

Search-R2: Enhancing Search-Integrated Reasoning via Actor-Refiner Collaboration

Bowei He^{♣◊♡*}, Minda Hu^{♣♡*}, Zenan Xu^{♡*}, Hongru Wang^{§†}, Licheng Zong[♣], Yankai Chen^{♣◊}
 Chen Ma[♭], Xue Liu^{♣◊}, Pluto Zhou^{♡†}, Irwin King^{♣†}

[♣]The Chinese University of Hong Kong [♡]LLM Department, Tencent

[♣]Mohamed bin Zayed University of Artificial Intelligence

[◊]McGill University [♭]City University of Hong Kong

[§]The University of Edinburgh

Abstract

Search-integrated reasoning enables language agents to transcend static parametric knowledge by actively querying external sources. However, training these agents via reinforcement learning is hindered by the *multi-scale credit assignment* problem: existing methods typically rely on sparse, trajectory-level rewards that fail to distinguish between high-quality reasoning and fortuitous guesses, leading to redundant or misleading search behaviors. To address this, we propose Search-R2, a novel Actor–Refiner collaboration framework that enhances reasoning through targeted intervention, with both components jointly optimized during training. Our approach decomposes the generation process into an Actor, which produces initial reasoning trajectories, and a Meta-Refiner, which selectively diagnoses and repairs flawed steps via a “cut-and-regenerate” mechanism. To provide fine-grained supervision, we introduce a hybrid reward design that couples outcome correctness with a dense process reward quantifying the information density of retrieved evidence. Theoretically, we formalize the Actor–Refiner interaction as a smoothed mixture policy, proving that selective correction yields strict performance gains over strong baselines. Extensive experiments across various general and multi-hop QA datasets demonstrate that Search-R2 consistently outperforms strong RAG and RL-based baselines across model scales, achieving superior reasoning accuracy with minimal overhead.

1 Introduction

Large language models are rapidly evolving from static knowledge repositories into dynamic, search-integrated agents that interact with external environments (Trivedi et al., 2023; Li et al., 2025). By combining iterative reasoning with active retrieval, these agents tackle knowledge-intensive tasks such as open-domain and multi-hop question answering that were previously intractable due to limited parametric knowledge and hallucinations. Consequently, this field has turned to Reinforcement Learning (RL) to optimize these systems (Jin et al., 2025; Chen et al., 2025), grounding agent behavior in task-specific performance objectives rather than imitation of human demonstrations.

However, training search-integrated agents with RL faces a key challenge: *multi-scale credit assignment*. In practice, agent behavior is a sequence of decisions, including query formulation, information filtering, and logical deduction, yet standard methods optimize policies with trajectory-level rewards such as final-answer correctness (Jin et al., 2025; Wang et al., 2025). Since this outcome-only signal provides no supervision over intermediate reasoning or the timing and necessity of retrieval, it induces credit misattribution across both retrieval and reasoning decisions (Zhang et al., 2025a). Consequently, efficient, logically coherent trajectories receive similar credit to trajectories that succeed only after redundant, costly, or poorly timed retrieval, reducing sample efficiency and yielding brittle reasoning chains. This limitation highlights a critical gap in current methodologies: *the inability to diagnose and repair error propagation*. As shown in Figure 1, a single irrelevant search query early in a trajectory can misguide the entire subsequent reasoning chain. Existing rejection sampling techniques (Ahn et al., 2024) are inefficient here, as they discard the entire trajectory rather than addressing the specific root cause of the deviation. To build robust agents, we must move beyond outcome-based filtering toward a paradigm that enforces both *global reasoning coherence* and *local search quality*.

To this end, we propose Search-R2, a novel Actor–Refiner collaboration framework designed to enhance search-integrated reasoning through targeted intervention. Unlike standard generation approaches, Search-R2 decomposes the reasoning process into two distinct roles: an Actor that generates initial reasoning trajectories with tool calls, and a Meta-Refiner that identifies localized failures, such as uninformative retrieval or logical gaps, and performs a “cut-and-regenerate” operation. This mechanism preserves valid reasoning prefixes while surgically repairing flawed

*The first three authors have equal contributions.

[†]Correspondence to: Hongru Wang <hrwang@ed.ac.uk>, Irwin King <king@cse.cuhk.edu.hk>, Pluto Zhou <plutozhou096@foxmail.com>

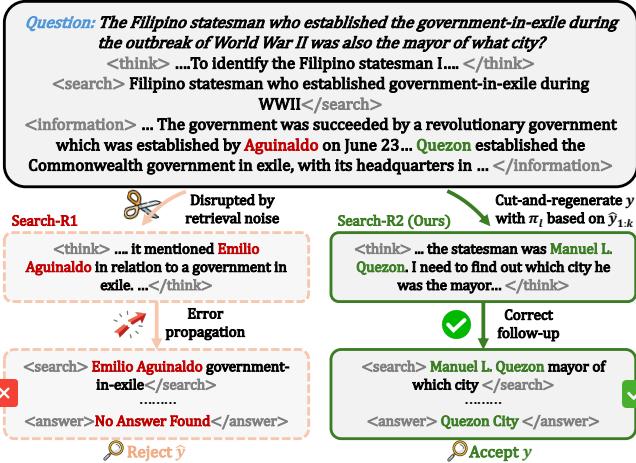


Figure 1: **Demonstration of Search-R1 and Search-R2.** While Search-R1 (Left) is disrupted by retrieval noise and falls into an error propagation loop, Search-R2 (Right) utilizes an Actor-Refiner collaboration. The Meta-Refiner identifies the deviation and applies a "cut-and-regenerate" mechanism to surgically repair the reasoning chain at the point of error, successfully redirecting focus from the incorrect entity (Aguinaldo) to the correct one (Quezon).

suffixes, significantly enhancing learning efficiency. The Actor and Meta-Refiner are jointly optimized during training, enabling mutual feedback between trajectory generation and selective refinement. To further provide dense supervision, we introduce a hybrid reward that combines outcome correctness with a process reward that quantifies the density of evidence information. We theoretically prove that our Actor–Refiner interaction, which is modeled as a smoothed mixture policy, strictly exceeds the performance of baselines like rejection sampling under satisfiable conditions. Experiments on seven benchmarks show consistent gains of the proposed Search-R2 over strong RAG and RL-based baselines across model sizes ranging from 7B to 32B, with minimal overhead.

In summary, our contributions are as follows:

1. *Problem Identification:* We formalize the multi-scale credit assignment problem in search-integrated reasoning, highlighting the inadequacy of trajectory-level rewards for optimizing intermediate search behaviors.
2. *Framework:* We propose Search-R2, an Actor–Refiner framework that integrates step-level process rewards with a trajectory-level “cut-and-regenerate” refinement mechanism, and jointly optimizes both the Actor and the Refiner.
3. *Theoretical Analysis:* We characterize the Meta-Refiner as a mixture policy and derive the theoretical conditions under which selective correction guarantees performance improvement over baseline sampling.
4. *Empirical Success:* We demonstrate state-of-the-art performance on seven across different-size models, showing that Search-R2 improves both the accuracy of final answers and the quality of the underlying search process.

2 Related Works

This section reviews prior work on search-integrated reasoning and multi-turn reinforcement learning.

2.1 Search-Integrated Reasoning

Search-integrated language agents augment large language models with the ability to actively query external information sources during problem solving, enabling them to overcome the limitations of static parametric knowledge (Jin et al., 2025; Chen et al., 2025). Prior work has explored search-augmented reasoning for tasks such as multi-hop question answering (Sun et al., 2025; Wu et al., 2025), deep research (Team et al., 2025; Hu et al., 2024), and web-based decision making (Zhou et al.; Hu et al., 2025), demonstrating that iterative search can substantially improve factual accuracy and coverage. More recent approaches integrate search into reinforcement learning frameworks (Chen et al., 2025; Qian & Liu, 2025; Song et al., 2025), allowing agents to learn when and how to issue search queries based on task-level feedback. However, existing methods typically optimize search behavior only through delayed, trajectory-level rewards, without explicitly assessing the quality of individual search decisions (Wen et al., 2026). As a result, agents often issue redundant, mistimed, or weakly informative queries, especially in long-horizon interactions (Gao et al., 2025), where suboptimal search decisions compound over time and degrade both task performance and learning efficiency.

