RASC: Reasoning-Aware Self-Consistency for Efficient and Faithful LLM Reasoning

Anonymous ACL submission

Abstract

Self-Consistency mitigates hallucinations in Large Language Models (LLMs) by sampling multiple reasoning paths, but it lacks a systematic approach to determine the optimal number of samples or select the most faithful rationale. To address this limitation, we introduce Reasoning-Aware Self-Consistency (RASC), a novel framework that enhances sampling efficiency and reasoning faithfulness by dynamically evaluating both outputs and rationales. RASC assesses the quality of reasoning and the consistency of answers for each generated sample, using these assessments to guide early stopping decisions and rationale selection.The framework employs criteria-based stopping and weighted majority voting, enabling more informed choices on when to halt sampling and which rationale to select. Our comprehensive experiments across diverse question-answering datasets demonstrate that RASC outperforms existing methods, reducing sample usage by approximately 70% while maintaining accuracy. Moreover, RASC facilitates the selection of high-fidelity rationales, thereby improving the faithfulness of LLM outputs. Our approach effectively addresses the efficiency-accuracy trade-off in LLM reasoning tasks, offering a new perspective for more nuanced, faithful, and effective utilization of LLMs in resource-constrained environments. [1]

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable complex reasoning capabilities, particularly when employing chain-of-thought (CoT) prompting (Wei et al., 2022) which allows LLMs to generate reasoning paths (RPs) before answers. Building upon this, Wang et al. (2022) introduced Self-Consistency (SC), a decoding strategy that significantly enhances reasoning performance by sampling multiple RPs and marginalizing over these samples. This approach has substantially improved LLM's performance across various reasoning domains (Tan et al., 2023; Ahn et al., 2024). However, SC's effectiveness is tied to the number of reasoning paths sampled, creating a critical trade-off between performance and computational cost. While the original paper show that sampling more paths (e.g., 40) generally leads to better performance, there's no systematic method to determine the optimal number of samples for any given task. This ambiguity often results in over-sampling, leading to excessive computational demands (Li et al., 2024b). Thus, we need more efficient methods that can maintain SC's reasoning benefits while reducing its computational burden.

Recent research has introduced early-stopping mechanisms to address the computational costs of Self-Consistency. Adaptive Consistency (ASC) (Aggarwal et al., 2023) employs incremental sampling, terminating when a clear majority emerges based on a predefined probability distribution, while Early Stopping Self-Consistency (ESC) (Li et al., 2024b) segments the preset sample size into windows and stops sampling when answers within a window achieve uniformity. Although these methods effectively reduce computational demands, they fail to address two critical limitations in the original self-consistency approach: (1) they treat all samples equally for sampling decisions, basing the stopping criteria solely on the consistency of final answers, with this uniform treatment extending to the use of majority voting; (2) these approaches lack of a formal mechanism to distinguish and prioritize the most faithful and high-quality rationale among the selected samples, which can be critical in domains that require precise explanations such as healthcare (Wang et al., 2024).

To address these limitations while preserving the benefits of self-consistency, we propose Reasoning-

[1] Code available at: https://anonymous.4open.science/r/SC_conf-2D4B/README.md

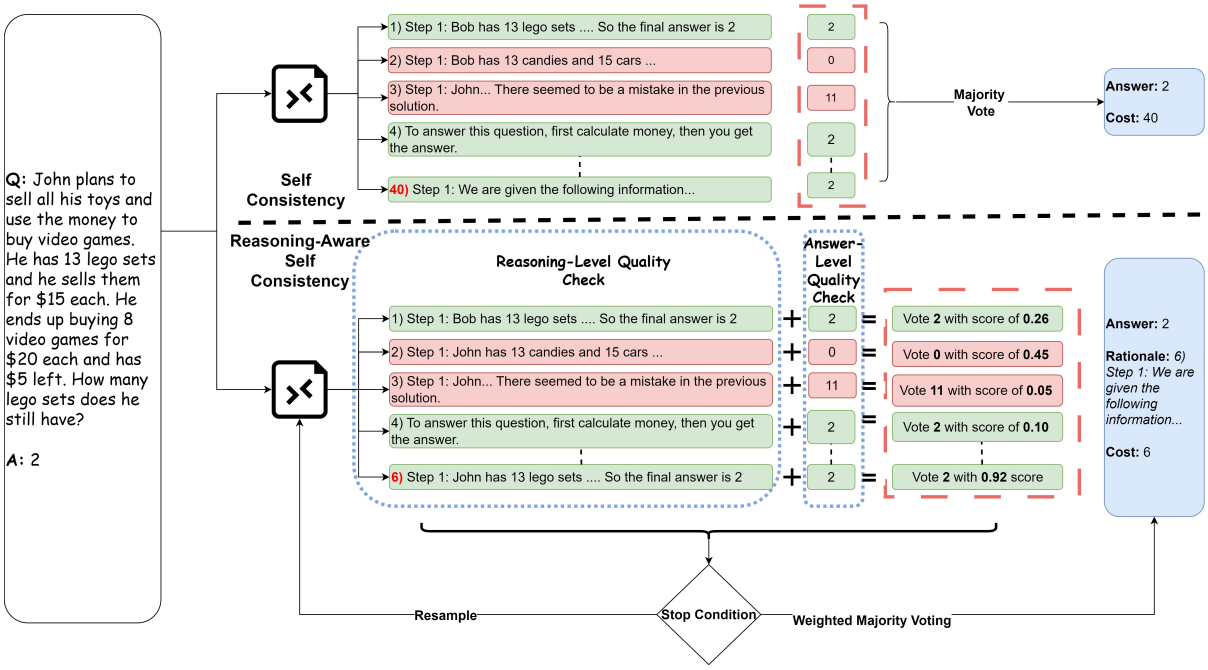

Figure 1: Comparison between Self Consistency and Reasoning-Aware Self Consistency (RASC) methods for a reasoning problem. The top half illustrates the standard Self Consistency approach, generating multiple reasoning paths and using majority voting for the final answer. The bottom half demonstrates RASC's improvements: it assigns scores based on the qualities of both answers and reasoning paths, effectively handling incorrect or irrelevant responses. RASC's resampling and stop condition mechanisms optimize sampling, enhancing efficiency and maintaining accuracy. Moreover, RASC enables the selection of the most faithful rationale among generated samples, improving the quality and reliability of the reasoning process in complex tasks

Aware Self-Consistency (RASC), a framework that enhances the original SC approach. As illustrated in Fig. 1, our method assesses both the rationales across samples and the consistency of final answers, which enables more nuanced and efficient sampling. RASC is motivated by the principle that higher-quality rationale often leads to better final answers (Khan et al., 2024). By leveraging this insight, RASC prioritizes robust reasoning in decision-making and enables early stopping when a consistent pattern of high-quality RP and consistency of answers is observed. This approach not only improves efficiency but also allows RASC to identify the most faithful and high-quality chains of thought among selected samples. Our approach significantly outperforms existing methods across 10 datasets spanning Commonsense, Mathematical, and Symbolic reasoning, reducing sample usage by 60%-80% while maintaining or improving accuracy. Furthermore, experiments demonstrate RASC's ability to select high-fidelity chain-of-thought reasoning, as well as its robustness across various datasets, prompting strategies, and multiple base LLM models, highlighting its adaptability to different efficiency-accuracy trade-offs and application needs. This combination of efficient sampling, high-fidelity reasoning selection, and broad applicability positions RASC as a versatile and powerful tool for enhancing the reasoning capabilities of large language models. In summary, our key contributions are as follows:

- **Analysis of Sampling-Based LLM Reasoning Techniques**: We present the first systematic evaluation of current SC-based sampling methods for LLM reasoning, offering new insights into their limitations, including the disregard for reasoning quality, uniform sample weighting, and the inability to identify the most faithful rationale.

- **Reasoning-Aware Self-Consistency (RASC) Framework**: We propose a novel framework that accounts for both rationale quality and answer consistency to dynamically select the number of samples and the best rationale for sampling-based LLM reasoning .

- **Robust Evaluation**: We showed RASC outperforms existing techniques in efficiency and explanation quality across various settings.

2 Related Work

2.1 Self-Consistency and Sampling Methods

Self-Consistency (SC) (Wang et al., 2022) mitigates LLM hallucinations by sampling multiple reasoning paths and selecting the most frequent solution, but incurs significant computational costs. Recent research has focused on reducing these costs through early-stopping approaches: Adaptive Consistency (ASC) (Aggarwal et al., 2023) employs incremental sampling, terminating when a clear majority emerges based on a predefined probability distribution, while Early Stopping Self Consistency (ESC) (Li et al., 2024b) segments the preset sample size into windows and stops sampling when answers within a window achieve uniformity. Although these methods effectively reduce computational demands, they primarily focus on checking the consistency of answers but overlook sample quality variations and the reasoning process's importance. Our work extends these approaches by addressing such limitations, aiming to enhance both the computational efficiency and faithfulness of self-consistency techniques.

