

# Boosting Self-Consistency with Ranking

Maria Marina<sup>1,2</sup>, Daniil Moskovskiy<sup>1,2</sup>, Sergey Pletenev<sup>1,2</sup>,  
Mikhail Salnikov<sup>1,2</sup>, Alexander Panchenko<sup>2,1</sup>, and Viktor Moskvoretskii<sup>3</sup>

<sup>1</sup>AIRI, <sup>2</sup>Skoltech, <sup>3</sup>EPFL

{Maria.Marina, A.Panchenko}@skol.tech

## Abstract

Self-consistency improves large language models by sampling multiple reasoning paths and selecting the most frequent answer, but majority voting often fails to recover correct answers that are already present among the samples. We address this limitation with *Ranking-Improved Self-Consistency* (RISC), which reformulates answer selection in self-consistency as a ranking problem. Instead of relying on a single uncertainty or confidence signal, RISC uses a lightweight LambdaRank model to score candidate answers with five carefully designed features that capture answer frequency, semantic centrality, and reasoning-trace consistency. We evaluate RISC on three datasets under a range of test-time budgets. Across datasets, RISC consistently achieves a better accuracy-efficiency trade-off than standard self-consistency and strong baselines, with particularly large gains on question answering benchmarks. Further analysis shows that the proposed features are individually useful and, more importantly, complementary, highlighting the value of learning to combine multiple informative signals for test-time answer selection.

## 1 Introduction

Test-time scaling has recently emerged as a promising direction for improving the performance of large language models (LLMs) (Snell et al., 2024; Muennighoff et al., 2025). Instead of relying solely on larger models or more training data, this paradigm improves performance by allocating additional compute during inference. Prior work has shown that additional computation can substantially improve performance through approaches such as extended reasoning (OpenAI, 2024; Guo et al., 2025), structured search methods like Tree-of-Thoughts (Yao et al., 2023), iterative self-correction (Kamoi et al., 2024; Moskvoretskii et al., 2025), and self-consistency decoding (Wang et al., 2023).

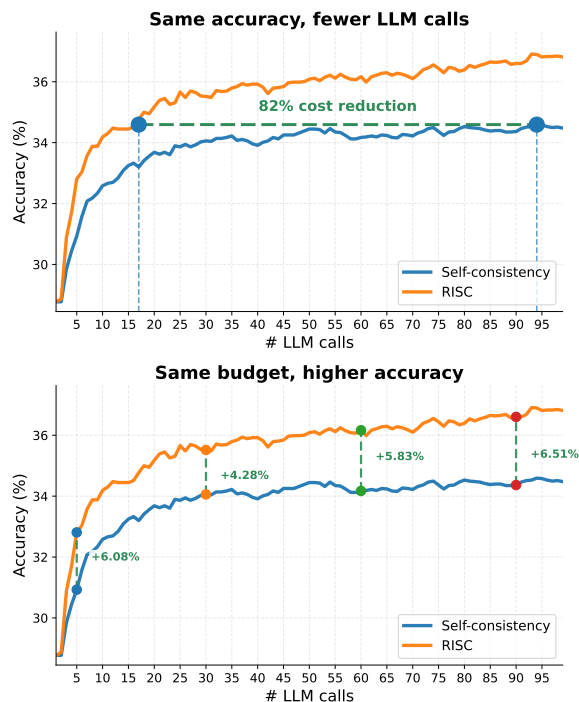


Figure 1: Accuracy versus the number of sampled responses on PopQA for self-consistency and RISC. RISC consistently achieves higher accuracy while substantially reducing computational cost: with only 18 samples, it already surpasses the performance of self-consistency with 99 samples. It also delivers systematic accuracy gains over self-consistency across the full range of LLM-call budgets.

In this paper, we focus on improving self-consistency, a test-time scaling technique where an LLM samples multiple candidate answers and selects the final prediction by majority vote (Wang et al., 2023). Despite its simplicity, self-consistency has proven highly effective while remaining computationally efficient and easy to apply across models, tasks, and hardware (Amballa et al., 2025; Min et al., 2024).

However, despite its strong performance, a large gap still remains between majority voting and an upper bound where an oracle selects the correct answer from the sampled candidates (Zhuang et al.,

2026; Dang et al., 2025). This gap suggests that the correct answer is often present among the samples, but the model is uncertain and fails to select it reliably. Several prior works have attempted to identify robust signals that help distinguish the correct answer among sampled candidates. Examples include CISC, which relies on verbal uncertainty estimates (Taubenfeld et al., 2025), and methods based on internal geometric structure, such as stable rank (Tang and Yang, 2025).

Nevertheless, these approaches rely on designing a single robust signal intended to capture the model’s parametric knowledge. While appealing in their simplicity, such hand-crafted, knowledge-driven methods often struggle to scale and generalize to new tasks, a pattern long observed in AI research (Sutton, 2019).

We therefore introduce **Ranking-Improved Self-Consistency (RISC)**, a lightweight extension of self-consistency. RISC treats answer selection among sampled candidates as a ranking task, where correct answers should appear higher in the output.

Under this formulation, RISC builds a lightweight ranker that scores candidates using a small set of interpretable features motivated by prior work. This design is aligned with recent QA research showing that lightweight question-level and external-knowledge signals can provide useful reliability information without relying on expensive additional LLM calls (Marina et al., 2025; Pletenev et al., 2025).

Our results show that RISC significantly outperforms previous methods, while remaining generalizable and performing strongly on challenging test-time scaling tasks such as question answering. We further analyze its behavior and demonstrate that each feature contributes meaningful improvements on its own, while their combination leads to additional performance gains.

The contributions of this paper are as follows:

1. We propose RISC, which reformulates answer selection in self-consistency as a *ranking problem*, and show that RISC consistently outperforms majority voting and strong confidence-based baselines across three benchmarks.
2. We design five interpretable features capturing answer frequency, semantic centrality, and reasoning-trace consistency, and show they are individually useful and mutually complementary.
3. We provide a detailed analysis via feature ablations and SHAP, revealing non-additive interac-

tions between features that explain why a learned combination outperforms any single handcrafted signal.

We make code publicly available.<sup>1</sup>

## 2 Related Work

**Test-Time Compute Scaling.** Recent research has shown that allocating additional computation at inference time can significantly improve the performance of large language models without modifying their parameters (Snell et al., 2024). This family of methods includes parallel strategies that generate multiple candidate outputs and select or combine them. These techniques have been successful on various tasks across mathematical reasoning (Lightman et al., 2024), code generation (Jimenez et al., 2024), and open-ended question answering (Chen et al., 2023). Our work falls within the parallel scaling paradigm: generate  $N$  candidate responses and select the best candidate.

**Self-Consistency and Majority Voting.** Self-consistency (Wang et al., 2023) is the most widely used parallel selection strategy: it samples multiple reasoning paths at different temperatures and selects the most frequent final answer by majority vote. It is a strong baseline because diversity across chains can cancel errors. Several extensions have been proposed, including weighted voting based on verbalized or confidence scores and ranked voting methods that use preferences (Chen et al., 2023; Wang et al., 2025). Yet a persistent gap remains between majority-vote accuracy and *oracle selection* (Wu et al., 2025), the accuracy achievable if one could always pick the correct candidate whenever it is present. This suggests that LLMs often generate correct answers that are not the plurality vote, motivating stronger selection criteria.

**Ranking and Learning to Rank.** Learning-to-rank methods (Burgess et al., 2006; Liu, 2009) have been widely applied in NLP, from passage retrieval to candidate selection in language models. Cobbe et al. (2021) train a verifier to score math solutions, and Lightman et al. (2024) extend this with process reward models that evaluate reasoning step by step. In best-of- $N$  selection (Snell et al., 2024), a separately trained scorer ranks  $N$  candidates and returns the highest-scoring one.