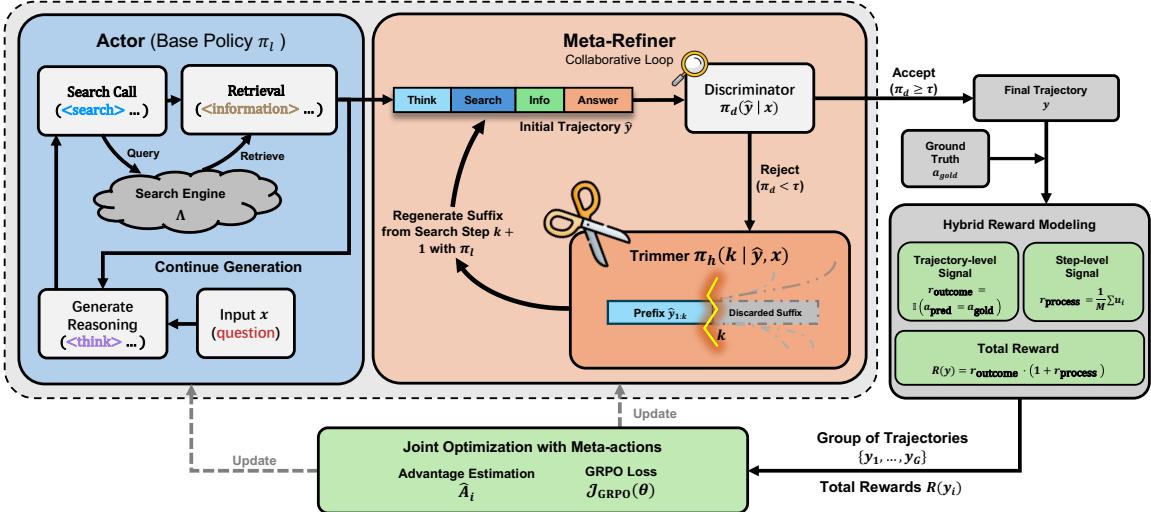


Figure 2: Overview of the Search-R2 framework. The Actor generates initial reasoning trajectories with search queries. The Meta-Refiner employs a Discriminator to detect errors and a Trimmer to identify the exact step of failure. Upon rejection, the trajectory is truncated and regenerated from the error point. The system is jointly optimized via GRPO using a hybrid reward.

2.2 Credit Assignment in Multi-Turn RL

Learning effective policies for multi-turn decision making remains a central challenge in reinforcement learning and agent research due to sparse rewards and difficult credit assignment (Devidze et al., 2022; Wang & Ammanabrolu, 2025). This challenge is particularly pronounced in search-integrated agents, where intermediate decisions such as query formulation and timing are evaluated only through final task outcomes (Zhang et al., 2025a). Prior work has proposed dense reward shaping (Zeng et al., 2025; Zhang et al., 2025b) and learned reward models (Zou et al., 2025), including Large Language Models (LLM)-based judges (Zha et al., 2025), to provide richer feedback signals. While these techniques improve optimization stability in some settings, they are most commonly applied to evaluate final responses or aggregate trajectory quality, leaving the quality of intermediate decisions underspecified. Consequently, policy optimization often suffers from low sampling efficiency, as many rollouts contain low-quality intermediate actions that contribute little to learning. These limitations motivate approaches that provide fine-grained supervision over intermediate decisions while remaining compatible with multi-turn optimization.

3 Methodology

We propose Search-R2, a novel Actor-Refiner collaboration framework designed to address the multi-scale credit assignment challenge. Rather than treating search-integrated reasoning as a monolithic generation task, our approach decouples the process into two distinct phases: an *Actor* generating initial reasoning chains, and a *Meta-Refiner* performing trajectory-level assessment and causal correction. This decomposition allows us to optimize both global reasoning coherence and local search quality simultaneously.

3.1 The Search-Integrated Reasoning Actor

The foundation of our system is an Actor policy, denoted as $\pi_I(\cdot | x)$, responsible for generating the initial reasoning trajectory \hat{y} . Given the search engine Λ , the π_I is trained to invoke Λ autonomously following a standard tool-use paradigm (Algorithm 2), to enable dynamic information acquisition. The model generates a chain of thought and, when necessary, emits a query within $<\text{search}> \dots </\text{search}>$ tags. The system halts generation, executes the query against Λ , appends the top- k results within $<\text{information}> \dots </\text{information}>$ tags, and resumes generation. This cycle repeats until the model outputs the final answer or reaches a step limit.

To initialize π_I , we utilize a structural template (Table 1) that enforces a strict format: Reasoning \rightarrow Search Call \rightarrow Answer. This acts as a soft constraint, ensuring adherence to the system's operational logic without imposing content-specific biases.

3.2 The Meta-Refiner for Hierarchical Correction

A core premise of our work is that suboptimal search decisions often occur in intermediate steps and silently misguide subsequent reasoning. Standard rejection sampling is inefficient for repairing such cascading errors.

Table 1: Template for Search-Integrated Reasoning following the implementation of Search-R1 Jin et al. (2025).

Answer the given question. You must conduct reasoning inside `<think>` and `</think>` first... if you lack knowledge, call search engine via `<search>` query `</search>`... return results in `<information>`... Final answer in `<answer>`... Question: `question`.

To address this, we introduce the Meta-Refiner, which performs *targeted causal intervention* rather than blind regeneration. The Meta-Refiner shares the underlying LLM with the Actor but is steered by control prompts to perform two sub-objectives.

1) *Discriminator for global coherence checking.* The Discriminator, denoted $\pi_d(\hat{y} | x) \in [0, 1]$, serves as a gate that enforces trajectory-level reasoning coherence. Given a reasoning trajectory \hat{y} , it estimates the probability that the reasoning remains globally coherent with the problem specified by x . We accept \hat{y} when $\pi_d(\hat{y} | x) \geq \tau$; otherwise, we flag it for refinement. Accordingly, the acceptance probability is a Bernoulli distribution $\alpha(\hat{y}|x) = P(\pi_d(\hat{y} | x) \geq \tau)$.

2) *Trimmer for local error localization.* To address the issue of error propagation, the Trimmer $\pi_h(k|\hat{y}, x)$ identifies the specific search step $k + 1$ where the reasoning or search query first deviated (the "root cause"). The system preserves the valid prefix $\hat{y}_{1:k}$, truncates the flawed suffix, and regenerates a new suffix using the base policy π_l . This "cut-and-regenerate" strategy preserves valuable partial reasoning, significantly improving sample efficiency compared to discarding the entire trajectory.

Together, the discriminator and trimmer implement an iterative accept-or-repair procedure. For each candidate trajectory, the discriminator first decides whether it is globally coherent. If it is rejected, the trimmer localizes the earliest deviation and triggers cut-and-regenerate editing to produce a revised trajectory. This collaborative process induces a smoothed mixture policy $q(y | x)$, formalized in Algorithm 1. Repeating this procedure up to a budget N_{\max} yields progressively improved trajectories and accumulates correction history, which strengthens the Meta-Refiner's ability to localize errors over time.

Algorithm 1 Meta-Refiner Execution Flow

```

1: Input: Context  $x$ , Policy  $\pi_l$ , Discriminator  $\pi_d$ , Trimmer  $\pi_h$ .
2: Generate initial trajectory  $\hat{y} \sim \pi_l(\cdot|x)$ 
3: while  $n < N_{\max}$  do
4:   if  $\pi_d(\hat{y}|x) \geq \tau$  then
5:     return  $\hat{y}$  {Accept}
6:   end if
7:   Sample cut-point  $k \sim \pi_h(\cdot|\hat{y}, x)$ 
8:    $y_{\text{prefix}} \leftarrow \hat{y}_{1:k}$ 
9:   Regenerate  $y_{\text{suffix}} \sim \pi_l(\cdot|x, y_{\text{prefix}})$ 
10:   $\hat{y} \leftarrow [y_{\text{prefix}}, y_{\text{suffix}}]$ 
11:   $n \leftarrow n + 1$ 
12: end while
13: return  $y = \hat{y}$ 

```

3.3 Hybrid Reward Modeling for Multi-Scale Supervision

To tackle the credit assignment issue where local search actions are conflated with global outcomes, we introduce a hybrid reward $R(y)$ that provides supervision at both scales.

Global Outcome Reward. We use Exact Match (EM) between the predicted answer a_{pred} and ground truth a_{gold} : $r_{\text{outcome}}(y) = \mathbb{I}(a_{\text{pred}} = a_{\text{gold}})$. This ensures the final output satisfies the user's intent.

Local Process Reward. To distinguish between trajectories that are correct by chance versus those supported by high-quality evidence, we quantify the utility of retrieved context. For a set of retrieved chunks $C = \{c_1, \dots, c_M\}$, an external judge evaluates the utility $u_i \in \{0, 1\}$ of each chunk. The process reward is the density of useful information: $r_{\text{process}}(y) = \frac{1}{M} \sum_{i=1}^M u_i$. Implementation specifics are outlined in Appendix K.