2.2 Chain-of-Thought Prompting and Reasoning Process

Chain-of-Thought (CoT) prompting (Wei et al., 2022) has significantly enhanced LLM performance through step-by-step reasoning, with variants (Kojima et al., 2022; Zhou et al., 2023) further refining this approach. Recent work has sparked the idea that higher-quality intermediate reasoning paths lead to better final answers (Khan et al., 2024), concerns about the faithfulness of CoT-generated explanations have also emerged, particularly the risk of hallucination snowballing (Zhang et al., 2024). This has prompted a research theme on how to evaluate the intermedaite reasoning paths (Lightman et al., 2024; Radhakrishnan et al., 2023; Golovneva et al., 2023) and developing methods for reasoning path correction for better reasoning (Wan et al., 2024; Miao et al., 2024; Weng et al., 2023).Building on these advancements, our work introduces RASC, which adapts CoT evaluation frameworks to the self-consistency setting to dynamically enhancing the reliability and faithfulness of SC-based reasoning.

3 Reasoning-Aware Self-Consistency

Self-Consistency (SC) and its variants enhance question-answering accuracy in Large Language Models (LLMs) by generating multiple answers, but often overlook reasoning processes, leading to inefficient sampling. We propose Reasoning-Aware Self-Consistency (RASC), a framework that optimizes sampling efficiency while maintaining accuracy by evaluating both reasoning paths and answers. RASC introduces a sufficiency scoring function that combines reasoning quality and answer consistency features, mapped to a continuous score via an optimized classifier. The framework employs a dynamic sampling process with a buffer of high-quality samples, stopping when a predefined capacity is reached. Finally, RASC uses weighted majority voting based on the scores to select the final answer and most faithful reasoning path. This approach integrates reasoning evaluation into the self-consistency process, offering a more faithful and computationally efficient method for complex question-answering tasks in LLMs.

3.1 Reasoning and Answer Level Features

We formalize our sufficiency scoring approach by defining a function $F : X \to [0, 1]$, where X represents the space of all possible reasoning-answer pairs. This score is a composite measure of reasoning quality and answer consistency, indicating whether it's sufficient to terminate sampling. For a given reasoning-answer pair $x = (r, a) \in X$, we approximate $F(x)$ through a composition of functions: $F(x) \approx f_{\theta^*}(\phi(x))$. Here, $\phi : X \to \mathbb{R}^n$ is a feature extraction function mapping the reasoning-answer pair to an n-dimensional real-valued vector, and $f_{\theta^*} : \mathbb{R}^n \to [0, 1]$ is an optimized classifier that outputs a continuous sufficiency score given the n-dimensional input value. To capture both reasoning quality and answer consistency, we design ϕ to extract two sets of features:

Reasoning Quality Features: These features assess the quality of the RPs generated through CoT prompting. As shown in Table 1, these include measures such as RP-Length, Num-of-Steps, Step-Relevance, and Question-Relevance. These features capture the coherence, relevance, and depth of the reasoning process.

Answer Consistency Features: These features, including Local-Consistency and Global-Consistency as shown in Table 1, evaluate the consistency of answers in the generated samples.

The feature extraction function ϕ now combines both Reasoning Quality and Answer Consistency features into a single vector: $\phi(x_i) = [\text{Local Consistency}(x_i), ..., \text{Global Relevance}(x_i)]$

Table 1: Answer-Level and Reasoning-Level Features for RASC. These features were designed based on our preliminary analysis (Appendix A.8) and insights from previous literature (Jin et al., 2024; Li et al., 2024a; Zhang et al., 2023; Huang et al., 2023; Hosking et al., 2023; Bang, 2023; Tu et al., 2020; Golovneva et al., 2023). They capture various aspects of reasoning quality and answer consistency. Symbols in parentheses indicate relationships between samples: *-Consistency targets exact matches, while -Relevance considers vocabulary overlap. This comprehensive set of features enables RASC to make informed decisions about sampling sufficiency. Refer to Table 17 in appendix for specific examples.

Feature	Description
Answer-Level Features	
Local-Consistency ($a_t \leftrightarrow a_{t-1}$)	Check if the current answer is the same as the prior answer.
Global-Consistency ($a_t \leftrightarrow a_1, ..., a_{t-1}$)	Check if the current answer is within the previous answers.
Reasoning-Level Features	
RP-Length	Reasoning path string length.
Num-of-Steps (r_t)	Number of steps in reasoning paths.
Step-Relevance (r_t)	The coherence between the reasoning steps (number of overlapping words).
Question-Relevance ($r_t \leftrightarrow Q$)	The similarity between the input question Q and the reasoning path.
Error-Admitting (r_t)	The LLM acknowledges making mistakes during the response.
Local-Relevance ($r_t \leftrightarrow r_{t-1}$)	The similarity between the current and previous generated reasoning paths.
Global-Relevance ($r_t \leftrightarrow r_1, ..., r_{t-1}$)	The similarity between the concatenation of all previous sample RPs and the current sample RP.

Feature Set Applicability and Extensibility: Our feature set provides a strong foundation for evaluating reasoning quality and answer consistency. We designed ϕ with a modular architecture to allow future integration of additional features as our understanding evolves. Importantly, these features are intentionally lightweight, avoiding computationally intensive extractors to optimize the cost of SC while maintaining effectiveness.

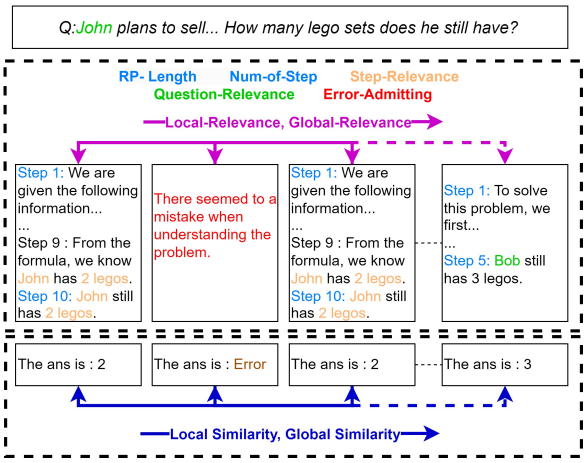

Figure 2: Reasoning-Aware quality features. Different colour corresponds to different feature visualizations. Ans: Output Answer. The upper dashed box is the reasoning-level features, and the lower dashed box is the answer-level features. The blue and purple arrows represent relative relations between different samples (consistency and relevance).

3.2 Sampling and Decision Process

Sufficiency Score Computation: We learn a scoring function $f : \mathbb{R}^n \rightarrow [0, 1]$ from a family of parameterized functions $\mathcal{F}$, where each function is denoted as f_θ for parameters θ. Given a dataset $\mathcal{D} = (x_i, y_i)_{i=1}^M$, where $x_i = \phi(r_i, a_i)$ is the feature vector of the i-th reasoning-answer pair (r_i, a_i), and $y_i \in 0, 1$ indicates the correctness of the answer, we obtain the optimal parameters θ^* by minimizing:

$$\theta^* = \arg\min_\theta \left(\frac{1}{M} \sum_{i=1}^{M} \mathcal{L}(f_\theta(x_i), y_i) \right) \quad (1)$$

Where $\mathcal{L}$ represents the loss function (e.g., cross-entropy).

Stop Condition: For a sequence of K sampled reasoning-answer pairs $x_i{}_{i=1}^K$, we compute sufficiency scores $SS_i = f_\theta(\phi(x_i))$ using the optimized function f_θ. We maintain a buffer $\mathcal{B}$ of high-quality pairs with sufficiency scores above a predefined threshold T:

$$\mathcal{B} = (r_i, a_i) \mid SS_i \geq T \quad (2)$$

The sampling process terminates when $|\mathcal{B}| \geq N$, where N is a predefined capacity.

Key Trade-off Parameters: N and T are crucial for balancing sample reduction and accuracy. N determines the number of high-quality samples required before stopping, while T sets the threshold for considering a sample as high-quality. Adjusting these parameters allows RASC to prioritize either greater sample reduction or higher accuracy.