Our approach is different: we score candidates *relative to the full candidate set* rather than indepen-

<sup>1</sup><https://github.com/s-nlp/RISC>

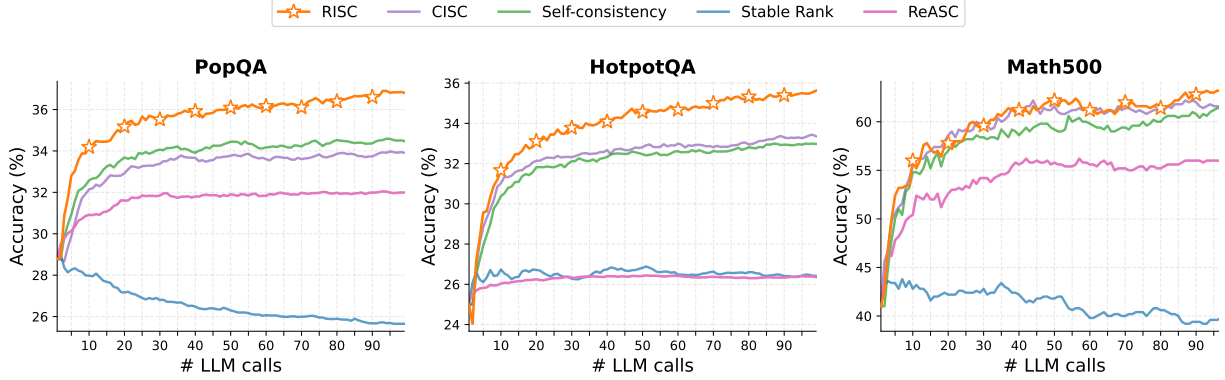


Figure 2: Comparison of RISC against Self-Consistency, Stable Rank, ReASC, and CISC on three datasets for Llama-3.1-8B-Instruct model. RISC consistently outperforms the baselines on the QA datasets across all LLM call budgets, while remaining competitive on MATH500.

dently, capturing inter-candidate agreement structure; and we use a lightweight gradient-boosted ranker over interpretable features rather than a neural reward model, requiring no additional model beyond the base LLM. In this way, we bring the set-relative perspective of LTR directly into the self-consistency framework, showing that a small set of interpretable features suffices to substantially close the gap to oracle selection.

### 3 Ranking-Improved Self-Consistency

In this section, we present *Ranking-Improved Self-Consistency* (RISC), which casts self-consistency as a ranking problem, and then derive the features that power the ranker.

#### 3.1 Self-Consistency as a Ranking Problem.

**Task Definition.** Let  $x \in \mathcal{X}$  denote an input prompt and  $y \in \mathcal{Y}$  a candidate output. Given a base generative model  $p_\theta(y | x)$ , self-consistency constructs a candidate set by drawing  $N$  independent samples,

$$\mathcal{C}(x) = \{y_1, y_2, \dots, y_N\}, \quad y_i \sim p_\theta(\cdot | x). \quad (1)$$

The goal is then to select, from  $\mathcal{C}(x)$ , the candidate that is most likely to yield a correct or otherwise desirable final answer. We formalize this as a *query-dependent ranking problem*, where the prompt  $x$  is the query and the sampled candidates in  $\mathcal{C}(x)$  are the items to be ranked.

We seek a scoring function  $f_\phi$  that assigns each candidate  $y \in \mathcal{C}(x)$  a score based on the prompt  $x$ , the candidate itself, and the full candidate set  $\mathcal{C}(x)$ :

$$f_\phi(x, y | \mathcal{C}(x)) \in \mathbb{R}. \quad (2)$$

The final prediction is then given by

$$\hat{y} = \arg \max_{y \in \mathcal{C}(x)} f_\phi(x, y | \mathcal{C}(x)). \quad (3)$$

In the simplest self-consistency setting, this score is given by the empirical frequency of an answer under repeated sampling, i.e., majority voting (Wang et al., 2023). More generally,  $f$  may be heuristic or learnable, and may depend on the candidate answer itself, its reasoning trace, or its agreement with other samples.

**Learning the Ranker.** The above-mentioned setting is naturally framed as ranking. Previous studies found that a reward model may be used as a ranker by scoring candidates independently (Cobbe et al., 2021), but this is only a special case. More generally, the objective is to rank candidates within the sampled set  $\mathcal{C}(x)$  for a fixed prompt  $x$ . This distinction matters because multiple candidates may be valid, semantically equivalent, or lead to the same correct final answer. Thus,  $f(x, y | \mathcal{C}(x))$  is better understood as a relative selection score.

Under this formulation, the scoring function is trained to rank candidates within the sampled set, rather than to estimate an absolute reward. From this perspective, it is often more natural to optimize the ordering of the entire candidate set  $\mathcal{C}(x)$  using listwise supervision, rather than relying only on pairwise comparisons. Given relevance labels or a partial ordering over sampled outputs, a simple listwise objective is

$$\mathcal{L}_{\text{list}} = -\mathbb{E}_{(x, \mathcal{C}(x))} \left[ \log \frac{\exp(f_\phi(x, y^+))}{\sum_{y \in \mathcal{C}(x)} \exp(f_\phi(x, y))} \right], \quad (4)$$

where  $y^+$  denotes a correct candidate in the sampled set. More generally, standard listwise learning-to-rank objectives can be used to optimize the ordering of the entire candidate pool.

Overall, this view treats self-consistency as selection from a prompt-specific set of sampled gen-

erations. Its effectiveness depends both on the diversity and quality of the candidate set  $\mathcal{C}(x)$  and on how well the scoring function ranks candidates within that set.

### 3.2 Training Ranker

In our study, we train a LightGBM (Shi et al., 2026) LambdaRank (Burgess et al., 2006) model and use NDCG (Järvelin and Kekäläinen, 2017) to incorporate list-aware information, placing greater emphasis on errors that more strongly affect the final ordering of the candidate set.

Formally, let  $s_i = f_\phi(x, y_i)$  denote the score assigned to candidate  $y_i$  in the sampled set  $\mathcal{C}(x)$ , and let  $r_i$  be its relevance label. Training is based on pairwise comparisons  $(y_i, y_j)$  within the same set, but the contribution of each pair is weighted by  $|\Delta\text{NDCG}_{ij}|$ , the change in the quality of the overall ranked list that would result from swapping their positions. Training details are described in Appendix A.

### 3.3 Feature descriptions

For each question, all traces (CoTs) are aggregated at the answer level. In other words, we assess answer quality either directly from the answer itself or indirectly by measuring how cohesive — or conversely, how different—the CoTs associated with the same answer are.

**Answer length.** Character length of the final answer extracted after CoT.

#### Intuition

This feature captures whether an answer appears in a clean, compact form. Correct answers are often expressed canonically (for example, as a short number or concise entity name), while incorrect answers are more likely to contain extra formatting, explanatory residue, or malformed text.

#### Ratio to best

For question  $i$ , let,  $n_{i,a}$  be the number of traces producing answer  $a$ , and let  $N_i = \sum_{a'} n_{i,a'}$  be the total number of traces for that question.

Define the answer share:  $\text{ans\_share}_{i,a} = \frac{n_{i,a}}{N_i}$ .

Let the largest answer share for question  $i$  be

$$\text{top1\_share}_i = \max_{a'} \text{ans\_share}_{i,a'}$$

Then, ratio-to-best is

$$\text{ratio\_to\_best}_{i,a} = \frac{\text{ans\_share}_{i,a}}{\text{top1\_share}_i}$$

#### Intuition

This feature measures how competitive an answer is relative to the strongest answer for the same question. It helps the ranker understand whether a candidate is nearly tied with the dominant answer or far behind it.