Overall reward. To prevent reward hacking (maximizing retrieval without solving the task), the process reward is gated by outcome

$$R(y) = r_{\text{outcome}}(y) \cdot (1 + r_{\text{process}}(y)). \quad (1)$$

This formulation explicitly reinforces the principle that high-quality search is a necessary condition for robust reasoning.

3.4 Joint Optimizing the Actor and Meta-Refiner

We leverage Group Relative Policy Optimization (GRPO) to optimize the shared weight θ of Actor and Meta-Refiner jointly (Shao et al., 2024). For each input x , we sample a group of G trajectories $\{y_1, \dots, y_G\}$ from the mixture distribution $q(\cdot|x)$. Crucially, we treat each y_i as an augmented execution trace comprising both the reasoning path from π_l and refinement actions sampled from the discriminator $\pi_d(y)$ and trimmer $\pi_h(k|y)$. The objective is to maximize:

$$\begin{aligned}\mathcal{L}_{\text{GRPO}}(\theta) &= \mathbb{E}_{x, \{y_i\}_{i=1}^G \sim q} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{L_i} \sum_{t=1}^{L_i} L_t(y_i, \theta) \right] \\ L_t(y_i, \theta) &= \left[r_t(\theta) \hat{A}_{i,t}, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{\text{KL}}[\pi_l || \pi_{\text{ref}}],\end{aligned}\quad (2)$$

where the advantage \hat{A}_i is computed via group normalization of the hybrid rewards, and $r_t(\theta)$ denotes the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, which measures the deviation of the current policy from the old policy. This allows the model to learn the optimal balance between generation and correction solely from the interaction outcome, effectively solving the multi-scale credit assignment problem end-to-end.

3.5 Mechanisms of Performance Gain

Unlike prior work that optimizes only the Actor, we jointly optimize both the Actor and the Meta-Refiner. To rigorously justify the necessity of optimizing the Meta-Refiner, as opposed to relying on static prompting or standard rejection sampling, we decompose the total expected sample reward improvement ΔJ , into three governing mechanisms. As formally derived in Appendix D, the net performance gain is not a byproduct of mere sampling volume, but strictly depends on the agent’s ability to satisfy specific covariance conditions. We characterize the gain decomposition as:

$$\Delta J = \underbrace{\mathcal{A}_{\text{prec}}}_{\text{Selection Precision}} + \underbrace{\mathcal{V}_{\text{inter}}}_{\text{Intervention Volume}} \times \underbrace{\mathcal{S}_{\text{trim}}}_{\text{Trimming Skill}}. \quad (3)$$

We next describe each term in Eq. 3 and explain how it contributes to the overall improvement:

Selection Precision $\mathcal{A}_{\text{prec}}$. This term represents the system’s capacity for global evaluation. Mathematically defined as $\text{Cov}_{\pi_l}(\alpha(y), R(y) - J_{\text{trim}}(y))$, it measures the alignment between the discriminator’s acceptance probability and the trajectory’s relative quality. A positive $\mathcal{A}_{\text{prec}}$ implies the discriminator successfully distinguishes which trajectories are worth preserving while exposing chains requiring correction. (e.g., those containing hallucinations or redundant steps) to the refinement process. By treating the entire interaction trace, such as reasoning, decision-to-accept, and decision-to-cut, as a single unified trajectory, GRPO naturally maximizes this covariance without requiring separate supervision signals for the Meta-Refiner.

Trimming Skill $\mathcal{S}_{\text{trim}}$. This term quantifies the effectiveness of the “cut-and-regenerate” mechanism. Defined as $\sum_k \text{Cov}(\pi_h(k|y), G_k(y))$, it measures the correlation between the selected cut-point k and the expected gain $G_k(y)$ from regenerating at that specific step. Therefore, a positive $\mathcal{S}_{\text{trim}}$ indicates that the Trimmer precisely locates the specific low-quality search action that caused the reasoning collapse, such as a failed search query or a logic error, where the trajectory first deviated. This behavior is reinforced by propagating the outcome reward back to the specific cut-point selection k , encouraging the agent to target pivotal moments of failure.

Intervention Volume $\mathcal{V}_{\text{inter}}$. Defined as $1 - \mathbb{E}[\alpha(y)]$, this term represents the volume of trajectories subjected to correction. It acts as a multiplier in the Eq. 3. Even a highly skilled trimmer ($\mathcal{S}_{\text{trim}} > 0$) contributes little if the discriminator is overly conservative (accepting flawed answers, $\mathcal{V}_{\text{inter}} \rightarrow 0$). Conversely, if the discriminator flags valid answers ($\mathcal{V}_{\text{inter}} \rightarrow 1$) while the trimmer is unskilled, the computational budget is wasted. The system must find a balance between exploration and exploitation, ensuring that it neither overlooks errors nor wastes resources.

The joint optimization seeks an equilibrium where $\mathcal{V}_{\text{inter}}$ is sufficiently large to correct errors but constrained enough to preserve sample efficiency. Under joint optimization with meta-refiner, if the agent accepts a low-quality trajectory, the resulting low group-relative advantage penalizes the discriminator, directly driving $\mathcal{A}_{\text{prec}}$ upward.

Summary of Success Conditions. Unlike standard RAG or Rejection Sampling, which rely solely on the actor policy’s generation probability, Search-R2 achieves a net positive gain ($\Delta J > 0$) for each rollout if and only if three conditions are met simultaneously. Formally, these correspond to $\mathcal{A}_{\text{prec}} > 0$, $\mathcal{S}_{\text{trim}} > 0$, and a calibrated $\mathcal{V}_{\text{inter}}$ that exposes sufficient samples for refinement without suppressing high-quality outputs. Furthermore, the Meta-Refiner supports iterative execution within N_{max} , where the posterior $q(\cdot|x)$ from iteration t serves as the base policy for $t+1$. The conditions for improvement remain valid in recursive settings.

4 Formalization

In this section, we present a theoretical framework for analyzing the mechanisms that drive the performance improvements of Search-R2. While the previous section detailed the algorithmic implementation of the Actor-Refiner collaboration, this section aims to mathematically quantify the specific contributions of the discrimination and refinement phases. We formalize the collaborative process as a smoothed mixture policy and derive a decomposition of the expected reward gain. A summary of the mathematical notation is provided in Appendix 6.

4.1 Performance Analysis

Our primary theoretical objective is to quantify the performance advantage of the Meta-Refiner over the base actor policy. We analyze the expected performance gain, $\Delta J = J_{meta} - J_{base}$, where $J_{base} = \mathbb{E}_{y \sim \pi_l}[R(y)]$ represents the standard actor's performance, and $J_{meta} = \mathbb{E}_{y \sim q}[R(y)]$ represents the performance under the Meta-Refiner distribution q . Analyzing this difference is crucial because it allows us to mathematically disentangle two sources of improvement, namely the *discriminative ability* to identify poor samples and the *trimming ability* to correct them.

Proposition 4.1 (Performance Decomposition of Meta-Refiner). *Let the induced trajectory distribution $q(y | x)$ of the Meta-Refiner be formalized as a mixture policy:*

$$q(y | x) = \pi_l(y | x)\alpha(y) + \int_{\hat{y}} \pi_l(\hat{y} | x)(1 - \alpha(\hat{y}))T'(\hat{y} | x, \hat{y}) d\hat{y}, \quad (4)$$

where π_l is the base policy, $\alpha(y) \in [0, 1]$ is the acceptance probability, and $T'(\hat{y} | x, \hat{y})$ is the normalized transition distribution of the trimmer for cutting and regenerating a rejected sample \hat{y} . Note that q is self-normalized (see Proof in Appendix B). The expected reward J_{meta} decomposes relative to the base performance J_{base} as:

$$J_{meta} = J_{base} + \underbrace{\text{Cov}_{\pi_l}(a(y), R(y) - J_{trim}(y))}_{\text{Selection Precision}} + \underbrace{(1 - Z_{acc})(\bar{J}_{trim} - J_{base})}_{\text{Correction Volume Gain}}. \quad (5)$$

Here, $\text{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$ denotes the covariance. $J_{trim}(\hat{y}) = \mathbb{E}_{y \sim T'(\cdot | \hat{y})}[R(y)]$ is the expected reward after correcting \hat{y} , $\bar{J}_{trim} = \mathbb{E}_{\pi_l}[J_{trim}(\hat{y})]$, and $Z_{acc} = \mathbb{E}_{\pi_l}[a(y)]$ is the global acceptance rate.

This derivation characterizes q as a smoothed mixture policy. The performance gain is driven by the discriminator's precision in identifying low-quality samples (*Selection Precision*) and the trimmer's ability to improve those samples (*Correction Volume Gain*).