Weighted Sampling for Answer and Rationale: Upon reaching the stop condition, RASC determines the final answer and reasoning path through weighted majority voting using the sufficiency scores of pairs in $\mathcal{B}$:

$$(\text{Ans}, \text{RP}) = \left(\arg\max_{a \in \mathcal{A}} \left(\sum_{(r_i, a_i) \in \mathcal{B}, a_i = a} SS_i \right), \arg\max_{r_j : (r_j, \text{Ans}) \in \mathcal{B}} SS_j \right) \quad (3)$$

where $\mathcal{A}$ represents the set of possible answers, $\mathcal{B}$ is the buffer of high-quality reasoning-answer pairs, and SS_i is the sufficiency score for the i-th pair. This approach ensures that higher-quality reasoning-answer pairs have greater influence on the final decision. The answer (Ans) with the highest weighted sum of sufficiency scores is selected, and then the most faithful reasoning path (RP) is chosen as the one with the highest sufficiency score among those supporting the selected answer. For complete details, refer to Algorithm 1.

Algorithm 1 RASC Early Stopping

Input: Query Q, Threshold T, Buffer Size N, Scoring Function F, Base LLM $\mathcal{M}$
Output: Final Answer A_{best}, Best Reasoning Apth RP_{best}
$B \leftarrow \{\}$ ## Initialize empty buffer
while $|B| < N$ **do**
  $(A_i, RP_i) \leftarrow \mathcal{M}(Q)$
  $SS_i \leftarrow F(A_i, RP_i)$
  if $SS_i \geq T$ **then**
    $B \leftarrow B \cup \{(A_i, RP_i, SS_i)\}$
  end if
end while
$\mathcal{A} \leftarrow \{A | (A, RP, SS) \in B\}$
$A_{best} \leftarrow \arg\max_{A \in \mathcal{A}} \sum_{(A', RP, SS) \in B : A' = A} SS$
$RP_{best} \leftarrow \arg\max_{RP} \{SS | (A_{best}, RP, SS) \in B\}$
return A_{best}, RP_{best}

4 Experiments

4.1 Experimetnal Setup

Baseline and Data We compare `RASC` against three established baseline methods: `SC` (Wang et al., 2022), `ASC` (Aggarwal et al., 2023), and `ESC` (Li et al., 2024b). Following the previous work, our main evaluation dataset comprises 16,725 samples spanning commonsense reasoning (Date Understanding, CommonsenseQA), symbolic reasoning (boolean expressions, disambiguation, last letters), and mathematical reasoning (GSM8K, ASDIV, MathQA, SVAMP). To assess generalizability on unseen data, we use out-of-distribution datasets including BigBench (bench authors, 2023) and StrategyQA (Geva et al., 2021), comprising an additional 10,824 examples that require implicit reasoning steps across various topics. See the Appendix for the configurations of the baseline methods (A.4) and the data (A.6).

Implementation Details: Our experiments utilize `LLAMA2-7B` (Meta, 2024), `GPT3.5-turbo/GPT4` (OpenAI, 2024), and `Vicuna-13B` (Chiang et al., 2023). For parameters of our algorithm, we make buffer size N being `5` with a quality threshold T of `0.5` based on experimental results in Figure 3. We use customized metrics that measure the trade-off between efficiency and accuracy to select the best hyper-parameters. We use `Logistic Regression` as a lightweight sufficiency scoring model, which shows a comparable correlation with human judgments and efficiency on running time. LLM inference is applied with a temperature of `0.5`. See Appendix A for further details on experimental setup, including computational resources, model configurations, and prompts. Additional results, including analysis of various base models and Pearson correlation of other NLG metrics with human juedgement, can be found in Appendix B.

4.2 Main Results

RASC Excels Across Diverse Reasoning Tasks and LLMs: Table 2 showcases RASC's performance across a wide spectrum of reasoning tasks and base Language Models (LLMs). Our comprehensive analysis reveals that RASC consistently achieves superior sample efficiency while maintaining similar accuracy across mathematical, commonsense, and symbolic reasoning tasks. This superior performance and efficiency are statistically significant across multiple benchmarks and models, as detailed in Appendix B.2. The robustness of

Table 2: Number of Generated Samples and Accuracy of Different Methods and Models on Various Benchmarks. Mathematical Reasoning: GSM8K, ASDIV, MathQA; Commonsense Reasoning: Date Understanding, CommonsenseQA; Symbolic Reasoning: Boolean Expressions, Disambiguation QA, Last Letter Concatenation. Methods: SC (Self-Consistency), ESC (Early-Stopping Self-Consistency), ASC (Adaptive Self-Consistency), RASC (Reasoning-Aware Self-Consistency). # of Gen: Average number of generated samples. Acc: Accuracy (%). Percentage changes are relative to SC.

Model	Method	Benchmark Category					
		Mathematical Reasoning		Commonsense Reasoning		Symbolic Reasoning	
		# of Gen	Acc	# of Gen	Acc	# of Gen	Acc
GPT-4	SC	40	87.5%	40	88.0%	40	97.3%
	ESC	6.86 (-82.9%)	87.3%	7.93 (-80.2%)	88.5%	5.54 (-86.2%)	97.3%
	ASC	5.89 (-85.3%)	87.3%	6.15 (-84.6%)	87.2%	4.43 (-88.9%)	97.3%
	RASC	**4.59** (-88.5%)	87.5%	**4.74** (-88.1%)	88.3%	**4.19** (-89.5%)	97.3%
GPT-3.5 Turbo	SC	40	69.1%	40	76.1%	40	85.3%
	ESC	14.94 (-62.7%)	69.3%	10.86 (-72.9%)	76.4%	7.55 (-81.1%)	86.0%
	ASC	12.49 (-68.8%)	67.6%	8.34 (-79.2%)	75.0%	6.39 (-84.0%)	84.7%
	RASC	**6.62** (-83.5%)	69.4%	**4.95** (-87.6%)	76.1%	**4.36** (-89.1%)	85.3%
Vicuna-13B	SC	40	41.7%	40	55.9%	40	47.3%
	ESC	20.29 (-49.3%)	40.5%	20.89 (-47.8%)	50.5%	23.05 (-42.4%)	44.0%
	ASC	17.44 (-56.4%)	42.3%	19.92 (-50.2%)	55.9%	25.77 (-35.6%)	49.3%
	RASC	**8.24** (-79.4%)	42.0%	**7.83** (-80.4%)	55.0%	**8.71** (-78.2%)	45.3%
Llama2-7B	SC	40	23.0%	40	68.9%	40	24.2%
	ESC	27.25 (-31.9%)	23.5%	8.85 (-77.9%)	68.9%	31.03 (-22.4%)	24.8%
	ASC	23.91 (-40.2%)	21.5%	7.97 (-80.1%)	68.3%	24.62 (-38.5%)	24.2%
	RASC	**10.71** (-73.2%)	23.2%	**5.11** (-87.2%)	68.9%	**13.09** (-67.3%)	24.8%

performance of RASC is further highlighted by its consistent effectiveness across diverse LLMs, from closed-source models like GPT-3.5 Turbo and GPT-4 to open-source alternatives such as Vicuna-13B and Llama2-7B.

Table 3: Performance and Time Analysis of Methods Using GPT-4. Analysis of other models and costs can be found in Appendix B.3.

Method	**Accuracy (%)**	**Inference Time (s)**	**Non-Inference Time (s)**	**Total Time (s)**
SC	90.9	398.9	0.01	398.9
ESC	91.0	67.8	0.54	68.3
ASC	90.6	54.9	0.40	55.3
RASC	**91.0**	**45.1**	2.05	**47.2**

Note: Values are averages per question over all data.

RASC Excels in Computational Efficiency: Table 3 empirically highlights RASC's improvement in computational efficiency over other methods. While SC incurs minimum additional computational overhead, it requires the longest inference time. ASC and ESC, on the other hand, introduce moderate overhead due to their stopping criteria calculations. RASC, although introducing the highest non-inference processing time (2 seconds per question) for feature extraction, substantially reduces the overall inference time, which typically constitutes the largest portion of the total computational cost. Notably, RASC reduces inference time by approximately 80% compared to SC, while maintaining similar accuracy, and outperforming other efficient methods. We thus claim that as long as the non-inference overhead remains relatively smaller than the savings in inference time, RASC will consistently prove more computationally efficient. These gains hold across other evaluated models and is especially significant when inference speed is slower, as demonstrated in Table 11 in the Appendix.

4.3 Analysis

Robust Performance and Tunable Efficiency: Fig. 3 shows RASC's performance across various threshold (T) and buffer size (N) configurations. RASC demonstrates a unique combination of performance stability and customizable efficiency. Accuracy improves consistently as T increases from

Table 4: Performance comparison of different feature sets across reasoning tasks. Answer-level features include final answer characteristics. Reasoning-level features capture the quality of the reasoning process. Combined Features incorporate both answer-level and reasoning-level features. See Table 1 for more feature details.