#### Distance from answer to answer centroid

Let  $V_{i,a} \in \mathbb{R}^d$  denote the embedding of candidate answer  $a$  for question  $i$ .

We define the count-weighted centroid for question  $i$  as

$$C_i^{\text{ans}} = \frac{\sum_{a'} n_{i,a'} V_{i,a'}}{\sum_{a'} n_{i,a'}}$$

The feature is the squared Euclidean distance from the answer embedding to this weighted centroid:

$$\text{ans\_centroid\_dist}_{i,a} = \|V_{i,a} - C_i^{\text{ans}}\|_2^2$$

#### Intuition

This feature measures how far a candidate answer lies from the semantic center of all answers proposed for the same question. Weighting the centroid by answer frequency makes it a “center of mass” of the answer distribution rather than an unweighted average over unique strings. Answers that are semantically central to the candidate set may be more plausible, while scattered or atypical answers are often less reliable.

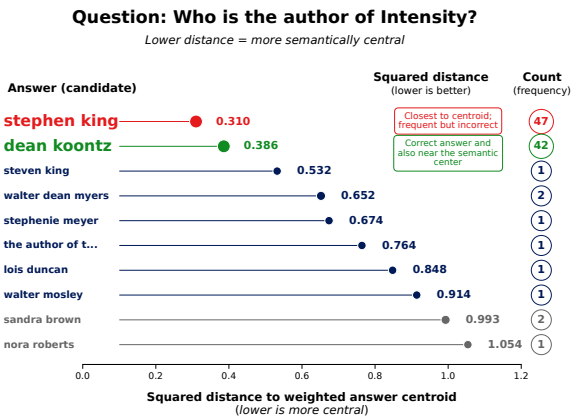


Figure 3: Example visualization of the answer-centroid distance feature. Each point shows the squared distance from a candidate answer embedding to the count-weighted answer centroid; lower values indicate greater semantic centrality.

#### Worst-step coherence

For a trace  $t$ , let its step embeddings be  $s_{t,1}, s_{t,2}, \dots, s_{t,m_t} \in \mathbb{R}^d$ . Define the within-trace step centroid as  $c_t = \frac{1}{m_t} \sum_{j=1}^{m_t} s_{t,j}$ .

For each step, compute its cosine similarity to the trace centroid:  $r_{t,j} = \cos(s_{t,j}, c_t)$ .

The trace-level worst-step coherence is  $r_t^* = \min_{1 \leq j \leq m_t} r_{t,j}$ .

For a fixed question  $i$  and answer  $a$ , let  $T_{i,a}$  be the set of traces producing answer  $a$ . The answer-level feature is the mean of the trace-level worst-step coherences:

$$\text{worst\_step\_coherence}_{i,a} = \frac{1}{|T_{i,a}|} \sum_{t \in T_{i,a}} r_t^*.$$

### Intuition

This feature captures whether every step in a reasoning trace stays semantically aligned with the overall trajectory. A low value suggests that at least one step is off-direction, which may indicate a derailment even if the final answer looks plausible.

**Question: The football manager who recruited David Beckham managed Manchester United during what timeframe?**

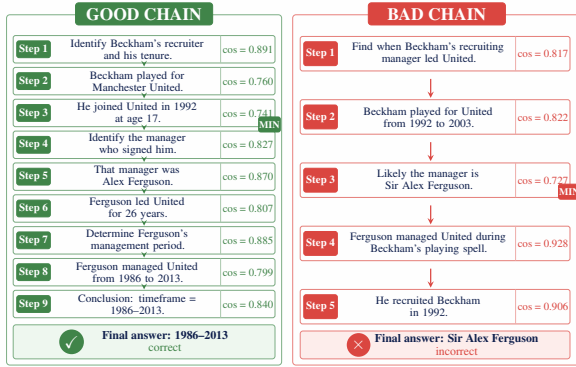


Figure 4: Example visualization of the worst-step coherence feature. Each row shows a reasoning step with its cosine similarity to the within-chain step centroid; lower values indicate weaker alignment with the chain’s overall reasoning trajectory. The minimum-coherence step is marked with MIN, and final answer boxes show whether the chain produced the correct answer. Step sentences are shortened with ellipses for readability. The full example is provided in Appendix F.

### Shared checkpoints count

For a fixed question  $i$  and answer  $a$ , consider the set of traces  $T_{i,a}$ . Each trace is converted into a sequence of prefixes. If trace  $t$  has steps  $x_{t,1}, x_{t,2}, \dots, x_{t,m_t}$ , then its prefixes are

$$\begin{aligned} p_{t,1} &= x_{t,1}, \\ p_{t,2} &= x_{t,1} \oplus x_{t,2}, \dots \\ p_{t,m_t} &= x_{t,1} \oplus x_{t,2} \oplus \dots \oplus x_{t,m_t}. \end{aligned}$$

Let  $e_{i,j}$  be the embedding of prefix  $p_{t,j}$ . Let the normalised depth of prefix  $j$  in trace  $t$  be  $d_{t,j} = \frac{j}{m_t}$ .

Two prefixes  $p_{t,j}$  and  $p_{t',k}$  between traces are considered to be matched if they share two conditions:

$$1. |d_{t,j} - d_{t',k}| \leq \tau_{\text{depth}}$$

$$2. \cos(e_{t,j}, e_{t',k}) \geq \tau_{\text{sim}}$$

A prefix  $p_{t,j}$  is called a shared checkpoint, denoted  $z_{t,j}$ , if it matches prefixes from at least a fraction  $q$  of the other traces with the same final answer:

$$\text{shared\_checkpoints}_{i,a} = \sum_{t \in T_{i,a}} \sum_{j=1}^{m_t} z_{t,j}.$$

### Intuition

This feature measures how much intermediate reasoning structure is shared among traces with the same answer. It asks not only whether traces reach the same final answer, but also whether they pass through similar semantic states at similar stages. Higher values indicate a more stable and reproducible reasoning pattern.

### Factoring $x^8 + 3x^4 - 4$

Two different solution traces with matched checkpoints.

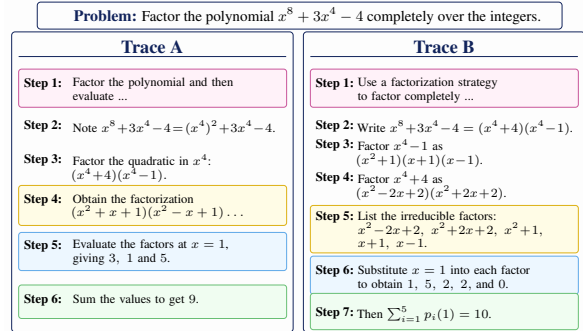


Figure 5: Example visualization of the shared-checkpoints feature. Matching colors indicate semantically shared checkpoints detected at similar reasoning depths across two reasoning traces. Step sentences are shortened with ellipses for readability. Full example is provided in Appendix G.

We encode answers, chain-of-thoughts, prefixes, and reasoning steps using the MiniLM<sup>2</sup> sentence embedding model (Wang et al., 2021) (just 22.7M parameters), which produces dense sentence-level embeddings via transformer encoding followed by pooling.