4.2 Decomposing the Correction Volume Gain

We further analyze the term $\Delta J_{trim} = \bar{J}_{trim} - J_{base}$, which represents the performance improvement provided by the trimming strategy. We aim to decompose this gain into the *baseline gain* and the *attribution ability* of the trimmer.

Preliminaries. Let \hat{y} be a draft sequence of length T from π_l rejected by the discriminator. We define the set of possible cut-points as $\mathcal{K} = \{1, \dots, T\}$. Let $\pi_h(k|\hat{y})$ be the trimmer policy (probability of cutting at index $k+1$) and let $V^{\pi_l}(\hat{y}_{1:k})$ be the value of regenerating the suffix from k .

Proposition 4.2 (Decomposition of Trimming Strategy). *Let $G_k(\hat{y}) = V^{\pi_l}(\hat{y}_{1:k}) - R(\hat{y})$ denote the **regeneration gain** at step k . The total correction gain ΔJ_{trim} decomposes into a covariance term representing the agent's skill and a mean term:*

$$\Delta J_{trim} = \underbrace{\sum_{k=1}^T \text{Cov}_{\hat{y}}(\pi_h(k|\hat{y}), G_k(\hat{y}))}_{\text{Trimming Skill}} + \underbrace{\bar{G}(\hat{y})}_{\text{Baseline Gain}}, \quad (6)$$

where $\bar{G}(\hat{y}) = \sum_k \mathbb{E}[\pi_h(k|\hat{y})]\mathbb{E}[G_k(\hat{y})]$ denotes the baseline gain (see proof in Appendix C).

This formulation isolates two drivers of performance:

- **Trimming Skill:** A positive covariance indicates that π_h concentrates probability mass on cut-points $k+1$ where the regeneration gain G_k is highest. This measures the agent's ability to identify the "root cause" of a bad generation. A positive covariance implies that the trimmer possesses the capacity for concentrating probability mass on the critical turning points k that yield the greatest regeneration gain (G_k) rather than performing random trimming.
- **Baseline Gain:** In high-dimensional reasoning tasks, arbitrarily truncating and regenerating a trajectory rarely improves the outcome (i.e., $\mathbb{E}[G_k(\hat{y})] \approx 0$ for random k). Consequently, $\bar{G} \approx 0$, implying that maximizing the correction gain ΔJ_{trim} relies almost entirely on the trimmer's skill in selecting precise cut-points.

Table 2: The main results on seven datasets. [†]/^{*} represents in-domain/out-of-domain datasets. All baselines except Search-R1 are conducted on the Qwen2.5-7B model. The best and second best performances are set as **bold** and underlined, respectively.

Methods	General QA				Multi-Hop QA			
	NQ [†]	TriviaQA*	PopQA*	HotpotQA [†]	2WikiMultiHopQA*	Musique*	Bamboogle*	Average
Direct Inference	13.4	40.8	14.0	18.3	25.0	3.1	12.0	18.1
CoT	4.8	18.5	5.4	9.2	11.1	2.2	23.2	10.6
IRCoT	22.4	47.8	30.1	13.3	14.9	7.2	22.4	23.9
Search-o1	15.1	44.3	13.1	18.7	17.6	5.8	29.6	20.6
RAG	34.9	58.5	39.2	29.9	23.5	5.8	20.8	30.4
SFT	31.8	35.4	12.1	21.7	25.9	6.6	11.2	20.7
R1-base	29.7	53.9	20.2	24.2	27.3	8.3	29.6	27.6
R1-instruct	27.0	53.7	19.9	23.7	29.2	7.2	29.3	27.1
Rejection Sampling	36.0	59.2	38.0	33.1	29.6	12.3	35.5	34.8
Search-R1(Qwen2.5-7B)	39.5	56.0	38.8	32.6	29.7	12.5	36.0	35.0
Search-R1(Qwen3-8B)	44.0	63.1	41.8	37.2	35.5	15.7	43.0	40.0
Search-R1(Qwen2.5-32B)	47.6	68.0	47.0	43.3	46.2	22.1	45.0	45.6
Search-R2(Qwen2.5-7B)	39.9	65.9	41.0	39.0	35.8	15.1	46.2	40.4
Search-R2(Qwen3-8B)	47.7	67.6	46.6	41.2	40.5	17.2	51.2	44.6
Search-R2(Qwen2.5-32B)	50.9	70.9	50.1	49.9	51.7	25.4	56.4	50.8

Table 3: Ablation results for Search-R2 on general and multi-hop question answering.

Method	General QA	Multi-Hop QA	Average
Qwen2.5-7B			
Search-R1	41.7	26.1	35.0
Search-R1 + Meta-Refiner	45.3	30.4	38.9
Search-R1 + Meta-Refiner + Process Reward	45.6	31.6	39.6
Search-R2 (Full Version)	46.5	32.4	40.4
Qwen3-8B			
Search-R1	46.5	31.4	40.0
Search-R1 + Meta-Refiner	49.4	35.3	43.4
Search-R1 + Meta-Refiner + Process Reward	49.9	36.1	44.0
Search-R2 (Full Version)	50.8	36.3	44.6
Qwen2.5-32B-Instruct			
Search-R1	51.5	37.8	45.6
Search-R1 + Meta-Refiner	54.2	42.7	49.3
Search-R1 + Meta-Refiner + Process Reward	54.3	43.3	49.5
Search-R2 (Full Version)	55.5	44.5	50.8

5 Experiments

5.1 Experiment Setup

Datasets: We evaluate search-integrated reasoning methods on two categories of datasets. For general question answering, we use NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2022). For multi-hop question answering, we use HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). We train on the union of the NQ and HotpotQA training splits. Evaluation is performed on the validation or test splits of all seven datasets, which allows us to measure in-domain performance on the training distributions as well as out-of-domain generalization to held-out datasets.

Methods: We compare Search-R2 against three baseline families and a strong reference model. (i) *Inference without retrieval*: direct inference and Chain-of-Thought (CoT) reasoning (Wei et al., 2022). (ii) *Inference with retrieval*: Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), IRCoT (Trivedi et al., 2023), and Search-o1 (Li et al., 2025). (iii) *Fine-tuning based methods*: supervised fine-tuning (SFT) (Chung et al., 2024), RL-based fine-tuning without search (R1) (Guo et al., 2025), and rejection sampling with a search engine (Ahn et al., 2024). (iv) *Reference*: Search-R1 (Jin et al., 2025), the backbone of our approach. We run experiments on three model backbones spanning multiple generations and scales, namely Qwen2.5-32B, Qwen2.5-7B, and Qwen3-8B (Yang et al., 2024; 2025).

Retriever: We use E5 (Wang et al., 2022) as the retriever and the 2018 Wikipedia dump (Karpukhin et al., 2020) as the knowledge source. For fairness, we directly utilize the available index file provided by (Jin et al., 2025) and set the number of retrieved passages to 3.

Table 4: The hyperparameter sensitivity experiment results with increasing maximum revision times (from 1 to 4) for each initial rollout trajectory. We conduct these experiments on the Qwen2.5-32B-Instruct model. The best performance is set as **bold**.

Max Revision	NQ	TriviaQA	PopQA	HotpotQA	2WikiMultiHopQA	Musique	Bamboogle	Average
1	50.8	69.4	49.0	47.6	49.4	24.2	54.4	49.3
2	50.9	71.0	50.6	48.7	50.7	25.5	54.4	50.2
3	51.4	71.2	50.4	49.3	51.4	25.7	54.4	50.6
4	51.6	71.2	50.8	49.3	51.6	26.0	55.6	50.9

Implementation Details: To ensure consistency with prior work (Jin et al., 2025), we use Exact Match (EM) as the evaluation metric and train all models with GRPO (Shao et al., 2024) for 300 steps. At each step, 512 prompts are randomly sampled, and $n = 5$ rollouts are generated for each prompt. Our training framework is based on the verl framework (Sheng et al., 2025) and sets the max assistant turns as 4. The max revision number per rollout is set as 1 by default. We use a learning rate of 1e-6 with a warmup ratio of 0.285. We provide more details in Appendix E.

5.2 Performance Comparison

Table 2 details the performance of Search-R2 against strong baselines on seven benchmarks. We observe that Search-R2 establishes a consistent performance lead. Notably, Search-R2 built on the Qwen2.5-7B backbone achieves a 16.1% EM gain over the Search-R1 rejection-sampling baseline, even when Search-R1 employs the stronger Qwen3-8B backbone. This confirms that the Actor-Refiner framework effectively compensates for reduced model scale by optimizing reasoning quality. When scaling the backbone from 7B to 32B, we observe a further performance gain, with average EM rising from 40.4 to 50.8. This consistent gain under model size scaling further highlights the effectiveness of our approach.