Feature Set	Mathematical Reasoning		Symbolic Reasoning		CommonSense Reasoning	
	# Gen	Acc (%)	# Gen	Acc (%)	# Gen	Acc (%)
Answer-level only	10.86	55.5	7.75	58.0	6.65	74.9
Reasoning-level only	5.50	50.4	4.30	55.9	3.08	72.4
Combined Features	**8.20**	**55.8**	**6.28**	**58.1**	**5.63**	**75.0**

Results are averaged across all datasets and models. # Gen: Average number of generated samples. Acc: Accuracy.

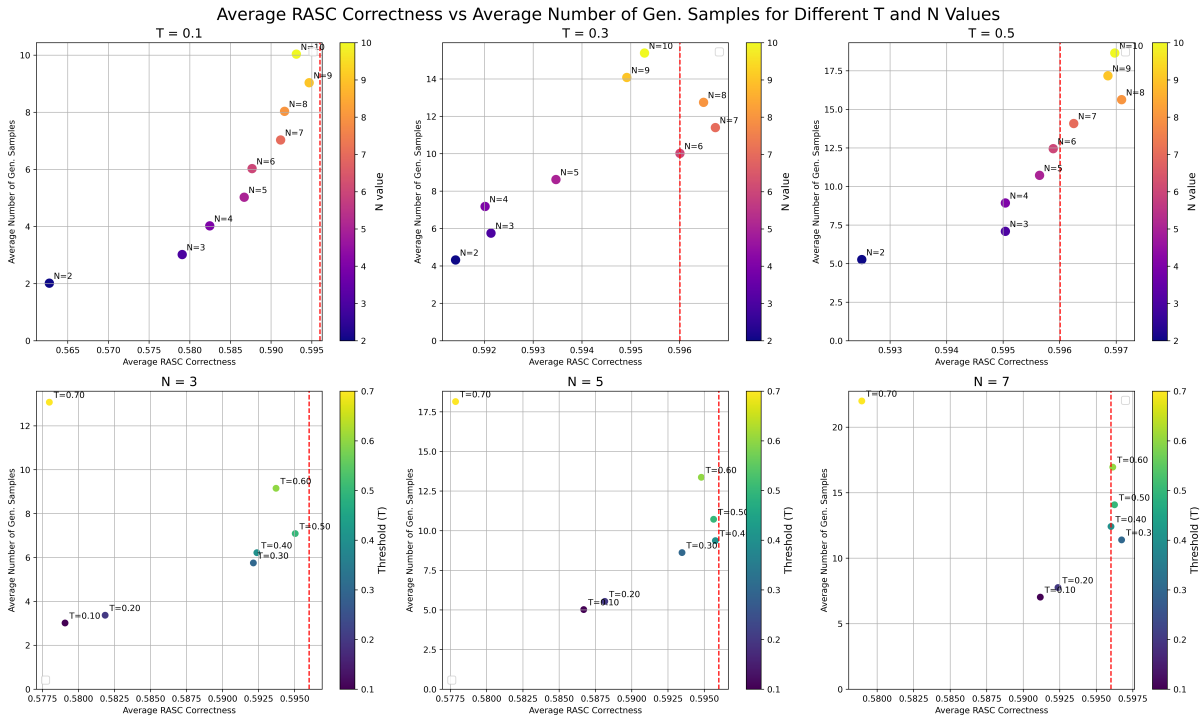

Figure 3: Effects of varying N and T on performance tradeoffs, illustrating how changes in these parameters impact both accuracy (RASC Correctness) and the average number of samples generated.

0.1 to 0.5, with the most significant gains between $T = 0.1$ and $T = 0.3$. Increasing N from 3 to 7 generally enhances accuracy, albeit with diminishing returns. This stability across hyperparameter values showcases RASC's robustness. RASC allows preference-based tuning: lower T and N values prioritize efficiency with minimal accuracy loss, ideal for resource-constrained scenarios, while higher values maximize accuracy at increased computational cost. This dual characteristic of robustness and adaptability enables RASC to optimize for specific accuracy or efficiency requirements across various computational environments.

Individual Impact of Reasoning-Level and Answer-Level Features: Our comprehensive ablation studies reveal the crucial interplay between reasoning-level and answer-level features across various reasoning tasks. Table 4 demonstrates that ***integrating both feature types consistently outperforms models using either feature set alone regarding the trade-off between accuracy and number of samples generated, validating our hypothesis of their complementary nature.*** While answer-level features provide a strong baseline, as established in previous work, our novel incorporation of reasoning-level features, providing additional insights form the intermediate reasoning, significantly enhances the model's capabilities in distinguishing better samples, thus reducing the number of generations needed. This synergistic combination enables a more holistic capture of the reasoning process, highlighting the importance of considering both the reasoning path and the final answer in evaluating and optimizing language model outputs.

Selection of High-Fidelity Chains of Thought: Table 5 demonstrates RASC's superiority over Self-

Consistency (SC) in rationale selection across traditional metrics and manual evaluation, with a 0.7-point increase in Human-Eval Score on a 5-point scale. Qualitative analysis in Appendix also supports this: for a travel distance question (Table 8, Sample 1), RASC provides a clear, correct explanation (human scores: 5) while SC introduces ambiguity (scores: 2). This analysis shows ***RASC's superior capability in selecting faithful and high-quality Chains of Thought (CoTs)***, consistently producing more accurate, coherent, and relevant explanations for improved problem-solving.

Table 5: Comparison of RASC and SC on CoT Faithfulness. This evaluation used 200 randomly selected questions from our training set. For RASC, we picked the CoT with the highest sufficiency score, and for SC, we selected a random CoT sample that produced the correct final answer. The evaluation assessed BARTScore, CTC, BLURT, and human-evaluated faithfulness scores. For BARTScore, CTC, and BLURT, we manually created golden CoTs as reference.

Metric	RASC	SC	Diff.
BARTScore (Yuan et al., 2021)	0.61	0.39	+0.22
CTC (Deng et al., 2021)	0.55	0.45	+0.10
BLURT (Sellam et al., 2020)	0.58	0.42	+0.16
Human-Eval Score[a]	4.7	4.0	+0.70

[a]On a scale of 1-5. All other scores range 0-1, representing ranked-based scores where 1 indicates better metrics for RASC over SC and 0 means the other way around. See Appendix A.11 for details on human evaluation criteria and golden CoT creation process.

4.4 Robustness and Generalizability

Table 6: Impact of Different Prompting Strategies on Performance on 100 Random Question from GSM8K dataset, a subset data from Mathematical Reasoning, with GPT-3.5 Turbo (a/b: a = accuracy (%), b = avg. samples)

Method	Zero-shot	Few-shot	Least-to-Most
SC	69.0 / 40.0	75.0 / 40.0	85.0 / 40.0
ESC	67.0 / 9.3	75.0 / 9.5	85.0 / 8.8
ASC	69.0 / 8.8	74.0 / 8.4	85.0 / 7.0
RASC	**69.0 / 5.1**	**75.0 / 5.3**	85.0 / **5.0**

RASC is consistent on Different Prompts: To check RASC's robustness to different input prompts, we evaluated RASC's performance using three distinct CoT prompting strategies: zero-shot (Kojima et al., 2022), few-shot (Wei et al., 2022), and least-to-most (Zhou et al., 2023). As Table.6 shows, RASC consistently outperforms other methods across all prompting strategies, demonstrating higher accuracy and lower sample (5 steps on average) requirements. This consistency highlights RASC's adaptability to various ways of eliciting reasoning from language models.

Cross-Domain Adaptability and Performance Scaling: To assess RASC's generalizability, we evaluated it on out-of-distribution datasets, including BigBench (bench authors, 2023) and StrategyQA (Geva et al., 2021) comprising 10,824 examples requiring implicit reasoning steps across various topics. Using GPT3.5 turbo as the base model, RASC demonstrated robust performance on these unseen, complex datasets. As illustrated in Figure 6 in Appendix, **RASC achieved the highest accuracy while significantly reducing the average number of samples generated compared to other methods**. Notably, RASC outperformed SC in accuracy (0.581 vs 0.579) while using only 14.5% of the samples (5.8 vs 40.0). It also surpassed both ESC and ASC in both accuracy and efficiency. This performance suggests RASC's resilience across varied data distributions, highlighting its ability to generalize effectively to novel reasoning tasks without compromising accuracy or computational efficiency.