## 4 Experimental Setup

### 4.1 Datasets

In this work, we utilize a suite of datasets to cover complex tasks: long-tail QA, multi-hop reasoning and math reasoning. **PopQA** (Mallen et al., 2023) is an open-domain question answering benchmark

<sup>2</sup>[hf.co/sentence-transformers/all-MiniLM-L6-v2](https://hf.co/sentence-transformers/all-MiniLM-L6-v2)

consisting of entity-centric questions derived from Wikipedia. Each question is associated with a popularity score reflecting the monthly Wikipedia page view count of the corresponding entity. Following [Mallen et al. \(2023\)](#), we partition the dataset in train and test set. **HotpotQA** ([Yang et al., 2018](#)) requires multi-hop reasoning over multiple supporting documents. In our work, we use the closed-book scenario, i.e., without gold or distractor contexts. **MATH500** ([Lightman et al., 2024](#)) is a subset of 500 competition-level mathematics problems spanning algebra, geometry, number theory, and other domains.

## 4.2 Baselines

**Self-Consistency.** Self-Consistency ([Wang et al., 2023](#)) samples  $m$  reasoning chains and selects the answer by majority vote:

$$\hat{a}_{\text{SC}} = \arg \max_a \sum_{i=1}^m \mathbf{1}[a_i = a]. \quad (5)$$

It requires no additional training or auxiliary models, but achieving reliable performance demands a large number of samples.

**Confidence-Informed Self-Consistency.** CISC ([Taubenfeld et al., 2025](#)) replaces uniform votes with confidence-weighted aggregation. A per-response confidence score  $c_i$  is obtained via the  $\mathbb{P}(\text{True})$  method ([Kadavath et al., 2022](#)) and normalized via temperature-scaled softmax, yielding:

$$\hat{a}_{\text{CISC}} = \arg \max_a \sum_{i=1}^m \mathbf{1}[a_i = a] \cdot \tilde{c}_i. \quad (6)$$

CISC reduces the required number of samples by over 40% on average while matching SC accuracy.

**Stable Rank.** [Tang and Yang \(2025\)](#) propose using the stable rank of the response hidden-state matrix  $\mathbf{H} \in \mathbb{R}^{T \times d}$  as a reference-free quality signal:

$$\text{SR}(\mathbf{H}) = \frac{\|\mathbf{H}\|_F^2}{\|\mathbf{H}\|_2^2}, \quad (7)$$

observing that higher stable rank correlates with response quality in best-of- $N$  selection.

**ReASC.** ReASC ([Kim et al., 2026](#)) replaces count-based stopping with a reliability-aware evidence sufficiency criterion. It first attempts to answer from a single sample, accepting the response

only if its confidence exceeds a calibrated threshold. Otherwise, it continues sampling and updates a Beta posterior with confidence-weighted pseudo-counts, so higher-confidence responses contribute more evidence than lower-confidence ones. Sampling stops when the posterior indicates sufficient support for the leading answer. This lets ReASC avoid unnecessary samples and reduce inference cost.

## 4.3 Experimental Details

**Generation Setup.** In all our experiments we use Llama 3.1 8B Instruct<sup>3</sup> ([Team, 2024](#)). As an additional generalizability check, we evaluate our method on the Olmo-3-7B-Instruct model; the results are reported in Appendix 13<sup>4</sup> ([Olmo et al., 2025](#)). This model shows extensive knowledge coverage and benefits most from scaling on all datasets. For all datasets, samples were generated with a temperature of  $t=0.9$ ,  $\text{top}_p=1.0$ , we generated 99 samples sequentially. The generation for the train and test splits was done at 0-shot with Chain-of-Thought ([Wei et al., 2022](#)); prompts are available in Appendix D.

**Evaluation Setup.** We evaluate QA performance using Inclusion Accuracy (In-Acc) ([Chen et al., 2017](#)) as a relaxed exact match by checking whether any reference answer occurs as an adjacent token subsequence within the extracted final answer  $\hat{a}$ .

**Efficiency and Accuracy Metrics.** Following CISC ([Taubenfeld et al., 2025](#)), we additionally evaluate response selection methods using two complementary metrics. The *cost reduction* is defined as:

$$\text{Cost Reduction} = 100 \times \left(1 - \frac{b}{b_{\text{SC}}}\right) \%, \quad (8)$$

where  $b$  is a fixed sample budget for a method and  $b_{\text{SC}}$  is the number of calls that standard self-consistency requires to reach the same accuracy level. The second metric captures the accuracy gain when all methods are given the same number of responses. Given accuracy  $\text{Acc}_{\text{method}}$  and  $\text{Acc}_{\text{SC}}$  at equal sample budgets, the *accuracy improvement* is:

$$\text{Acc. Improvement} = 100 \times \left(\frac{\text{Acc}_{\text{method}}}{\text{Acc}_{\text{SC}}} - 1\right) \%. \quad (9)$$

<sup>3</sup>[hf.co/meta-llama/Llama-3.1-8B-Instruct](https://hf.co/meta-llama/Llama-3.1-8B-Instruct)

<sup>4</sup>[hf.co/allenai/Olmo-3-7B-Instruct](https://hf.co/allenai/Olmo-3-7B-Instruct)

Dataset	Cost Reduction			Accuracy Improvement			Headroom Over Maximum		
	@5	@10	@15	@5	@50	@99	@50	@75	@99
PopQA	61.54	71.43	70.00	6.08	4.75	6.77	4.31	5.34	6.42
HotpotQA	37.50	50.00	64.29	5.95	6.49	8.03	4.79	6.63	7.98
Math500	37.50	28.57	28.57	4.80	5.07	3.61	1.30	0.33	2.93
Average	45.51	50.00	54.29	5.61	5.44	6.14	3.47	4.10	5.78

Table 1: Performance summary on PopQA, HotpotQA, and Math500 under different LLM-call budgets. Here, @ $k$  denotes results at a budget of  $k$  LLM calls. We report cost reduction, accuracy improvement, and headroom over the maximum, where headroom is defined relative to SC’s maximum achieved accuracy.

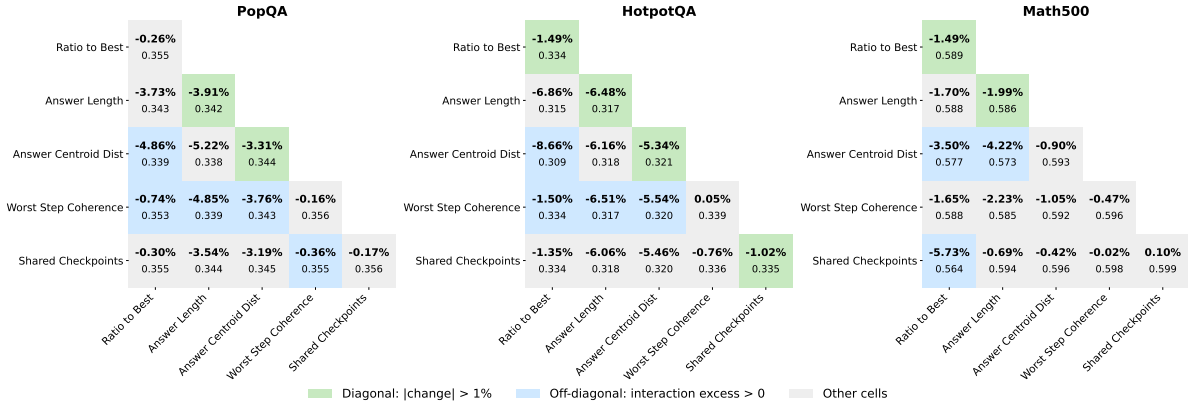


Figure 6: **Feature ablation across three datasets.** Heatmaps show the percentage change in mean accuracy, averaged over budgets 1–99 LLM calls, relative to the full ranker. Diagonal cells represent single-feature ablations, and lower-triangular cells represent two-feature ablations. Raw mean accuracies for the ablated models are shown in parentheses. Cells are colored by effect type: diagonal cells with an absolute drop greater than 1% are highlighted, while off-diagonal cells are highlighted when the joint ablation is larger than the sum of individual drops. All other cells are shown in light gray.

