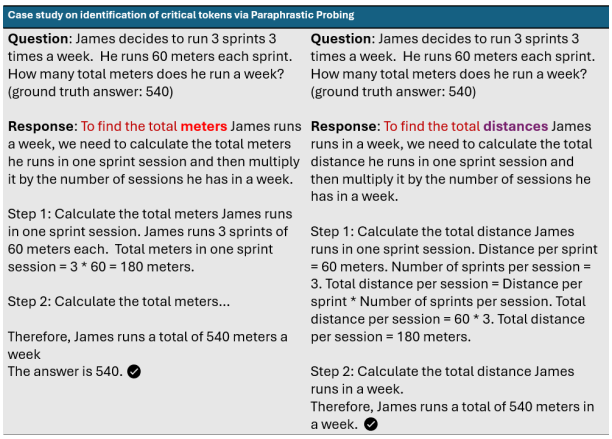


Figure 8: Effect of the critical token on an initial reasoning path that yields the correct answer. Critical tokens are highlighted in bold red, and alternative tokens in bold purple.

annotated error steps. Our method contributes a fully self-contained pipeline that identifies and corrects errors using the model's own sensitivity to surface-form perturbations.

Besides, prior work (Lin et al., 2025) introduces the concept of critical tokens, their identification primarily relies on brute-force rollout evaluation, and the proposed contrastive estimation lacks a clear inference-time criterion. In contrast, our method introduces a paraphrastic probing mechanism that leverages surface-form sensitivity to efficiently and explicitly identify critical tokens.

Unlike Self-Para-Consistency (Chen et al., 2024b), which uses paraphrases simply for sampling diversity, our method leverages them both to identify critical tokens and to verify reasoning robustness through a consistency mechanism. Overall, our contribution is not only a combination of techniques, but a new perspective that connects paraphrastic sensitivity with reasoning stability, enabling both more efficient diagnosis and more reliable refinement of reasoning.

Besides, although we select the top candidate for the primary experiments to maintain computational efficiency, the framework itself naturally extends to the multi–critical-token setting. For multiple critical tokens, we can generate alternative tokens for each identified position and apply paraphrase consistency across the new rollouts. This allows the model to refine multiple segments of its intermediate reasoning steps rather than only one.

To address potential ties when multiple candidates achieve the same consistency score, we introduce a similarity-weighted consistency scheme inspired by weighted majority voting. Specifically, we assign a weight to each paraphrased question based on its similarity to the original question, and compute a weighted consistency score accordingly. Intuitively, paraphrases that are less similar to the original question provide stronger evidence of robustness, as they introduce greater surface-form variation. Therefore, they are assigned higher weights. This allows the method to prioritize answers that remain consistent across more diverse formulations of the problem. As a result, the similarity-weighted consistency mechanism effectively resolves tie cases and selects answers that exhibit stronger robustness to paraphrastic perturbations, leading to more reliable final predictions.

Table 6: Comparison between our method and baselines with DeepSeek-R1-Distill-Llama-70B model on mathematical reasoning benchmarks using pass@1 accuracy.

Method	AIME2024	AIME2025	BRUMO2025	HMMT2025
Chain-of-Thought	56.67	38.00	43.33	30.00
Predictive Decoding	60.00	40.66	44.66	30.66
Phi-Decoding	64.00	46.67	48.00	31.33
PPCV (Ours)	**70.00**	**56.66**	**56.66**	**33.33**

Table 7: Comparison of our method and the baseline approach on Llama-3.1-8B-Instruct, evaluated using pass@k (k=4).

Method	GSM8K	GSM-Hard	Math500	SVAMP	ARC
Phi-Decoding	92.15	53.57	52.60	91.19	90.73
PPCV (Ours)	**93.83**	**61.41**	**59.39**	**94.48**	**94.24**

We have shown that replacing critical tokens can correct intermediate reasoning paths and lead to the correct answer. In this section, we examine how our method influences reasoning paths that are already correct. First, we conduct a case study on an instance with a correct answer, where we follow our pipeline to identify the critical token and roll out new reasoning paths using alternative tokens. As illustrated in Figure 8, the new rollouts also yield the correct answer. Second, our pipeline incorporates both the initial reasoning path $r_0^{q_0}$ and its paraphrased variants $r_0^{q_n}$ for evaluation. The robustness of correct rollouts across paraphrased forms ensures high consistency scores, allowing them to stand out as the final answer. These findings suggest that our pipeline preserves the performance of LLMs on problems that can already be solved correctly by CoT (Wei et al., 2022).

F Main results

To further assess the effectiveness of our method, we evaluate it on a larger reasoning model such as DeepSeek-R1-Distill-Llama-70B. We apply our full pipeline and compare against all baselines across multiple mathematical reasoning benchmarks. The results, presented in Table 6, show that our method consistently outperforms the baselines. These improvements demonstrate that our approach remains effective for stronger reasoning models and generalizes well beyond the smaller models.

we also conduct additional experiments to report pass@k performance for both our method and the baselines. In these experiments, we use Phi-Decoding as the representative baseline and evaluate on two models: Llama-3.1-8B-Instruct and Qwen3-32B. We set k= 4 and generate 12 samples per question to obtain stable estimates of pass@4. The results across multiple benchmarks are presented in Table 7 and Table 8, respectively. Consistent with our main findings, our method achieves higher pass@k scores compared to the baseline methods, indicating that paraphrastic critical-token refinement continues to provide benefits in a multi-sample setting. These results further validate the robustness of our approach under stochastic sampling and demonstrate improved hit rates when multiple outputs are available.