5 Conclusion

We have introduced Reasoning-Aware Self-Consistency (RASC), an innovative framework that enhances the efficiency and faithfulness of large language model reasoning. By dynamically evaluating both reasoning paths and final answers, RASC significantly reduces the computational cost of self-consistency methods without compromising performance and enables more informed sampling decisions and rationale selection. Empirical results across diverse reasoning tasks and multiple language models demonstrate that RASC reduces sample usage by 60-80% while maintaining accuracy compared to existing methods. Furthermore, RASC shows superior capability in selecting high-fidelity chains of thought, enhancing the explainability of model outputs. The framework's adaptability to various prompting strategies and out-of-domain datasets underscores its robustness and practical applicability. RASC thus presents a promising direction for optimizing the reasoning capabilities of large language models, simultaneously advancing computational efficiency, accuracy, and output faithfulness in a unified approach.

6 Limitations

While RASC demonstrates significant improvements in efficiency and faithfulness across various reasoning tasks, it is important to acknowledge some limitations and areas for future research:

- **Feature Engineering Dependency:** RASC relies on a set of manually designed features to evaluate reasoning quality. While these features have proven effective empirically, they may not capture all aspects of high-quality reasoning across all domains. Future work could explore more advanced feature extraction techniques, possibly leveraging unsupervised learning or transfer learning approaches to identify relevant features automatically.

- **Faithfulness evaluation:** While we have shown improvements in faithfulness of the selected chain of thought reasoning, the evaluation relies on automated metrics and limited human evaluation. A more extensive human evaluation could provide stronger evidence on our work.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, et al. 2023. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. Available online. Accessed: 2024-06-01.

Fu Bang. 2023. Gptcache: An open-source semantic cache for llm applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 212–218.

BIG bench authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Mingkai Deng, Bowen Tan, Zhengzhong Liu, Eric Xing, and Zhiting Hu. 2021. Compression, transduction, and creation: A unified framework for evaluating natural language generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7580–7605, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Preprint*, arXiv:2101.02235.

Olga Golovneva, Moya Peng Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansy, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Matias. 2022. True: Re-evaluating factual consistency evaluation. *Preprint*, arXiv:2204.04991.

Tom Hosking, Phil Blunsom, and Max Bartolo. 2023. Human feedback is not gold standard. *arXiv preprint arXiv:2309.16349*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Mingyu Jin, Qinkai Yu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, Mengnan Du, et al. 2024. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, and Tim Rocktäschel. 2024. Debating with more persuasive llms leads to more truthful answers. In *Proceedings of the International Conference on Machine Learning (ICML)*.

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.

Junlong Li, Fan Zhou, Shichao Sun, Yikai Zhang, Hai Zhao, and Pengfei Liu. 2024a. Dissecting human and llm preferences. *arXiv preprint arXiv:2402.11296*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024b. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Meta. 2024. Meetllama3. Blog post. Available: https://llama.meta.com/llama3/.

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*.

OpenAI. 2024. Introducing gpt-3.5 turbo. Blog post. Available: https://platform.openai.com/docs/models/gpt-3-5-turbo.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Sam McCandlish, Sheer El Showk, Tamera Lanham, Tim Maxwell, Venkatesa Chandrasekaran, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. 2023. Question decomposition improves the faithfulness of model-generated reasoning. *Preprint*, arXiv:2307.11768.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of ACL*.

Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Can chatgpt replace traditional kbqa models? an in-depth analysis of the question answering performance of the gpt llm family. In *International Semantic Web Conference*, pages 348–367. Springer.

Zhengkai Tu, Wei Yang, Zihang Fu, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2020. Approximate nearest neighbor search and lightweight dense vector reranking in multi-stage retrieval architectures. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*, pages 97–100.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2024. Cot rerailer: Enhancing the reliability of large language models in complex reasoning tasks through error detection and correction. *Preprint*, arXiv:2408.13940.

Li Wang, Xi Chen, XiangWen Deng, Hao Wen, MingKe You, WeiZhi Liu, Qi Li, and Jian Li. 2024. Prompt engineering in consistency and reliability with the evidence-based guideline for llms. *npj Digital Medicine*, 7(1):41.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277. Curran Associates, Inc.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah Smith. 2024. How language model hallucinations can snowball. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, Honolulu, Hawaii. PMLR.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. *Preprint*, arXiv:2205.10625.

A Appendix: Experimental Details

A.1 Base Models

The language models used in our experiments have varying numbers of parameters:

- `LLAMA2-7B`: This model has 8 billion parameters (Meta, 2024).
- `GPT3.5-turbo`: The exact number of parameters for this model is not publicly disclosed by OpenAI (OpenAI, 2024).
- `Claude-3-Haiku`: The number of parameters for this model is not publicly available from Anthropic (Anthropic, 2024).
- `Vicuna-13B`: This model has 13 billion parameters (Chiang et al., 2023).

A.2 Computational Resources and Infrastructure

Our experiments were conducted using various models and computing resources. Below, we detail the computational budget for each model and the specifications of our computing infrastructure.

Computational Budget

- `LLAMA2-7B`: We consumed approximately 100 GPU hours for generating all the collected data, running the model, generating CoT samples, and processing the results.
- `GPT3.5-turbo`: Approximately $500 was spent on API calls to OpenAI, covering all experiments including data generation, CoT samples, and result processing.
- `Claude-3-Haiku`: The total cost for API calls to Anthropic was approximately $300.
- `Vicuna-13B`: We utilized approximately 150 GPU hours for experiments, including experimental running, inference on CoT samples, and data processing.

Computing Infrastructure Our experiments were conducted on a computing infrastructure equipped with the following hardware:

- GPU: NVIDIA GeForce RTX 3070 Ti
- CPU: 16 cores 11th Generation Intel Core i7 Processors
- RAM: 64GB DDR4
- Storage: 1TB NVMe SSD

This infrastructure provided the necessary computational power and storage capacity to efficiently run our experiments across various models and datasets.

A.3 Model Configurations

- **Temperature**: For all LLM calls, we used a temperature setting of 0.7 to balance creativity and coherence in the generated outputs.
- **Max Tokens**: The maximum number of tokens for each response was set to 1024 for consistency across models.
- **Top-p (nucleus sampling)**: We used a top-p value of 0.95 for all model calls to ensure diverse yet relevant outputs.

A.4 Algorithm Configurations

- **ESC Algorithm**: We used a window size of 5 for the Entropy-based Stopping Criteria (ESC) algorithm.
- **Adaptive Consistency Algorithm**: We used the default settings with stopping_criteria=BetaStoppingCriteria(0.95) and max_gens = 40.

A.5 Used Packages

In this study, several online available Python packages are used to conduct experiments and analysis:

- NLTK: For calculating Jaccard Similarity, Ngram, tokenizer
- statistics: For computing logistic regression
- PyTorch: For using pre-trained LLM
- pandas: For data manipulation
- json: Loading and saving JSON data
- sklearn: For supervised learning models training and evaluation
- adaptive_consistency: For implementing adaptive consistency algorithm
- Levenshtein: For computing Levenshtein distance
- transformers: For Hugging Face pre-trained model usage
- LangChain: For LLM API usage and answer parser

A.6 Datasets

To assess the generalizability and robustness of our proposed Reasoning-Aware Self-Consistency (RASC) method, we utilize a diverse range of datasets across multiple reasoning categories. Our evaluation leverages Question Answering datasets spanning three main categories:

Mathematical Reasoning: GSM8K, ASDIV, MathQA, SVAMP

Commonsense Reasoning: Date Understanding, CommonsenseQA

Symbolic Reasoning: Boolean Expressions, Disambiguation, Last Letters

These datasets provide a comprehensive foundation for evaluating our method across various problem domains and task types.

Out-of-Domain (OOD) Data: To rigorously test the adaptability and robustness of our RASC method, we incorporate out-of-domain (OOD) data, specifically StrategyQA (Geva et al., 2021) and BigBench (bench authors, 2023). StrategyQA is a question answering benchmark designed for multi-hop reasoning, featuring:

- 2,780 examples requiring implicit reasoning steps.
- Diverse, short questions covering a wide range of strategies.
- Annotations including reasoning step decomposition and supporting evidence.
- A novel data collection procedure to ensure challenging and diverse questions.

BigBench is a comprehensive benchmark consisting of:

- 204 tasks contributed by 450 authors across 132 institutions.
- Diverse topics including linguistics, math, common-sense reasoning, and social bias.
- Tasks designed to be beyond the capabilities of current language models.
- Evaluations across various model scales and architectures.

By using these datasets, we aim to evaluate RASC's performance on questions that require complex, multi-step reasoning strategies not explicitly stated in the question. This allows us to verify the performance of our method on unseen data that demands sophisticated reasoning capabilities, providing insights into its real-world applicability and resilience across different computational environments and language model architectures. Using GPT3.5 turbo as the base model, RASC demonstrated robust performance on these unseen, complex datasets, comprising a total of 10,824 examples requiring implicit reasoning steps across various topics.