For the QA datasets, we observe that beyond a certain budget RISC attains an accuracy that self-consistency never matches, even at the maximum number of sampled calls. To quantify this gap, we introduce a third metric, the *headroom over maximum SC*, defined as:

$$100 \times \left( \frac{\text{Acc}_{\text{method}}(b)}{\max_{k \leq K} \text{Acc}_{\text{SC}}(k)} - 1 \right) \%. \quad (10)$$

Together, these metrics allow us to compare methods both in terms of how much computation they save and how much accuracy they add at fixed cost.

## 5 Results

Figure 2 shows a consistent pattern across all three datasets: RISC improves steadily with budget and stays above both CISC and vanilla self-consistency over most of the range, while Stable Rank is clearly weaker and sometimes degrades as more calls are added. On PopQA and HotpotQA, the gap appears early and widens with budget. On MATH500, the gap is smaller, but RISC remains competitive at low

budgets and finishes highest at the largest budgets. The advantage is therefore not limited to low budgets: across all three datasets, RISC remains strong or continues improving at high budgets, rather than saturating as early as CISC or self-consistency.

The quantitative results in Table 1 support this pattern: RISC consistently provides a better efficiency–accuracy trade-off than the baselines across all three datasets. Positive cost-reduction values show that it often matches self-consistency with substantially fewer calls, especially on PopQA and HotpotQA. The accuracy-improvement metric is positive in every reported setting, showing that the method is not only cheaper at matched quality but also more accurate at the same budget. Finally, positive headroom-over-maximum values show that in many settings RISC reaches accuracy levels that self-consistency never attains within the sampled range, particularly on the QA datasets.

We evaluate out-of-domain generalization by testing how rankers trained on the training split of one dataset perform on the test split of another dataset (Figure 7). Performance remains competi-

Train dataset	Test dataset		
	HotpotQA	PopQA	MATH500
HotpotQA	0.339	0.350	0.499
PopQA	0.324	0.356	0.524
MATH500	0.314	0.331	0.598

■ In-domain   
■ Within 10% of target   
■ >10% below target

Figure 7: **Out-of-domain transfer.** Mean ranker quality over 1–99 LLM calls for each train–test dataset pair. Diagonal cells show in-domain performance; off-diagonal cells show out-of-domain transfer. Cells are colored by relative transfer quality: green indicates performance within 10% of the target dataset’s in-domain score, while red indicates larger degradation. Transfer remains competitive on HotpotQA and PopQA, but drops substantially when transferring to MATH500.

tive on HotpotQA and PopQA, but drops substantially when transferred to MATH500.

Appendix B provides some intuition for why several strong baselines perform surprisingly poorly.

## 6 Analysis

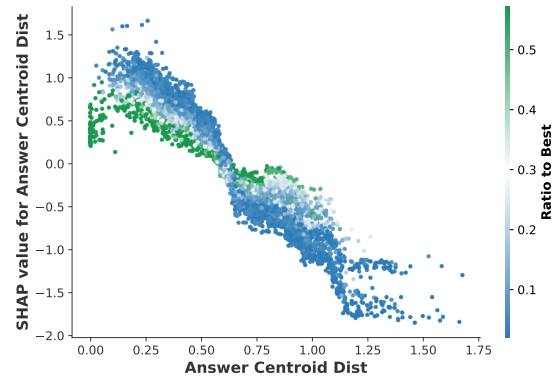
### 6.1 Feature Ablation

Figure 6 shows the results of single- and two-feature ablations. On the QA datasets, Answer Length and Answer Centroid Dist are the most individually impactful features, with HotpotQA showing sharper drops than PopQA. On MATH500, no single feature matters much in isolation — the model is robust to any individual removal.

The more interesting pattern emerges from two-feature ablations, which reveal complementarity: removing two features together causes significantly larger drops than the sum of their individual effects. On HotpotQA and PopQA, Ratio to Best and Answer Centroid Dist are the most complementary pair. MATH500 shows the most striking case: Ratio to Best and Shared Checkpoints are nearly irrelevant individually, yet removing both causes a -5.7% drop — a strong sign that the ranker relies on their combination rather than either signal alone.

### 6.2 SHAP feature analysis

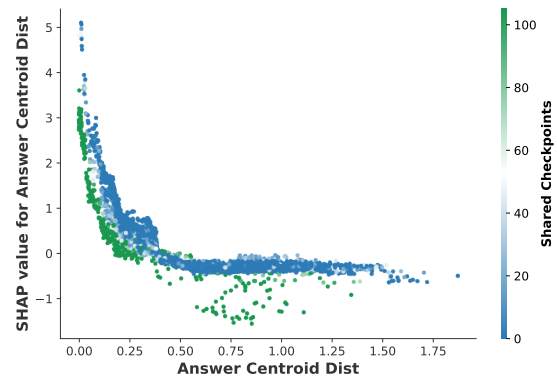
To further analyze feature interactions, Figure 8 shows SHAP dependence plots for the most influential features per dataset. In each plot, the x-axis is the raw feature value, the y-axis is its SHAP contribution to the ranker score, and the color encodes a second feature. The key signal is vertical color



(a) PopQA



(b) HotpotQA



(c) MATH500

Figure 8: SHAP dependence plots for selected features across PopQA, HotpotQA, and MATH500. The x-axis shows the feature value, the y-axis its contribution to the ranker score, and color encodes a second feature. Vertical color separation at the same x-value reveals that the two features interact — the contribution of one depends on the value of the other.

separation: if green and blue points sit at different heights for the same x-value, the two features interact — the effect of the first depends on the value of the second.

The dependence plots reveal that *features do not act independently* — their contributions are conditioned on the values of other features, producing

non-additive interactions that vary across datasets. In PopQA, a semantically central answer is rewarded more strongly when it is not the dominant one, while a semantically distant answer is not penalized if it is sufficiently common. In HotpotQA, high Ratio to Best is penalized when reasoning overlap is weak, suggesting it reflects coincidental rather than genuine agreement. In MATH500, semantically central answers reached via independent reasoning paths are favored over those produced by correlated chains.

## 7 Conclusion

In this study, we presented a new framing of self-consistency as a ranking problem, considering information from the ranking order. Adopting this view, we construct a lightweight ranker trained to rank candidate answers by their correctness. Further we test it on three diverse datasets and show that, RISC consistently improves the accuracy-cost trade-off over standard self-consistency and strong baselines. The gains are especially pronounced on the QA datasets, where the ranker frequently matches self-consistency with substantially fewer samples and, at higher budgets, can surpass the best achieved accuracy.

To analyze the ranker’s properties, we conduct a feature analysis showing that the ranker benefits from combining complementary signals rather than relying on any single handcrafted criterion. Overall, these results suggest that ranking is a promising and practical direction for improving test-time scaling in LLMs.

## Limitations

Our study has several limitations. First, although we evaluate the proposed approach using generations from two models, Llama-3.1-8B-Instruct and Olmo-3-7B-Instruct, both are relatively small instruction-tuned models. Therefore, it remains unclear how well the approach generalizes across broader model families and substantially different model scales.

Second, the gains on mathematical reasoning benchmarks are more limited than on question answering tasks. This suggests that the proposed features and ranking formulation may be better suited to settings such as QA, where semantic agreement and answer clustering are more informative, than to domains like math reasoning, where correctness can depend on finer-grained intermediate steps.