Table 8: Comparison of our method and the baseline approach on Qwen3-32B, evaluated using pass@k (k=4).

Method	AIME2024	AIME2025	BRUMO2025	HMMT2025
Phi-Decoding	41.61	30.19	45.09	13.11
PPCV (Ours)	**49.71**	**34.28**	**51.42**	**19.28**

Table 9: The comparison of performance between Paraphrased Majority Voting (PMV) and our proposed PPCV using pass@1 accuracy.

Method	GSM8K	GSM-Hard	Math500	SVAMP	ARC
Chain-of-Thought	77.40	28.00	31.00	83.00	58.91
PMV	83.55	33.16	40.60	87.70	61.63
PPCV (Ours)	**88.24**	**49.73**	**50.00**	**89.60**	**88.31**

G Ablation Study

We also perform a comparison against a Paraphrased Majority Voting (PMV) strategy is essential to demonstrate that the performance gains of our method are indeed derived from the critical token correction, rather than simply the benefit of having multiple paraphrased samples. Thus, we have performed the suggested additional experiment. We adopt Paraphrased Majority Voting (PMV) as the baseline. For a fair comparison, this approach uses the same total number of trajectories generated from the paraphrased questions as PPCV, and reaches the final answer using standard majority voting over the rollouts from the paraphrased questions. Specifically, we use four variants of each question, with 10 rollouts per variant. We compare the PMV baseline against our full PPCV method and ensure the same number of paraphrased questions as in our method. The results shown in Table 9 demonstrate that the performance yielded by PMV is significantly lower than that of our proposed PPCV method. This result confirms that simply rolling out trajectories from paraphrased questions and aggregating the answers is insufficient. The substantial performance gap between PMV and PPCV decisively highlights

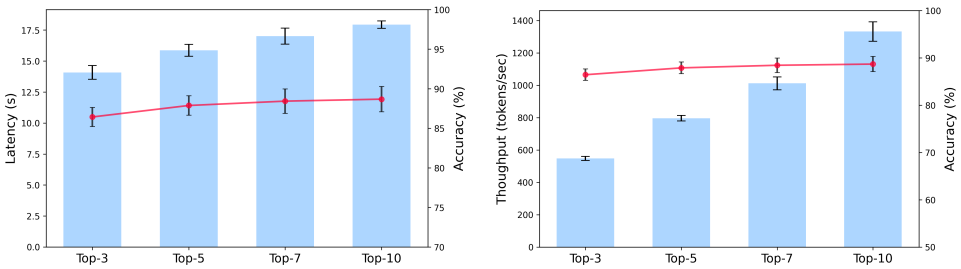
the necessity and effectiveness of our extracted critical tokens and the subsequent corrective reasoning step.

H Computational Cost Analysis

We also conduct a trade-off analysis between performance and efficiency by varying the number of alternative tokens considered at the critical position. Specifically, we test top-3, top-5, top-7, and top-10 alternatives using GSM8K, sampling 200 random questions and repeating the experiment five times to compute confidence intervals. The results are presented in Figure 9. We observe a clear trade-off. Reducing the number of alternative tokens lowers both latency and throughput, while causing a slight reduction in accuracy. This provides a practical mechanism for adjusting performance-efficiency trade-offs in real deployments. Depending on resource availability and target accuracy, practitioners can choose the appropriate number of alternative tokens.

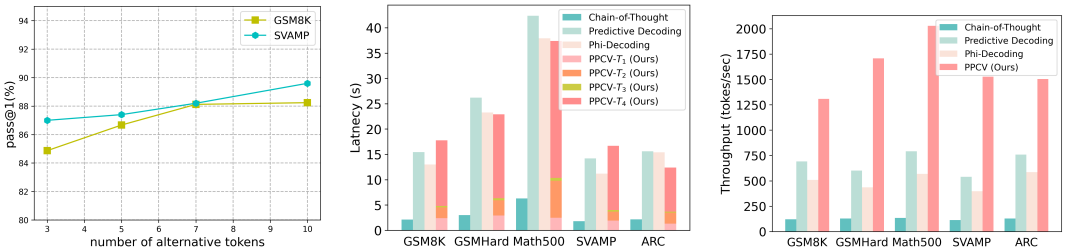
I Use of Large Language Models

In our work, we focus on improving large Malaguena models, and LLMs are used in our experimental evaluation. In addition, we employ LLMs to check grammatical errors and refine sentence structures.



(a) (b)

Figure 9: Trade-off between performance and efficiency for our method under different numbers of top-k tokens (k = 3, 5, 7, 10), measured by latency (left) and throughput (right).



(a) (b) (c)

Figure 10: (a) The impact of number of sampled alternative tokens on the performance. (b) Latency comparison between the baselines and our method, measured as the average inference time per question (in seconds). T_1, T_2, T_3, T_4 denote time for paraphrased question generation, initial answer generation, forward pass and new rollouts from truncated trajectories. (c) Throughput comparison between the baselines and our method, measured in output tokens per second.