A.7 Data Processing

We used a custom data preprocessing pipeline implemented in Python to clean and prepare the input data for our experiments:

- For text normalization, we employed NLTK's word_tokenize and WordNetLemmatizer.
- We used the json_parser module to parse the results and extract quality features from the model outputs.
- Our preprocessing pipeline includes steps for handling missing data, removing duplicates, and standardizing text formats.

For more detailed information on our data processing techniques, please refer to our GitHub repository https://anonymous.4open.science/r/SC_conf-2D4B/README.md.

A.8 Common LLM Errors and Feature Selection

A.8.1 Common LLM Errors

As a crucial part of the solution, a faithful reasoning path is even more important than simply getting the correct answer to the question. Therefore, it is crucial for sampling methods, such as self-consistency, to consider the content quality of the reasoning path itself. To better understand the types of errors that LLM often makes during the response, we utilize common mathematical reasoning and commonsense reasoning datasets to systematically summarize the common errors. We start with the common errors observed by Golovneva et al. (2023) and form an initial set of potential quality features defined in Table 17. After human evaluation and statistical analysis, the most frequent mistakes that a bad reasoning path has include inconsistency with the question, lack of step coherence (lack of a logical flow), calculation mistakes, and hallucinations. In addition, we observe that LLMs are more likely to hallucinate when the generated context gets significantly longer. In addition, we observe that the RP always leads to an incorrect answer when it 'admits' that a mistake has been made in any of the proposed steps. However, the purpose of this study is not to comprehensively analyzing common mistakes or design automated metrics, we purposely exclude features that require intensive computational cost, such as features that have to be extracted by utilizing external machine learning models (SenteceBert, Transformers, or other neural network-based feature extractors).

A.8.2 Answer-level Features

These features include measures such as consistency between consecutive answers (Local-Consistency), consistency with the most common previous answer (Global-Consistency),

and detection of deviations from expected response formats (Parsing-Error).

A.8.3 Reasoning-level Features

Our novel contribution lies in introducing reasoning-level features, which capture the intrinsic qualities of each Reasoning Path (RP). These features evaluate various aspects of the RP, such as the length and number of reasoning steps (RP-Length, Num-of-Steps), the logical flow between steps (Step-Relevance), and the relevance to the input (Question-Relevance). We also include time depedent features that compare the current RP with previous ones (Local-Consistency, Global-Consistency) and detect self-acknowledged mistakes or uncertainties (Error-Admitting). Refer to Table 17 for more examples. While similar work like ROSCOE (Golovneva et al., 2023) introduces sets of reasoning-level features, our approach extends this by incorporating time-dependent features that compare the current RP with previous ones, allowing us to evaluate the consistency and progression of reasoning across multiple steps

A.9 Preliminary Feature Analysis

Table 7 presents the correlation coefficients between each feature and the Correctness variable, along with the p-values from t-tests.

Feature	Correlation	P-value
Local-Consistency	0.367	0.00
Global-Consistency	0.403	0.00
RP-Length	-0.0861	0.00
Num-of-Steps	-0.0476	0.00
Step-Relevance	-0.0264	1.25e-169
Question-Relevance	-0.0423	0.00
Error-Admitting	-0.0492	0.00
Local-Relevance	0.0530	0.00
Global-Relevance	0.0850	0.00

Table 7: Correlation with Correctness and P-values (3 significant figures)

Global-Consistency (0.403) and Local-Consistency (0.367) show the strongest positive correlations with Correctness, suggesting these similarity measures between the actual solution and generated content are most predictive of correctness. All features show statistically significant differences between correct and incorrect instances (p-values ≈ 0). RP-Length, Error-Admitting, Question-Relevance, Num-of-Steps, and Step-Relevance have weak negative correlations, indicating that longer solutions or those with more steps are slightly less likely to be correct. Word clouds of internal mistakes (Figures 4 and 5) reveal that both LLAMA3 and GPT-3.5-TURBO models frequently acknowledge potential errors and limitations, using phrases like "There seemed to be a mistake" and "Not enough information". These findings suggest that improving the model's ability to generate solutions similar to the actual solution and encouraging concise responses could enhance performance, while the models' self-awareness of mistakes could be leveraged to improve reliability.

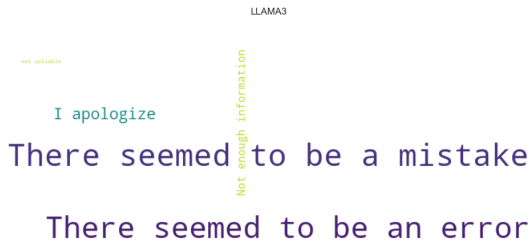

Figure 4: Word Cloud of Internal Mistakes for LLAMA3

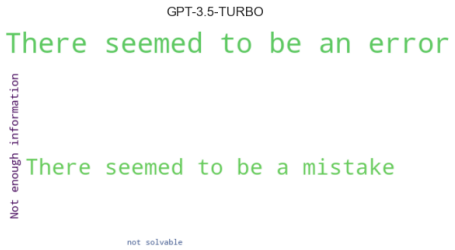

Figure 5: Word Cloud of Internal Mistakes for GPT-3.5-TURBO

A.10 Sample Generation Prompts

In this study, we explore three different prompting strategies: Chain of Thought (CoT), Few-Shot, and Least-to-Most. Each strategy aims to enhance the performance of Large Language Models (LLMs) in generating reasoning paths (RPs) for given questions.

A.10.1 Chain of Thought (CoT) Prompting

CoT prompting explicitly requires LLMs to reason through the question *step by step*. We utilize zero-shot prompting to generate RPs in our datasets. The prompt is defined as follows:

System Message: *You are a professional specialized in {subject}. You need to help me answer the given question. Notice that you need to solve the question step by step and as detailed as possible. Do not jump to the answer directly. You must follow the RP format instructions.*
Human Message: *{question}*

A.10.2 Few-Shot Prompting

Few-shot prompting provides the LLM with examples of how to solve problems using step-by-step reasoning. This approach helps guide the model in generating structured and detailed responses. The prompt is structured as follows:

System Message: *You are a professional specialized in {subject}. Your task is to answer the given question using step-by-step reasoning. Follow the examples provided, breaking down your thought process into clear steps before providing the final answer.*
Human Message: *Here are two examples of how to solve problems using step-by-step reasoning:*
[Example 1 and Example 2 are provided here, demonstrating step-by-step problem-solving]
Now, please solve the following question using a similar step-by-step approach: {question}

A.10.3 Least-to-Most Prompting

The Least-to-Most approach involves breaking down complex problems into simpler sub-problems, solving them in order of increasing complexity, and using those solutions to address the main question. This strategy is particularly useful for tackling intricate problems. The prompt is structured as follows:

System Message: *You are an expert problem solver specialized in {subject}. Your task is to break down and solve complex problems using the Least-to-Most approach. This means you'll divide the main problem into simpler sub-problems, solve them in order of increasing complexity, and use those solutions to address the main question.*
Human Message: *Let's solve problems using the Least-to-Most approach. Here's an example:*
[An example demonstrating the Least-to-Most approach is provided here]
Now, please use this Least-to-Most approach to solve the following problem. Break it down into simpler sub-problems, solve them in order, and then use those solutions to address the main question: {question}

Each of these prompting strategies aims to elicit detailed, step-by-step reasoning from the LLM, enabling us to generate high-quality reasoning paths for analysis and comparison.

A.11 Human Evaluation Criteria

Annotator Profile: The evaluation was conducted by a group of 10 individuals: 7 males and 3 females. The age range of the annotators was 23-30 years old, with a median age of 25. All annotators hold technical degrees spanning mathematics, computer science, and statistics, ensuring their capability to comprehend the logic behind the generated rationales effectively. Their educational backgrounds are as follows:

- 1 Assistant Professors (1 in Computer Science)
- 6 PhD candidates (1 in Computer Science, 4 in Data Science, 1 in Statistics)
- 3 Master's students (1 in Data Science, 1 in Electrical Engineering, 1 in Computational Mathematics)

Geographically, the annotators represent two countries:

- 3 from the United States
- 7 from China

Professionally, 8 are currently in academia (1 assistant professors, 5 PhD candidates, 2 Master's students), while 2 work in industry with Master's degrees and 1-3 years of relevant work experience related to technology.

Ethical Considerations and Consent: Prior to participating in the evaluation process, all annotators were provided with information detailed the

study's objectives, the nature of their participation, and how their data would be utilized. Specifically, annotators were informed that their ratings, demographic information, and any generated "golden" CoTs would be used for research purposes and potentially published anonymously. They were assured of the confidentiality of their personal information and the anonymization of their responses in any resulting publications. Annotators were also informed of their right to withdraw from the study at any point and request the removal of their data. The data retention policy, including the duration of data storage and the method of eventual deletion, was clearly outlined. Furthermore, annotators were provided with contact information for the research team to address any concerns or questions about their data usage.