Moreover, the performance gains are more pronounced on complex reasoning tasks. For instance, Search-R2 achieves a 5.5-point improvement on 2WikiMultiHopQA and an 11.4-point improvement on Bamboogle (+25.3% relative gain). These tasks typically require multi-step retrieval and reasoning, where early mistakes and noisy intermediate search results can cascade and derail the remaining trajectory. By using the Meta-Refiner to detect deviations and sample high-quality traces, Search-R2 mitigates such error propagation and yields larger gains across different benchmarks. Finally, to further verify that these gains stem from targeted refinement rather than additional computation, we compare Search-R2 against the Search-R1 baseline trained using a doubled rollout budget ($n = 10$). As reported in Appendix G, Search-R2 ($n = 5$, max revision = 1) still performs better, indicating that surgical correction is substantially more sample-efficient than brute-force sampling.

5.3 Ablation Study

To rigorously disentangle the sources of improvement in Search-R2, we perform a component-wise analysis by sequentially integrating the Meta-Refiner, Process Reward, and Joint Optimization modules into the Search-R1 baseline. For the intermediate configurations (Search-R1 + Meta-Refiner and + Process Reward), we optimize the policy solely on reasoning traces, excluding intervention refinement from the Meta-Refiner. As can be seen in Table 3, each module contributes positively to overall performance. Firstly, the integration of the Meta-Refiner drives the largest performance leap (+11.1% on Qwen2.5-7B), suggesting that the Meta-Refiner acts as a crucial scaffold for reasoning coherence. Secondly, integrating the process reward yields consistent performance gains by explicitly valuing high-information-density retrieval. It guides the Actor under sparse feedback in complex reasoning settings. Finally, the full Search-R2 setup with joint optimization achieves the highest accuracy. These results support our strategy: unlike static methods, it enables the Actor and Meta-Refiner to co-adapt, allowing the policy to precisely localize errors and internalize the cut-and-regenerate mechanism for higher sample efficiency. Limited by the space, further details can be found in Appendix (Table 8).

5.4 Sensitivity to the Maximum Revision Limit

We evaluate sensitivity to the maximum revision limit by varying the max revision value from 1 to 4 using Qwen2.5-32B as the backbone. In these experiments, we disable process reward modeling and joint optimization to focus on the effect of allowing additional revisions. As shown in Table 4, increasing max revision yields consistent gains. Notably, max revision = 4 reaches an average score of 50.9, essentially matching the fully optimized Search-R2 with a single revision (50.8). This comparison highlights an efficiency trade-off that our proposed joint optimization strategy can successfully distill the benefits of a larger revision budget into a more efficient policy that achieves comparable accuracy with one correction step.

We also observe rapidly diminishing gains as the revision limit increases. The absolute EM gain drops from 0.9 points when increasing revisions from 1 to 2 to 0.3 points from 3 to 4. This pattern suggests that early revisions primarily correct errors that are relatively easy to fix, such as retrieval noise or shallow hallucinations, whereas

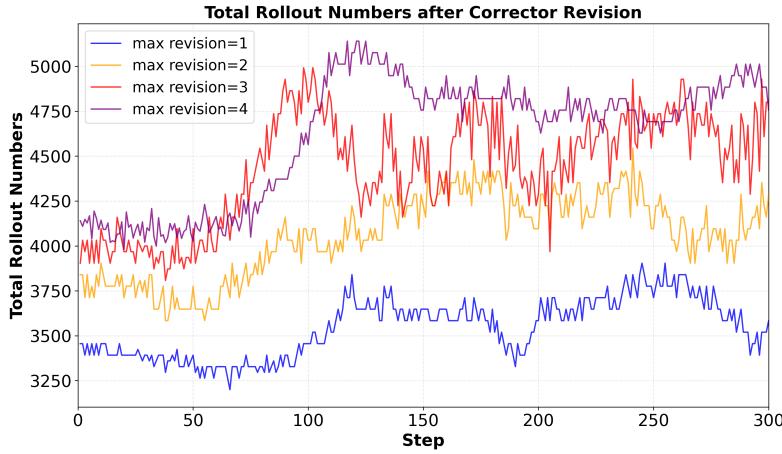


Figure 3: The total rollout numbers after revision (initial rollout numbers + refined rollout numbers) corresponding to different max revision settings.

the remaining failures are less responsive to repeated refinement. Figure 3 corroborates this trend, showing that most trajectories trigger at most one revision. Given higher max revision limits, harder cases rarely activate further refinement. Consequently, we set max revision = 1 as the default operating point, which captures most of the benefit at low revision cost.

Table 5: Average time cost for each training step (seconds/step).

Model	Search-R1	Search-R2	Relative Change	$\Delta \text{EM} (\%)$	$\Delta \text{Time} (\%)$
Qwen2.5-7B	177.8	193.2	+ 8.66%	1.78	
Qwen3-8B	141.5	147.3	+ 4.10%	2.80	
Qwen2.5-32B	458.4	469.5	+ 2.43%	4.69	

5.5 Efficiency Analysis

We now examine whether the Search-R2 pipeline introduces substantial computational overhead in practice. Surprisingly, Table 5 shows that Search-R2 increases training time by only 5.06% on average relative to the Search-R1 baseline. This modest overhead is largely due to the cut-and-regenerate mechanism, which preserves valid prefixes rather than discarding entire trajectories, thereby reducing wasted computation. Moreover, the relative overhead decreases with model scale and drops to 2.43% for the 32B model, suggesting that the marginal refinement cost becomes less significant as distributed training overhead grows. At inference time, Search-R2 introduces no additional latency because the Meta-Refiner is decoupled at deployment.

To quantify training cost-effectiveness, we report the ratio $\Delta \text{EM}(\%) / \Delta \text{Time}(\%)$, which measures accuracy improvement per unit increase in training time. As shown in Table 2, this ratio exceeds 1 for all models, indicating that accuracy gains consistently outpace the added compute. Moreover, the ratio further improves with scale, increasing from 1.78 at 7B to 4.69 at 32B, which suggests that Search-R2 becomes more cost-effective for larger backbones.

5.6 Trajectory Quality Comparison

To better understand trajectory quality, we compare Search-R2 against Search-R1 using GPT-5.1 as an automated judge. The evaluation covers six dimensions: *evidence groundedness*, *information density*, *non-redundancy efficiency*, *query timing quality*, *trajectory coherence*, and *uncertainty handling*. For each of the seven test datasets, we randomly sample 100 paired trajectories, evaluating Search-R1 and Search-R2 on the same prompt, for a total of 700 pairs. The judge assigns each trajectory an independent three-level score, with 0 indicating poor quality, 1 acceptable, and 2 strong. We then compare the paired scores and record a win when Search-R2 scores higher, a fail when it scores lower, and a tie otherwise¹. As shown in Figure 4, Search-R2 outperforms Search-R1 across all dimensions, indicating more grounded, efficient, and coherent search and reasoning behavior. Detailed rubrics, full results, and evaluation prompts are provided in Appendix I.

¹We omit ties in Figure 4 to improve readability.

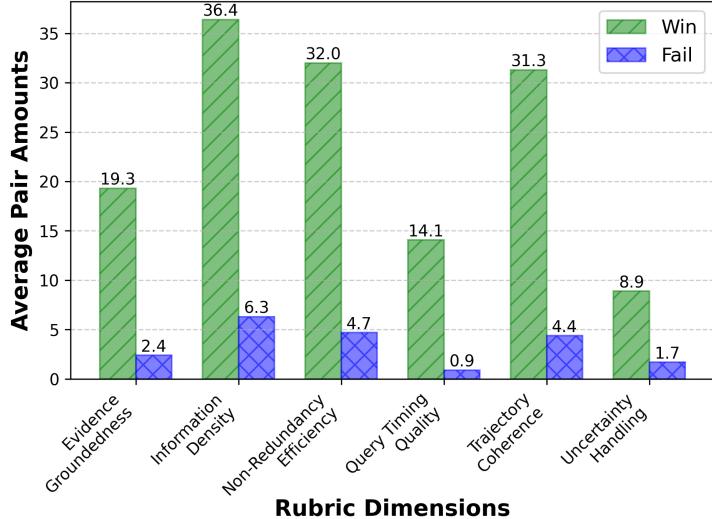


Figure 4: Average counts of Search-R2 winning and failing against Search-R1 across all seven datasets for each rubric.