Third, although RISC consistently improves over standard self-consistency, there remains a substantial gap to oracle performance. In other words, the correct answer is often present among the sampled candidates, but the ranker still fails to identify it reliably in many cases. This highlights considerable room for improvement in test-time answer selection.

## Acknowledgements

The work was supported by the grant for research centers in the field of AI provided by the Ministry of Economic Development of the R.F. in accordance with the agreement 000000C313925P4F0002 and the agreement with Skoltech №139-10-2025-033.

## References

- Avinash Amballa, Aditya Parashar, Aditya Vikram Singh, Jinlin Lai, and Benjamin Rozenoyer. 2025. Quasi-random Multi-Sample Inference for Large Language Models. In *Frontiers in Probabilistic Inference: Learning meets Sampling*.
- Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. [Learning to Rank with Nonsmooth Cost Functions](#). In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 193–200. MIT Press.
- Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. [Reading Wikipedia to Answer Open-Domain Questions](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics.
- Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. [Universal Self-Consistency for Large Language Model Generation](#). *Preprint*, arXiv:2311.17311.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training Verifiers to Solve Math Word Problems](#). *CoRR*, abs/2110.14168.
- Xingyu Dang, Christina Baek, Kaiyue Wen, Zico Kolter, and Aditi Raghunathan. 2025. [Weight Ensembling Improves Reasoning in Language Models](#). *CoRR*, abs/2504.10478.

- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. [DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning](#). *Nat.*, 645(8081):633–638.
- Kalervo Järvelin and Jaana Kekäläinen. 2017. [IR evaluation methods for retrieving highly relevant documents](#). *SIGIR Forum*, 51(2):243–250.
- Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. [SWE-bench: Can Language Models Resolve Real-world Github Issues?](#) In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. [Language Models \(Mostly\) Know What They Know](#). *CoRR*, abs/2207.05221.
- Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. [When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs](#). *Trans. Assoc. Comput. Linguistics*, 12:1417–1440.
- Junseok Kim, Nakyeong Yang, Kyungmin Min, and Kyomin Jung. 2026. [Reliability-Aware Adaptive Self-Consistency for Efficient Sampling in LLM Reasoning](#). *CoRR*, abs/2601.02970.
- 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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Tie-Yan Liu. 2009. [Learning to Rank for Information Retrieval](#). *Found. Trends Inf. Retr.*, 3(3):225–331.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. [When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9802–9822. Association for Computational Linguistics.
- Maria Marina, Nikolay Ivanov, Sergey Pletenev, Mikhail Salnikov, Daria Galimzianova, Nikita Krayko, Vasily Konovalov, Alexander Panchenko, and Viktor Moskvoretskii. 2025. [Llm-independent adaptive RAG: let the question speak for itself](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 8697–8709. Association for Computational Linguistics.
- Marcus J. Min, Yangruibo Ding, Luca Buratti, Saurabh Pujar, Gail E. Kaiser, Suman Jana, and Baishakhi Ray. 2024. [Beyond Accuracy: Evaluating Self-Consistency of Code Large Language Models with IdentityChain](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Viktor Moskvoretskii, Chris Biemann, and Irina Nikishina. 2025. [Self-Taught Self-Correction for Small Language Models](#). *CoRR*, abs/2503.08681.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. 2025. [s1: Simple test-time scaling](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 20275–20321. Association for Computational Linguistics.
- Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, and 49 others. 2025. [Olmo 3](#). *Preprint*, arXiv:2512.13961.
- OpenAI. 2024. [OpenAI o1 System Card](#). *CoRR*, abs/2412.16720.
- Sergey Pletenev, Maria Marina, Nikolay Ivanov, Daria Galimzianova, Nikita Krayko, Mikhail Salnikov, Vasily Konovalov, Alexander Panchenko, and Viktor Moskvoretskii. 2025. [Will it still be true tomorrow? multilingual evergreen question classification to improve trustworthy QA](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 8603–8620. Association for Computational Linguistics.
- Yu Shi, Guolin Ke, Damien Soukhavong, James Lamb, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu, Nikita Titov, and David Cortes. 2026. [lightgbm: Light Gradient Boosting Machine](#). R package version 4.6.0.99.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. [Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters](#). *CoRR*, abs/2408.03314.
- Richard S. Sutton. 2019. [The Bitter Lesson](#). <http://www.incompleteideas.net/IncIdeas/BitterLesson.html>. Accessed: 2026-03-15.

Yixuan Tang and Yi Yang. 2025. [SR-GRPO: Stable Rank as an Intrinsic Geometric Reward for Large Language Model Alignment](#). *Preprint*, arXiv:2512.02807.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. 2025. [Confidence Improves Self-Consistency in LLMs](#). In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pages 20090–20111. Association for Computational Linguistics.

Llama Team. 2024. [The Llama 3 Herd of Models](#). *CoRR*, abs/2407.21783.

Weiqin Wang, Yile Wang, and Hui Huang. 2025. [Ranked Voting based Self-Consistency of Large Language Models](#). *Preprint*, arXiv:2505.10772.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. [MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pre-trained Transformers](#). In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL-IJCNLP 2021 of *Findings of ACL*, pages 2140–2151. Association for Computational Linguistics.

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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. [HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

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](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zhan Zhuang, Xiequn Wang, Zebin Chen, Feiyang Ye, Ying Wei, Kede Ma, and Yu Zhang. 2026. [One-Token Verification for Reasoning Correctness Estimation](#). *CoRR*, abs/2603.01025.

## A Training Details

To improve generalization across inference budgets, we replicate each question across multiple budget settings by retaining only the first  $n$  sampled answers and treating each budget-specific subset as a separate ranking instance, where  $n$  corresponds to the number of available LLM calls.

We exclude training questions that do not induce a meaningful ranking signal, where all candidate answers are either incorrect or correct. The data is then split into 80% training and 20% validation at the original question level, ensuring that budget-specific variants of the same question do not appear in both splits.

After hyperparameter selection on the validation set, we retrain the ranker on the full training data.

Default hyperparameters

```
DEFAULT_LGB_PARAMS = dict(
    objective="lambdarank",
    metric="ndcg",
    n_estimators=600,
    learning_rate=0.005,
    max_depth=5,
    num_leaves=20,
    subsample=0.9,
    colsample_bytree=1.0,
    min_data_in_leaf=20,
    feature_fraction_bynode=0.8,
    reg_lambda=1.0,
    reg_alpha=1e-3,
    random_state=42,
)
```

Hyperparameter grid

```
PARAM_GRID = {
    "n_estimators": [500, 600, 700],
    "learning_rate": [0.005, 0.01],
    "num_leaves": [24, 31],
    "max_depth": [5, 6],
    "min_data_in_leaf": [20, 40],
```

```

    "reg_lambda": [0.3, 1.0],
    "feature_fraction_bynode": [0.8, 1.0],
}

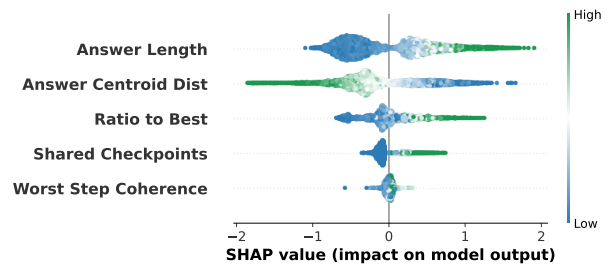
```

## B Baselines vs SC performance

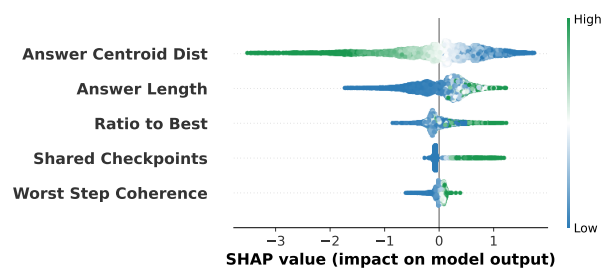
**Stable Rank vs. Majority Vote.** Here we provide an additional analysis of why SR performed poorly. Stable rank captures the geometric spread of a response’s hidden-state representation, but this property is only weakly associated with correctness. Wrong (but elaborate) responses can exhibit high representational diversity, while correct (but concise) answers may concentrate on few dimensions.