Evaluation Instruction: Human evaluators were asked to rate the generated Chain of Thought (CoT) responses on two main criteria:

1. **Overall Quality** [1-5]: This criterion assesses whether the generated response answers the question in a well-justified manner. The scale is interpreted as follows:
   - 1: Incomprehensible and completely incorrect. The response is unintelligible or entirely unrelated to the question.
   - 2: Mostly incorrect with major flaws in reasoning. The response may be partially related to the question but contains significant errors or misunderstandings.
   - 3: Partially correct but with notable gaps or minor errors. The response addresses the question but lacks full justification or contains some inaccuracies.
   - 4: Mostly correct with minor imperfections. The response is well-justified and accurate, with only slight room for improvement.
   - 5: Clear, correct, and fully justified. The response comprehensively answers the question with sound reasoning and no errors.
2. **Coherency** [1-5]: This criterion evaluates whether the whole generated response makes sense, regardless of its correctness in addressing the context. The scale is interpreted as follows:
   - 1: Completely incoherent. The response is a jumble of words or concepts with no logical flow or structure.
   - 2: Mostly incoherent with occasional comprehensible parts. The response has severe issues with logical flow, making it very difficult to follow.
   - 3: Partially coherent but with noticeable lapses in logic or structure. The response has a basic structure but contains confusing or disconnected elements.
   - 4: Mostly coherent with minor clarity issues. The response has a clear logical flow with only slight ambiguities or structural weaknesses.
   - 5: Perfectly coherent and easy to understand. The response has a clear, logical structure with smooth transitions between ideas.

The overall score for each evaluated response is calculated as the average of these two metrics.

In addition to manually evaluating the best CoT responses (based on our method) and a randomly chosen CoT from SC, evaluators were also tasked with creating a "golden" CoT when necessary so that we have a reference available to calculate the BARTScore (Yuan et al., 2021). This process involved:

1. Reviewing a random CoT for human-like reasoning.
2. Assessing whether the flow of logic closely resembles what a human solver would do, considering:
   - Natural, step-by-step reasoning processes
   - Logical order of steps
   - Inclusion of relevant intermediate calculations or considerations
   - Appropriate level of detail for human-like problem-solving
3. Rewriting the CoT if it did not closely resemble human-like reasoning, ensuring:
   - Maintenance of the correct answer while improving the reasoning process
   - Structure reflecting human approach to problem-solving
   - Clear, concise, and easy-to-follow response

This comprehensive evaluation process ensures a thorough assessment of the quality and coherency of generated CoT responses, as well as the creation of human-like golden standards when needed. See Table 8 for two examples.

A.12 Evaluation Metric for Accuracy-Cost Trade-Off

To quantitatively assess the trade-off between accuracy and cost, we introduced a custom metric that balances both factors. This metric normalizes the accuracy and cost values and computes a weighted average to provide a single score representing the overall performance of a method.

Let acc denote the accuracy of a method, and cost denote the computational cost, measured as the number of samples generated. Additionally, let sc_acc and sc_cost represent the accuracy and cost of the Self-Consistency (SC) method, respectively. Let single_sample_acc and direct_cost denote the accuracy from using only the first sample's answer and the cost of the direct sampling method, respectively.

The metric is defined as:

$$\text{metric} = 0.5 \times \text{acc_factor} + 0.5 \times \text{cost_factor} \quad (4)$$

where acc_factor and cost_factor are normalized values of accuracy and cost, calculated as follows:

$$\text{acc_factor} = \begin{cases} 1 & \text{if acc} \geq \text{sc_acc} \\ 0 & \text{if acc} \leq \text{sam_acc} \\ \frac{\text{acc} - \text{sam_acc}}{\text{sc_acc} - \text{single_acc}} & \text{otherwise} \end{cases} \quad (5)$$

$$\text{cost_factor} = \begin{cases} 1 & \text{if cost} \leq \text{direct_cost} \\ 0 & \text{if cost} \geq \text{sc_cost} \\ \frac{\text{sc_cost} - \text{cost}}{\text{sc_cost} - \text{dir_cost}} & \text{otherwise} \end{cases} \quad (6)$$

The acc_factor is normalized to be between 0 and 1, where 1 corresponds to an accuracy greater than or equal to the SC method, and 0 corresponds to an accuracy less than or equal to the accuracy using just the first sample's answer. Similarly, the cost_factor is normalized to be between 0 and 1, where 1 corresponds to a cost less than or equal to the direct sampling method, and 0 corresponds to a cost greater than or equal to the SC method. The metric function calculates the weighted average of acc_factor and cost_factor, giving equal importance to both accuracy and cost. A higher value of the metric indicates a better trade-off between accuracy and cost. This function enables us to compare different methods and configurations based on their ability to achieve high accuracy while minimizing computational cost.

A.13 Reproducibility

To ensure reproducibility of our results:

- We set a fixed random seed (42) for all randomized processes.
- All experiments were run using Python 3.8.10 in a conda environment. The full list of dependencies and their versions can be found in the `requirements.txt` file in our repository.

B Appendix: Additional Results

B.1 Impact of Base Estimator:

The performance of different sufficiency scoring models, as summarized in Table 9, identifies Logistic Regression as the most effective model in terms of accuracy and sample utilization. This finding shows the importance of selecting an appropriate sufficiency scoring model to maximize the benefits of the RASC approach. These results are obtained by training the sufficiency scoring models on the training set and then fine-tuning the best set of hyperparameters (N and T) using the customized metrics. The final evaluation is performed on the test set to ensure the robustness of the model.

Among the sufficiency scoring models evaluated **Logistic Regression** achieves the highest accuracy of `46.0%` while requiring the lowest average number of samples (`5.87`) compared to other models. This indicates that it effectively captures the relationship between the features extracted from the RP and the likelihood of hallucination, which allows RASC to make informed decisions during the weighted majority voting process.

The accuracy of Random model drops below the baseline of Self-Consistency. The poor performance of the Random model, which assigns sufficiency scores randomly, emphasizes the importance of using a well-designed scoring model in the RASC approach. Without a meaningful assessment of the generated content's quality and consistency, the weighted majority voting process

Table 8: Comparison of Best CoT, SC CoT, and Golden CoT with Human Evaluation

Sample	Question	Answer	RASC Rationale	SC Rationale	Golden CoT	Ratings (Qua, Coh)	
						Best	SC
1	Elvis starts driving from his house and travels west for 5 hours. Then he turns around and travels east for 8 hours. If he was driving at an average speed of 18mph for both parts of the journey, how far is he from his house now?	54 miles	If Elvis traveled west for 5 hours at 18mph, he covered 5 x 18 = 90 miles. When he traveled east for 8 hours at 18mph, he covered 8 x 18 = 144 miles. The distance he is from his house now is the difference between the two distances, which is 144 - 90 = 54 miles. The answer is 54 miles.	If Elvis drove west for 5 hours at 18mph, he covered a distance of 5 * 18 = 90 miles. When he turned around and drove east for 8 hours at 18mph, he covered a distance of 8 * 18 = 144 miles. The total distance he covered is 90 + 144 = 234 miles. However, we do not know the direction of his house. If he is now east of his house, he is 144 - 90 = 54 miles away. If he is now west, he is 90 - 144 = -54 miles away. So, we need more information to determine the distance he is from his house.	Elvis drove for 5 hours at 18mph, so he covered 5 * 18 = 90 miles to the west. He then turns around and drives for 8 hours at 18mph, so he covers 8 * 18 = 144 miles to the east. The total distance from his house is 144 - 90 = 54 miles to the east. The answer is 54 miles.	5, 5	1, 3
2	Are the founders of Skype from Asia?	No	The founders of Skype are from Europe, not Asia. The founders are Estonians Ahti Heinla, Priit Kasesalu, and Jaan Tallinn. So the answer is no.	The founders of Skype are Niklas Zennström and Janus Friis, both of whom are from Europe (Sweden and Denmark, respectively). Therefore, the answer is no.	The founders of Skype are from Europe. The founders of Skype are Estonians Ahti Heinla and Priit Kasesalu. Estonia is a country located in Northern Europe, not Asia. So the answer is no.	4, 5	2, 4

Note: Qua = Quality, Coh = Coherency. Ratings are on a scale of 1-5, with 5 being the highest.

Table 9: Performance Comparison of Sufficiency Scoring Models

Model	**Accuracy (%)**	**Num. Samples**
Random	41.9	9.77
NB	45.9	7.91
LR	**46.0**	**7.87**
RF	45.8	8.38
HHEM	42.4	9.92

Table 10: *HHEM: Hallucination Detector Model based on microsoft/deberta-v3-base, initially trained on NLI data

cannot effectively distinguish between reliable and unreliable samples, leading to suboptimal results.