6 Conclusions

In this work, we introduced Search-R2, a search-integrated reasoning framework designed to mitigate the LLM fragility when facing retrieval noise. Experiments show that while standard approaches like Search-R1 are susceptible to error propagation loops caused by misleading initial context, Search-R2’s Actor-Refiner collaboration with joint optimization effectively interrupts these failures. By employing a dynamic “cut-and-regenerate” mechanism, Search-R2 enables models to correct reasoning trajectories in real-time. These findings highlight the critical importance of integrating active refinement into search-integrated reasoning, offering a path toward more reliable agent behavior.

References

- Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 225–237, 2024.
- Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.
- Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. Exploration-guided reward shaping for reinforcement learning under sparse rewards. *Advances in Neural Information Processing Systems*, 35:5829–5842, 2022.
- Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*, 2025.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.
- Minda Hu, Licheng Zong, Hongru Wang, Jingyan Zhou, Jingjing Li, Yichen Gao, Kam-Fai Wong, Yu Li, and Irwin King. SeRTS: Self-rewarding tree search for biomedical retrieval-augmented generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1321–1335, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.71. URL <https://aclanthology.org/2024.findings-emnlp.71/>.

Minda Hu, Tianqing Fang, Jianshu Zhang, Jun-Yu Ma, Zhisong Zhang, Jingyan Zhou, Hongming Zhang, Haitao Mi, Dong Yu, and Irwin King. WebCoT: Enhancing web agent reasoning by reconstructing chain-of-thought in reflection, branching, and rollback. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5155–5173, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.276. URL <https://aclanthology.org/2025.findings-emnlp.276/>.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*, 2025.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 7, 2022.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Hongjin Qian and Zheng Liu. Scent of knowledge: Optimizing search-enhanced reasoning with information foraging. *arXiv preprint arXiv:2505.09316*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, et al. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *arXiv preprint arXiv:2505.16834*, 2025.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Hufeng Yin, Jialong Wu, Jingren Zhou, et al. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Acting less is reasoning more! teaching model to act efficiently, 2025. URL <https://arxiv.org/abs/2504.14870>.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Dixin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Ruiyi Wang and Prithviraj Ammanabrolu. A practitioner’s guide to multi-turn agentic reinforcement learning. *arXiv preprint arXiv:2510.01132*, 2025.

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

Tongyu Wen, Guanting Dong, and Zhicheng Dou. Smartsearch: Process reward-guided query refinement for search agents. *arXiv preprint arXiv:2601.04888*, 2026.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search. *arXiv preprint arXiv:2506.20670*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*, 2025.

Kaiwen Zha, Zhengqi Gao, Maohao Shen, Zhang-Wei Hong, Duane S Boning, and Dina Katabi. Rl tango: Reinforcing generator and verifier together for language reasoning. *arXiv preprint arXiv:2505.15034*, 2025.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, et al. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *arXiv preprint arXiv:2505.14069*, 2025a.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. Rlvmr: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. *arXiv preprint arXiv:2507.22844*, 2025b.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. Reasonflux-prm: Trajectory-aware prms for long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2506.18896*, 2025.

A Notation Table

Table 6 summarizes the mathematical notations and symbols used throughout the formalization and analysis of the Search-R2 framework.

Table 6: Summary of Notations and Symbols

Symbol	Description
<i>Policies and Models</i>	
$\pi_l(y x)$	The Base Policy (Actor) responsible for generating reasoning trajectories and search queries.
$\pi_d(\hat{y} x)$	The Discriminator (part of Meta-Refiner) that estimates the probability of a trajectory's global coherence.
$\pi_h(k \hat{y}, x)$	The Trimmer (part of Meta-Refiner) that identifies the specific step k (cut-point) where an error occurred.
$q(y x)$	The smoothed mixture policy induced by the interaction between the Actor and the Meta-Refiner realized by Algorithm 1.
Λ	The external search engine used for retrieval.
<i>Trajectory and Search</i>	
x	The input context or question.
\hat{y}	The initial reasoning trajectory generated by the Actor π_l .
y	The final trajectory output after potential refinement from $q(y x)$.
a_{pred}	The predicted final answer extracted from the trajectory.
a_{gold}	The ground truth answer.
k	The index of a step in the trajectory (specifically used as the cut-point).
$\hat{y}_{1:k}$	The valid prefix of trajectory \hat{y} up to step $k + 1$.
C	A set of retrieved chunks $\{c_1, \dots, c_M\}$.
<i>Rewards and Optimization</i>	
$R(y)$	The total hybrid reward described in Section 3.3, combining outcome and process signals.
$r_{\text{outcome}}(y)$	The outcome reward (Binary Exact Match) indicating if $a_{\text{pred}} = a_{\text{gold}}$.
$r_{\text{process}}(y)$	The process reward quantifying the information density of retrieved evidence.
$\mathcal{L}_{\text{GRPO}}(\theta)$	The objective function for Group Relative Policy Optimization.
\hat{A}_t	The advantage estimate computed via group normalization.
<i>Theoretical Analysis</i>	
$\text{Cov}(X, Y)$	The covariance between variables X and Y , defined as $\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$.
$\alpha(\hat{y} x)$	The Discriminator π_d 's acceptance probability of a trajectory, defined as $P(\pi_d(\hat{y} x) \geq \tau)$.
τ	The predefined threshold for the Discriminator π_d to accept a trajectory.
Z_{acc}	The global acceptance rate, defined as $\mathbb{E}_{\pi_l}[\alpha(y)]$.
J_{base}	The expected performance of the base policy: $\mathbb{E}_{y \sim \pi_l}[R(y)]$.
J_{meta}	The expected performance of the meta-refiner policy: $\mathbb{E}_{y \sim q}[R(y)]$.
ΔJ	The net performance gain: $J_{\text{meta}} - J_{\text{base}}$.
$\mathcal{A}_{\text{prec}}$	<i>Selection Precision:</i> Covariance measuring the Discriminator's ability to identify low-quality samples.
S_{trim}	<i>Trimming Skill:</i> Covariance measuring the Trimmer's ability to locate the root cause of errors.
V_{inter}	<i>Intervention Volume:</i> The probability mass allocated to the trimming process ($1 - Z_{\text{acc}}$).
$J_{\text{trim}}(\hat{y})$	The expected reward after correcting a specific rejected trajectory \hat{y} .
$G_k(\hat{y})$	The regeneration gain at step k : $V^{\pi_l}(\hat{y}_{1:k}) - R(\hat{y})$.
$V^{\pi_l}(\hat{y}_{1:k})$	The value of regenerating the suffix starting from step k using the base policy π_l .

B Proof for Performance Decomposition of Meta-Refiner

Proof. **1. Normalization Check.** We first verify that $q(y|x)$ integrates to 1.

$$\begin{aligned}
\int q(y|x) dy &= \int \pi_l(y) \alpha(y) dy + \int \left[\int \pi_l(\hat{y}) (1 - \alpha(\hat{y})) T'(y | \hat{y}) d\hat{y} \right] dy \\
&= \mathbb{E}_{\pi_l}[\alpha(y)] + \int \pi_l(\hat{y}) (1 - \alpha(\hat{y})) \underbrace{\left[\int T'(y | \hat{y}) dy \right]}_{=1} d\hat{y} \\
&= Z_{acc} + \mathbb{E}_{\pi_l}[1 - \alpha(\hat{y})] \\
&= Z_{acc} + (1 - Z_{acc}) = 1.
\end{aligned} \tag{7}$$

2. Expected Reward Derivation. The expected reward J_{meta} is the integral of $R(y)$ over the mixture components:

$$J_{meta} = \underbrace{\int R(y) \pi_l(y) \alpha(y) dy}_{\text{Term A (Accepted)}} + \underbrace{\int R(y) \left[\int \pi_l(\hat{y}) \bar{\alpha}(\hat{y}) T'(y | \hat{y}) d\hat{y} \right] dy}_{\text{Term B (Rejected)}}, \tag{8}$$

where $\bar{\alpha}(\hat{y}) = 1 - \alpha(\hat{y})$.