More fundamentally, stable rank operates as a per-response quality signal and cannot exploit the error-cancellation mechanism that gives majority voting its strength. Correct answers cluster while errors disperse, so as sample size grows, majority voting benefits from stronger consensus. However, SR-based selection becomes increasingly susceptible to outlier responses with anomalously high geometric spread. This explains the observed performance degradation at larger N, and this is why CISC works better. The confidence signal in CISC works because it is fed back into a weighted majority vote, rather than using it for single-response selection.

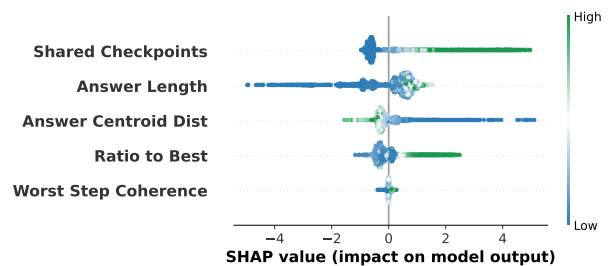
## C SHAP analysis



(a) PopQA



(b) HotpotQA



(c) MATH500

Figure 9: SHAP feature importance plots for three datasets, illustrating the average impact of selected features on the model output across the evaluated samples.

## D Prompts

Self-Consistency Prompt Template (PopQA)
<p><b>[System]</b> You are a helpful reasoning assistant for question answering. Answer the user’s question directly based on your knowledge.</p>
<p><b>[User]</b> Reason step by step very concisely before giving a final answer. Strictly follow the format: Step-by-step reasoning: [your reasoning] Final Answer: [your answer]</p> <p>Question: <i>{question}</i></p>

Figure 10: Prompt template used for self-consistency chain-of-thought sampling. The model is queried  $N$  times with temperature  $T > 0$ ; the final answer is then selected by majority vote over extracted answers.

MATH500 Prompt Template
<p><b>[System]</b> You are a helpful and expert mathematical assistant. Solve the given math problem step-by-step.</p>
<p><b>[User]</b> Reason step-by-step to solve the math problem. At the end of your solution, strictly follow this format to present your final answer: Final Answer: <code>\boxed{answer}</code> Ensure the answer within <code>\boxed{}</code> is concise and contains only the final value or expression.</p> <p>Problem: <i>{problem}</i></p>

Figure 11: Prompt template used for the MATH500 dataset. The system prompt establishes the role of a mathematical assistant, while the user instructions enforce a chain-of-thought structure with a specific `\boxed{}` format for the final answer extraction.

## E ReASC algorithm

Given an input  $x$ , ReASC (Kim et al., 2026) samples responses  $y_i \sim p_\theta(\cdot | x)$  with extracted answers  $a_i = \text{Ans}(y_i)$  and confidence scores  $s_i = S(y_i)$ . If the first sampled response satisfies  $s_1 \geq \tau_{\text{gate}}$ , the algorithm returns  $a_1$ . Otherwise, it accumulates weighted evidence

$$v(a) = \sum_{i: a_i=a} \max\left(1, \exp\left(\lambda \frac{s_i - \mu}{\sigma}\right)\right),$$

Self-Consistency Prompt Template (HotpotQA)
<p><b>[System]</b> You are a helpful assistant that answers questions by reasoning step by step.</p>
<p><b>[User]</b> Think through the question carefully, then provide your final answer as a short phrase on the last line in the format: Final Answer: [your answer]</p> <p>Question: <i>{question}</i></p>

Figure 12: Prompt template used for self-consistency chain-of-thought sampling. The model is queried  $N$  times with temperature  $T > 0$ ; the final answer is then selected by majority vote over extracted answers.

where  $(\mu, \sigma)$  are calibration statistics and  $\lambda$  controls sensitivity to confidence. After each sample, letting  $a^{(1)}$  and  $a^{(2)}$  denote the answers with the largest and second-largest evidence, ReASC defines

$$\alpha = v(a^{(1)}) + 1, \quad \beta = v(a^{(2)}) + 1,$$

and stops when

$$1 - I_{1/2}(\alpha, \beta) \geq C_{\text{thresh}}.$$

The final prediction is  $\hat{a} = \arg \max_a v(a)$ .

## F Example of Worst-Step Coherence feature

Here, we present a comparison between two traces for the same question–answer pair. However, in the actual calculations, these computations are performed over all traces for each question–answer pair, rather than just the two traces shown here.

**Question:** The football manager who recruited David Beckham managed Manchester United during what timeframe?

### Good Chain

0.890848

To answer this question, I need to identify the football manager who recruited David Beckham and then determine during which timeframe he managed Manchester United.

0.760034

First, I know that David Beckham is a retired English football player who played for several clubs, including Manchester United.

0.740996

He was recruited to Manchester United in 1992, when he was just 17 years old.

0.826989

Next, I need to identify the manager who signed David Beckham to Manchester United.

0.869573

After some research, I found that the manager who recruited David Beckham was Alex Ferguson.

0.807464

Alex Ferguson is a well-known former manager of Manchester United, and he played a significant role in shaping the club's history during his 26-year tenure.

0.885220

Finally, to determine the timeframe during which Alex Ferguson managed Manchester United, I need to consider that David Beckham arrived at the club in 1992.

0.798782

Alex Ferguson's tenure as manager of Manchester United began in 1986, and he remained in the position until his retirement in 2013.

0.839749

Therefore, considering all the information, I can conclude that Alex Ferguson managed Manchester United during the timeframe from 1986 to 2013.

The answer is: from 1986 to 2013.

### Bad Chain

0.816500

To determine the timeframe during which the football manager who recruited David Beckham managed Manchester United, let's consider the key information.

0.821974

David Beckham played for Manchester United from 1992 to 2003.

0.727240

Given this, it is likely that the manager we are looking for is Sir Alex Ferguson.

0.927873

Sir Alex Ferguson was the manager at Manchester United during the entire period in which David Beckham played for the team.

0.905895

Sir Alex Ferguson is widely known to have recruited David Beckham during his tenure, when Beckham moved from Manchester United's youth team to the first team in 1992.

The answer is: Sir Alex Ferguson.

## G Example of Shared Checkpoints feature

Here, we present a comparison between two traces for the same question–answer pair. However, in the actual calculations, these computations are performed over all traces for each question–answer pair, rather than just the two traces shown here.

Trace A vs Trace B for the math problem: color-coded shared checkpoints

**Problem.** Let

$$x^8 + 3x^4 - 4 = p_1(x)p_2(x) \cdots p_k(x),$$

where each non-constant polynomial  $p_i(x)$  is monic with integer coefficients and cannot be factored further over the integers. Compute

$$p_1(1) + p_2(1) + \cdots + p_k(1).$$

### Trace A

Step 1: The problem requires us to factor the given polynomial  $x^8 + 3x^4 - 4$  into irreducible polynomials and then find the sum of the values obtained by evaluating each of these polynomials at  $x = 1$ .<sup>4</sup>

Step 2: To start, we can try to factor the given polynomial using the sum of squares identity,  $a^2 - b^2 = (a - b)(a + b)$ . We notice that  $x^8 + 3x^4 - 4$  can be expressed as  $(x^4)^2 + 2x^4 + 1 - 2x^4 - 4$ , which simplifies to  $(x^4 + 1)^2 - (x^2)^2$ . Applying the sum of squares identity, we get  $(x^4 + 1 - x^2)(x^4 + 1 + x^2)$ .