The HHEM Model, which is a DeBERTa-based hallucination detector sourced from Hugging Face (Honovich et al., 2022), does not perform as well as the other models in this context. This suggests that relying solely on a pre-trained model without considering the specific characteristics and requirements of the RASC approach may not yield the best results. The superior performance of models like Logistic Regression (LR), Naive Bayes (NB), and Random Forest (RF), which utilize manual feature engineering tailored to the RASC framework, highlights the importance of crafting features that capture the nuances of the generated content as we

discussed in the method section.

B.2 Statistical Significance

To rigorously evaluate the performance of RASC, we conducted statistical significance experiments across multiple benchmarks using various language models. The comparison between RASC (Reasoning-Aware Self-Consistency) and SC (Self-Consistency) methods reveals interesting patterns across different reasoning types. In terms of correctness, Tables 15 show no statistically significant differences between RASC and SC across common sense, mathematical, and symbolic reasoning tasks. However, the steps comparison in Tables 16 demonstrates that RASC consistently requires significantly fewer steps than SC across all reasoning types, with p-values of 0 indicating strong statistical significance. This holds the same for ASC and RASC comparison as demonstrated in Table 13 and Table 14 These results suggest that while RASC maintains comparable correctness to SC, it achieves this with substantially improved efficiency in terms of the number of steps required.

Table 11: Performance and Time Analysis of Methods Using Llama2.

Method	Accuracy (%)	Inference Time (s)	Non-Inference Time (s)	Total Time (s)
SC	38.7	560.00	0.01	560.01
ESC	39.1	313.32	0.40	313.72
ASC	38.0	263.62	0.54	264.16
RASC	39.0	**134.96**	2.05	**137.01**

Note: All values are averages per 1 question.

Table 12: Pearson Correlation of Different Metrics with Human Ratings

Metric	Pearson Correlation
RASC Score	0.68
BARTScore	0.52
BLURT	0.58
CTC	0.65

Note: Higher correlation indicates closer alignment with human judgments. All correlations are statistically significant ($p < 0.01$).

Note: SC stands for Self-Consistency, a baseline method for comparison.

B.3 Additional Performance and Time Analysis

This section focuses on the performance and time analysis of the open-source Llama2 model.

Table 11 compares different methods using the Llama2 model. RASC achieves the highest accuracy (38.86%) while significantly reducing inference time. Compared to SC, RASC improves accuracy by 0.33 percentage points and reduces total time by 66.23

These results demonstrate RASC's effectiveness in enhancing both performance and time efficiency. While we don't provide direct cost analysis for closed-source models, the substantial reduction in inference time achieved by RASC would likely translate to significant cost savings in scenarios where API calls are charged based on usage time.

B.4 Comparison with Existing NLG Metrics:

To evaluate the effectiveness of our sufficiency scoring method, we compared it against three popular automated metrics for natural language generation: BARTScore (Yuan et al., 2021), BLURT (Sellam et al., 2020), and CTC (Deng et al., 2021). Using the same 100 samples from the faithfulness analysis, we calculated the Pearson correlation between these scores and human-annotated quality ratings.

As shown in Table 12, RASC's sufficiency scores demonstrate a higher correlation with human ratings (0.68) compared to BARTScore (0.52), BLURT (0.58), and CTC (0.65). This suggests that our scoring method more closely aligns with human judgments of CoT quality in reasoning tasks, validating its effectiveness in assessing the quality of generated reasoning paths. The substantial improvement over BARTScore and the slight edge over BLURT and CTC underscore RASC's capability to capture nuances in reasoning quality that align more closely with human evaluations.

B.5 Test Data Results

To evaluate RASC's generalizability, Table 6 presents the performance comparison between RASC and other methods on this dataset.

Table 13: Mean Difference, P-value, 95% Confidence Interval, and Significance of Correctness Comparison between RACS and ASC Methods on Various Reasoning Types

Comparison	Mean Difference	P-value	95% CI	Significant
RACS vs ASC (common sense reasoning)	-0.0075	0.6697	[-0.0565, 0.0414]	No
RACS vs ASC (mathematical reasoning)	-0.004	0.745	[-0.0384, 0.0383]	No
RACS vs ASC (symbolic reasoning)	0.0033	0.9044	[-0.0738, 0.0805]	No

Table 14: Mean Difference, P-value, 95% Confidence Interval, and Significance of Steps Comparison between RACS and ASC Methods on Various Reasoning Types

Comparison	Mean Difference	P-value	95% CI	Significant
RACS vs ASC (common sense reasoning)	-3.4301	0	[-4.2164, -2.6439]	Yes
RACS vs ASC (mathematical reasoning)	-7.4156	0	[-8.0905, -6.7408]	Yes
RACS vs ASC (symbolic reasoning)	-7.7112	0	[-9.4171, -6.0052]	Yes

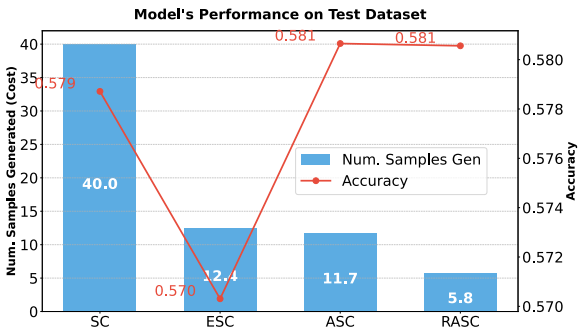

Figure 6: Performance of models on out-of-distribution datasets using GPT3.5 turbo, assessing method's robustness across diverse reasoning tasks.

Table 15: Mean Difference, P-value, 95% Confidence Interval, and Significance of Correctness Comparison between RASC and SC Methods on Various Reasoning Types

Comparison	Mean Difference	P-value	95% CI	Significant
RASC vs SC (common sense reasoning)	-0.0075	0.6697	[-0.0565, 0.0414]	No
RASC vs SC (mathematical reasoning)	-0.0068	0.5819	[-0.0412, 0.0275]	No
RASC vs SC (symbolic reasoning)	0.0067	0.8104	[-0.0705, 0.0839]	No

Table 16: Mean Difference, P-value, 95% Confidence Interval, and Significance of Steps Comparison between RASC and SC Methods on Various Reasoning Types

Comparison	Mean Difference	P-value	95% CI	Significant
RASC vs SC (common sense reasoning)	-34.7975	0	[-35.0068, -34.5882]	Yes
RASC vs SC (mathematical reasoning)	-32.5354	0	[-32.7298, -32.3409]	Yes
RASC vs SC (symbolic reasoning)	-32.4240	0	[-32.9474, -31.9007]	Yes

Table 17: Answer-Level and Reasoning-Level Feature Extraction Example.

Feature	Example
Answer-Level Features	
Local-Consistency ($a_t \leftrightarrow a_{t-1}$)	Ans1 = 3, Ans2 = 2 → Local-Consistency = 0; Ans1 = 3, Ans2 = 3 → Local-Consistency = 1
Global-Consistency ($a_t \leftrightarrow a_1, ..., a_{t-1}$)	Prev_ans = [A,B], Cur_ans = A → Global-Consistency = 1; Prev_ans = [A,B], Cur_ans = C → Global-Consistency = 0
Parsing-Error (a_t)	Ans: error → Parsing-Error = 1; Ans: 2 → Parsing-Error = 0
Reasoning-Level Features	
RP-Length	RP: "I love LLM" → RP-Length = 3
Num-of-Steps (r_t)	RP: "Step1:xxx, Step2:xxx" → Num-of-Steps = 2
Step-Relevance (r_t)	Step1: Paper reviewer loves LLMs; Step2: They use LLM to review my paper → Step-Relevance = 3/7
Question-Relevance ($r_t \leftrightarrow Q$)	Question: John loves lego, RP: Bob has 2 legos → Question-Relevance = 0.33; Question: John loves lego, RP: John has 2 legos → Question-Relevance = 0.66
Error-Admitting (r_t)	RP: It seems that I made a mistake in the previous steps → Error-Admitting = 1
Local-Relevance ($r_t \leftrightarrow r_{t-1}$)	RP1: I love LLM, RP2: I love LLM → Local-Relevance = 1; RP1: I love LLM, RP2: Bob hates LLM → Local-Relevance = 0.33
Global-Relevance ($r_t \leftrightarrow r_1, ..., r_{t-1}$)	RP1-2: [I love LLM, LLM is good], RP3: I love LLM → Global-Relevance = 0.67; RP1-2: [I love LLM, LLM is good], RP3: Life is good → Global-Relevance = 0.33