Analyzing Term A: Using the covariance identity $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y] + \text{Cov}(X, Y)$:

$$A = \mathbb{E}_{y \sim \pi_l}[R(y)\alpha(y)] = J_{base}Z_{acc} + \text{Cov}_{\pi_l}(\alpha, R).$$

Analyzing Term B: By Fubini's Theorem, we swap the order of integration:

$$\begin{aligned}
B &= \int \pi_l(\hat{y}) (1 - \alpha(\hat{y})) \left[\int R(y) T'(y | \hat{y}) dy \right] d\hat{y} \\
&= \int \pi_l(\hat{y}) (1 - \alpha(\hat{y})) J_{trim}(\hat{y}) d\hat{y} \\
&= \mathbb{E}_{\hat{y} \sim \pi_l} [(1 - \alpha(\hat{y})) J_{trim}(\hat{y})].
\end{aligned} \tag{9}$$

Let $\bar{J}_{trim} = \mathbb{E}_{\pi_l}[J_{trim}(\hat{y})]$. Applying the covariance identity again:

$$B = (1 - Z_{acc}) \bar{J}_{trim} - \text{Cov}_{\pi_l}(\alpha, J_{trim}). \tag{10}$$

Synthesis: Combining A and B, and grouping the covariance terms:

$$\begin{aligned}
J_{meta} &= [J_{base}Z_{acc} + (1 - Z_{acc}) \bar{J}_{trim}] + [\text{Cov}_{\pi_l}(\alpha, R) - \text{Cov}_{\pi_l}(\alpha, J_{trim})] \\
&= [J_{base}Z_{acc} + (1 - Z_{acc}) \bar{J}_{trim}] + \text{Cov}_{\pi_l}(\alpha, R - J_{trim}).
\end{aligned}$$

Subtracting J_{base} from both sides yields the final gain:

$$\Delta J = \text{Cov}_{\pi_l}(\alpha, R - J_{trim}) + (1 - Z_{acc})(\bar{J}_{trim} - J_{base}).$$

□

C Proof for Decomposition of Trimming Strategy

Proof. The total expected gain is the difference between the expected return after trimming and the baseline:

$$\begin{aligned}
\Delta J_{trim} &= \mathbb{E}_{\hat{y}} \left[\sum_{k=1}^T \pi_h(k|\hat{y}) V^{\pi_l}(\hat{y}_{1:k}) \right] - \mathbb{E}_{\hat{y}}[R(\hat{y})] \\
&= \mathbb{E}_{\hat{y}} \left[\sum_{k=1}^T \pi_h(k|\hat{y}) (V^{\pi_l}(\hat{y}_{1:k}) - R(\hat{y})) \right] \\
&= \sum_{k=1}^T \mathbb{E}_{\hat{y}} [\pi_h(k|\hat{y}) G_k(\hat{y})].
\end{aligned} \tag{11}$$

Applying the covariance identity $\mathbb{E}[XY] = \text{Cov}(X, Y) + \mathbb{E}[X]\mathbb{E}[Y]$ to each term in the summation yields the proposition. □

D Drivers of Performance Gain

Building upon Propositions 4.1 and 4.2, we decompose the total system improvement, ΔJ , into three governing factors. These components isolate the specific contributions of the discriminator’s judgment, the Meta-Refiner’s localization capability, and the overall frequency of intervention.

Definition D.1 (Selection Precision). Let $\mathcal{A}_{\text{prec}}$ quantify the covariance between the acceptance probability $\alpha(y)$ and the sample’s relative advantage (current reward minus potential correction value):

$$\mathcal{A}_{\text{prec}} \triangleq \text{Cov}_{\pi_l} (\alpha(y), R(y) - J_{\text{trim}}(y)). \quad (12)$$

A positive $\mathcal{A}_{\text{prec}}$ indicates that the discriminator functions as an effective filter, preferentially preserving samples where the existing reward $R(y)$ outweighs the expected value of a correction $J_{\text{trim}}(y)$.

Definition D.2 (Trimming Skill). Let $\mathcal{S}_{\text{trim}}$ quantify the alignment between the cut-point policy π_h and the regeneration gain $G_k(\hat{y})$ across all possible cut-points k :

$$\mathcal{S}_{\text{trim}} \triangleq \sum_{k=1}^T \text{Cov}_{\pi_l} (\pi_h(k | \hat{y}), G_k(\hat{y})). \quad (13)$$

A positive $\mathcal{S}_{\text{trim}}$ implies the Meta-Refiner correctly identifies cut-points k that yield higher regeneration gains.

Definition D.3 (Intervention Volume). Let $\mathcal{V}_{\text{inter}}$ represent the total probability mass allocated to the trimming process (the rejection rate):

$$\mathcal{V}_{\text{inter}} \triangleq 1 - Z_{\text{acc}} = \mathbb{E}_{\pi_l}[1 - \alpha(y)]. \quad (14)$$

This term dictates the magnitude of the opportunity space available for the Trimmer to act.

Substituting these definitions into the total gain equation yields the following decomposition:

$$\Delta J = \mathcal{A}_{\text{prec}} + \mathcal{V}_{\text{inter}} \cdot (\mathcal{S}_{\text{trim}} + \bar{G}). \quad (15)$$

We leverage GRPO with meta-actions to jointly optimize the Actor and the Meta-Refiner. By treating each trajectory y_i as an augmented execution trace, comprising both the reasoning tokens from π_l and the meta-actions sampled from the Discriminator $\pi_d(\hat{y})$ and Trimmer $\pi_h(k|\hat{y})$. GRPO inherently maximizes ΔJ . This formulation ensures that the policy gradient updates align with the maximization of $\mathcal{A}_{\text{prec}}$ and $\mathcal{S}_{\text{trim}}$.

E Supplementary Implementation Details

Hardware All experiments were conducted on multiple 8-node GPU clusters. Each node features dual-socket AMD EPYC 9K84 processors, providing a total of 192 physical cores and 384 threads per node, organized into two NUMA nodes. Storage infrastructure includes a 480 GB SATA SSD for the OS and environment, alongside two enterprise-grade 7.68 TB NVMe SSDs for high-throughput local data caching. Nodes are linked via a high-speed interconnect and share a distributed file system for dataset storage and checkpoint synchronization.

Configurations The model is trained on a unified search-integrated reasoning dataset stored in Parquet format. **Data & Rollout:** We set the maximum prompt and response lengths to 4096 and 3000 tokens, respectively. To prevent information loss, truncation is disabled; prompts exceeding the limit are filtered out. We utilize SGLang as the rollout engine to facilitate efficient multi-turn generation with tool calls, maintaining the raw chat format. Each prompt samples $n = 5$ rollout trajectories per GRPO step, with a maximum of 4 assistant turns per trajectory. The context length during rollout is capped at 15,000 tokens to accommodate interleaved reasoning and retrieved evidence. For validation, we employ greedy decoding (sampling disabled). **Optimization:** The Actor is trained via PPO-style updates using GRPO advantages. We utilize a learning rate of 1e-6 with a warmup ratio of 0.285. The global PPO mini-batch size is 512, with a per-GPU micro-batch size of 4. To stabilize training, we apply a low-variance KL penalty (coefficient 0.001) rather than incorporating it into the reward; entropy regularization is disabled. Training utilizes Fully Sharded Data Parallel (FSDP) with full state offloading. Tensor model parallelism is set to 8 for the 32B model and 2 for the 7B/8B models. **Meta-Refiner:** The Meta-Refiner functions as an internal agent sharing weights with the Actor but utilizing distinct prompts. It is trained jointly with the Actor and remains active during rollout, performing at most one revision per trajectory. Intervention decisions are determined by comparing log-probabilities of candidate actions (revision vs. no-revision); a revision is triggered only if its log-probability exceeds that of the no-revision decision (margin ≥ 0.0).

Resource Links We provide the necessary resource links of models, retrievers, and software, to help reproduce our implementation and experiments as follows: **Models:** Qwen2.5-32B-Instruct², Qwen2.5-7B³, Qwen3-8B⁴,

²<https://huggingface.co/Qwen/Qwen2.5-32B-Instruct>

³<https://huggingface.co/Qwen/Qwen2.5-7B>

⁴<https://huggingface.co/Qwen/Qwen3-8B>

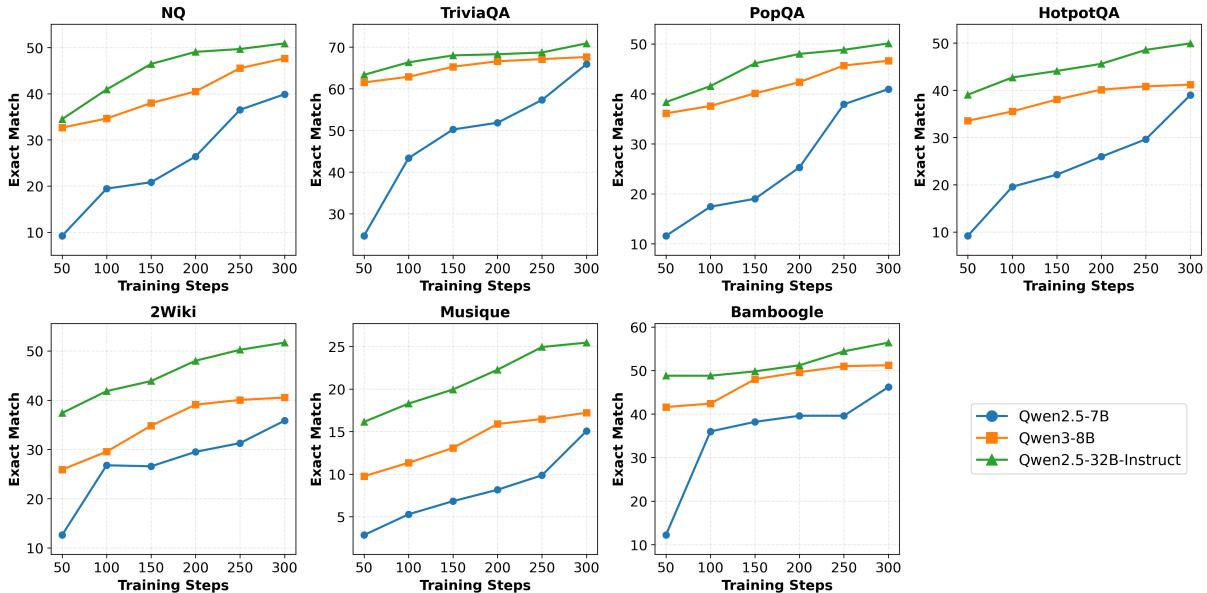


Figure 5: Detailed training dynamics of Search-R2 with different base models across all seven datasets.