Step 3: Further simplifying, we employ the difference of squares again:  $x^4 + 1 - x^2 = (x^2 - x + 1)(x^2 + x + 1)$ , and  $x^4 + 1 + x^2 = (x^2 + x + 1)(x^2 - x + 1)$ .

Step 4: We see that the factor  $(x^2 + x + 1)$  appears in both factors. Factoring this out first allows us to obtain the factorization:  $(x^4 + 3x^2 - 4) = (x^2 + x + 1)(x^2 - x + 1)(x^2 + 4)$ .<sup>1</sup>

Step 5: Now, we need to evaluate each of these factors at  $x = 1$ . When we substitute  $x = 1$  into each factor, we get  $1 + 1 + 1 = 3$  from the first polynomial,  $1 - 1 + 1 = 1$  from the second polynomial, and  $1 + 4 = 5$  from the last polynomial.<sup>2</sup>

Step 6: The sum of these values is  $3 + 1 + 5 = \boxed{9}$ .<sup>3</sup>

### Trace B

## Step 1: The given expression is  $x^8 + 3x^4 - 4$ . To factorize this expression into the product of non-constant polynomials  $p_i(x)$ , we can use the difference of squares factorization.

##<sup>4</sup> Step 2: We can rewrite  $x^8 + 3x^4 - 4$  as a difference of squares:  $(x^4 + 4)(x^4 - 1)$ , where the latter expression can be further factorized as a difference of squares.





## Step 3: Now, we apply the difference of squares identity, which gives us:  $(x^4 - 1) = (x^2 + 1)(x + 1)(x - 1)$ .

## Step 4: Combining the initial factorization with the further factorization, we get the complete factorization of  $x^8 + 3x^4 - 4 = (x^4 + 4)(x^2 + 1)(x + 1)(x - 1)$ .

## Step 5: Finally, since  $x^4 + 4$  cannot be factored further over the integers, we have  $k = 4$ , and the polynomials are  $p_1(x) = x^4 + 4$ ,  $p_2(x) = x^2 + 1$ ,  $p_3(x) = x + 1$ , and  $p_4(x) = x - 1$ .

##<sup>1</sup> Step 6: We can now evaluate each  $p_i(1)$  by substituting  $x = 1$  into each of the polynomials. This gives us  $p_1(1) = (1)^4 + 4 = 5$ ,  $p_2(1) = (1)^2 + 1 = 2$ ,  $p_3(1) = 1 + 1 = 2$ , and  $p_4(1) = 1 - 1 = 0$ .

##<sup>2</sup> Step 7: Therefore, we find that  $p_1(1) + p_2(1) + p_3(1) + p_4(1) = 5 + 2 + 2 + 0 = 9$ .<sup>3</sup>

Pair	Color	Sim.	Checkpoint in Trace A	Checkpoint in Trace B
1		0.823	Step 4: We see that the factor $(x^2 + x + 1)$ appears in both factors. Factoring this out first allows us to obtain the factorization: $(x^4 + 3x^2 - 4) = (x^2 + x + 1)(x^2 - x + 1)(x^2 + 4)$ .	Step 5: Finally, since $x^4 + 4$ cannot be factored further over the integers, we have $k = 4$ , and the polynomials are $p_1(x) = x^4 + 4$ , $p_2(x) = x^2 + 1$ , $p_3(x) = x + 1$ , and $p_4(x) = x - 1$ . ##
2		0.823	Step 5: Now, we need to evaluate each of these factors at $x = 1$ . When we substitute $x = 1$ into each factor, we get $1 + 1 + 1 = 3$ from the first polynomial, $1 - 1 + 1 = 1$ from the second polynomial, and $1 + 4 = 5$ from the last polynomial.	Step 6: We can now evaluate each $p_i(1)$ by substituting $x = 1$ into each of the polynomials. This gives us $p_1(1) = (1)^4 + 4 = 5$ , $p_2(1) = (1)^2 + 1 = 2$ , $p_3(1) = 1 + 1 = 2$ , and $p_4(1) = 1 - 1 = 0$ . ##
3		0.823	Step 6: The sum of these values is $3 + 1 + 5 = \boxed{9}$ .	Step 7: Therefore, we find that $p_1(1) + p_2(1) + p_3(1) + p_4(1) = 5 + 2 + 2 + 0 = 9$ .
4		0.800	Step 1: The problem requires us to factor the given polynomial $x^8 + 3x^4 - 4$ into irreducible polynomials and then find the sum of the values obtained by evaluating each of these polynomials at $x = 1$ .	Step 1: The given expression is $x^8 + 3x^4 - 4$ . To factorize this expression into the product of non-constant polynomials $p_i(x)$ , we can use the difference of squares factorization. ##

## H Inference on the Olmo-3-7B-Instruct model

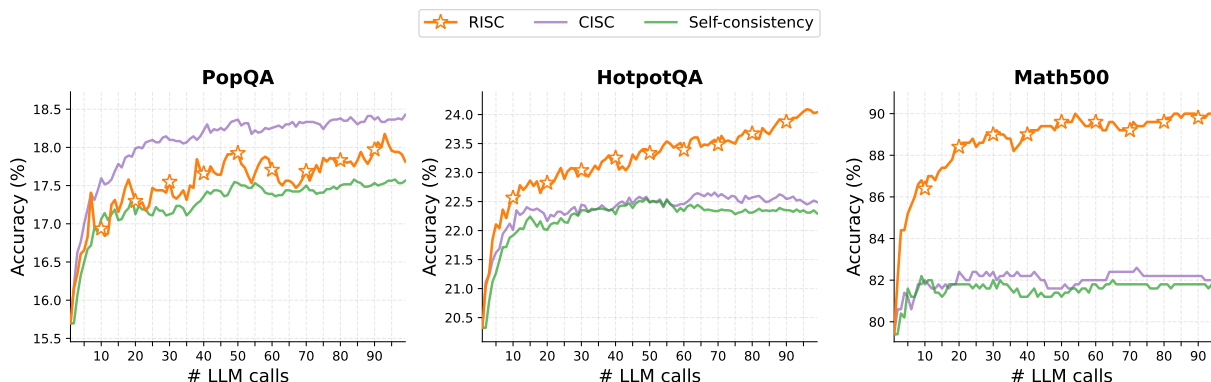


Figure 13: Comparison of RISC against Self-Consistency, Stable Rank, ReASC, and CISC on three datasets for Olmo-3-7B-Instruct model. RISC consistently outperforms the baselines on the HotpotQA and MATH500 datasets.

Compared to the results obtained with Llama-3.1-8B-Instruct, we observe a performance drop on PopQA, which is likely due to the disproportionately small number of high-quality training examples, i.e., questions for which both correct and incorrect candidate answers are available. For HotpotQA, we observe a similar trend. In contrast, on MATH500, the method shows a substantial performance improvement compared to Llama-3.1-8B-Instruct. The majority answer is often already correct, as reflected by the top-1 accuracy. However, among the remaining cases, errors frequently occur when the majority answer is wrong because generations stop at an intermediate reasoning step without producing the final answer. Since p-true-based CISC may still assign high confidence to such incomplete reasoning traces, RISC handles these cases more effectively. Since such cases are the main source of top-1 errors when the majority choice is wrong, correcting them yields a large apparent performance gain.