and DeepSeek-R1-Distill-Qwen-7B⁵; *Retriever*: E5⁶, 2018 Wikipedia dump⁷, and index file⁸; *Softwares*: verl⁹, FSDP¹⁰, and SGLang¹¹.

F Training Dynamics

To investigate the training dynamics of our agentic RL framework, Figure 5 visualizes EM scores across the seven experimental datasets, plotted from 0 to 300 steps at 50-step intervals. We observe consistent trends across all three models and datasets, with performance converging as training approaches 300 steps. Extending training beyond this point yields negligible performance gains and increases the risk of model collapse due to instabilities such as train-inference mismatch and automatic mixed-precision overflow—challenges inherent to the current RL training infrastructure. Furthermore, while performance gaps persist between models of different sizes—confirming that parameter scale remains a critical factor in tool-use and reasoning—Search-R2 enables smaller models (e.g., Qwen2.5-7B and Qwen3-8B) to approach the performance of substantially larger models like Qwen2.5-32B-Instruct on tasks such as NQ and TriviaQA. This underscores the framework’s efficacy in enhancing search-integrated reasoning for compact models, facilitating their adoption in practical scenarios.

G Comparison against Search-R1 with Double Rollout Numbers

To verify that the performance gains of Search-R2 are not merely an artifact of increased rollout volume, we trained the Search-R1 agent with doubled rollouts ($n = 10$, compared to the default $n = 5$). This setting serves as a proxy for a naive refinement strategy where every trajectory is regenerated from scratch, in contrast to Search-R2’s targeted refinement of intermediate turns. As shown in Table 7, Search-R2 ($n = 5$, max revision = 1) consistently outperforms Search-R1 ($n = 10$) throughout the training process. At the final step 300, Search-R2 achieves a score of 50.8, surpassing Search-R1 by 6.28%. While increasing n to 10 improves Search-R1, it fails to match the performance of Search-R2. This confirms that our gains stem from the Meta-Refiner’s ability to identify and correct specific flaws, rather than simple sample scaling. Furthermore, Search-R2 is significantly more efficient: while Search-R1 ($n = 10$) requires generating 5,120 trajectories per step, Search-R2 generates approximately 3,300 on average, as the Meta-Refiner selects only $\sim 30\%$ of trajectories for revision. This reduction lowers the computational overhead from 803.2 seconds/step (Search-R1) to 469.5 seconds/step (Search-R2), demonstrating the efficiency of the Meta-Refiner module.

⁵<https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B>

⁶<https://huggingface.co/intfloat/e5-base-v2>

⁷<https://huggingface.co/datasets/PeterJinGo/wiki-18-corpus>

⁸PeterJinGo/wiki-18-e5-index

⁹<https://github.com/volcengine/verl/tree/main>

¹⁰<https://docs.pytorch.org/docs/stable/fsdp.html>

¹¹<https://github.com/sgl-project/sglang>

Table 7: Performance comparison between Search-R2 (initial rollout number as 5 per prompt and max revision as 1) and Search-R1 with double rollout numbers (10 per prompt instead of default 5 per prompt). Here, Qwen2.5-32B-Instruct is taken as the base model.

Training Steps	NQ	TriviaQA	PopQA	HotpotQA	2WikiMultiHopQA	Musique	Bamboogle	Average
Search-R1 ($n = 10$)								
50	33.9	63.3	38.2	38.7	36.6	15.7	46.4	39.0
100	38.3	65.3	40.3	41.4	40.2	17.4	45.6	41.2
150	43.8	67.6	45.2	43.7	43.9	18.9	49.6	44.7
200	46.6	68.0	47.3	44.3	44.5	21.1	48.8	45.8
250	49.0	68.3	49.1	45.2	46.6	23.2	50.4	47.4
300	49.7	68.6	49.1	45.9	47.8	24.0	49.6	47.8
Search-R2 ($n = 5$, max revision = 1)								
50	34.5	63.3	38.3	39.1	37.4	16.1	48.8	39.7
100	40.9	66.3	41.6	42.7	41.8	18.3	48.8	42.9
150	46.5	68.0	46.1	44.1	43.9	19.9	49.8	45.5
200	49.1	68.3	48.0	45.6	48.0	22.3	51.2	47.5
250	49.7	68.7	48.8	48.6	50.2	24.9	54.4	49.3
300	50.9	70.9	50.1	49.9	51.7	25.4	56.4	50.8

Table 8: The detailed ablation study results of Search-R2 with different base LLMs on seven datasets. The Meta-Refiner, process reward, and joint optimization modules are incorporated into the original Search-R1 framework in an incremental manner.

Method	NQ	TriviaQA	PopQA	HotpotQA	2WikiMultiHopQA	Musique	Bamboogle	Average
Qwen2.5-7B								
Search-R1	39.5	56.0	38.8	32.6	29.7	12.5	36.0	35.0
Search-R1 + Meta-Refiner	39.3	64.4	40.3	37.0	34.6	12.7	44.0	38.9
Search-R1 + Meta-Refiner + Process Reward	39.6	64.9	40.5	37.4	34.9	14.2	45.6	39.6
Search-R2 (Full Version)	39.9	65.9	41.0	39.0	35.8	15.1	46.2	40.4
Qwen3-8B								
Search-R1	44.0	63.1	41.8	37.2	35.5	15.7	43.0	40.0
Search-R1 + Meta-Refiner	46.2	65.9	45.1	40.2	40.2	16.2	49.6	43.4
Search-R1 + Meta-Refiner + Process Reward	46.7	66.4	45.6	40.7	40.6	16.7	51.0	44.0
Search-R2 (Full Version)	47.7	67.6	46.6	41.2	40.5	17.2	51.2	44.6
Qwen2.5-32B-Instruct								
Search-R1	47.6	68.0	47.0	43.3	46.2	22.1	45.0	45.6
Search-R1 + Meta-Refiner	50.8	69.4	49.0	47.6	49.4	24.2	54.4	49.3
Search-R1 + Meta-Refiner + Process Reward	50.1	70.0	49.4	47.6	49.9	24.3	55.6	49.5
Search-R2 (Full Version)	50.9	70.9	50.1	49.9	51.7	25.4	56.4	50.8

H Detailed Ablation Study Results

As a supplement to Section 5.3, we provide the detailed ablation study results on each dataset in Table 8.

I Supplementary Introduction to Trajectory Quality Analysis

I.1 Rubric Explanation

We evaluate trajectory quality using six rubric dimensions that capture complementary aspects of search-integrated reasoning beyond final answer correctness.

Evidence Groundedness measures whether key claims and intermediate conclusions in the trajectory are explicitly supported by retrieved information. A high score indicates that reasoning steps consistently reference or rely on evidence obtained through search, while a low score reflects unsupported claims or hallucinated content.

Information Density assesses the usefulness of retrieved information relative to the total search results. Trajectories with high information density primarily retrieve content that directly contributes to solving the task, whereas low scores indicate noisy, weakly relevant, or distracting retrievals.

Non-Redundancy Efficiency evaluates how effectively the trajectory uses its search budget. High-scoring trajectories avoid repeated or unnecessary queries and demonstrate efficient progression toward task-relevant information, while low scores reflect redundant searches or inefficient exploration.

Table 9: The trajectory quality comparison results among six rubric dimensions on seven datasets. $^{\dagger}/^*$ represents in-domain/out-of-domain datasets. All experiments are conducted on the Qwen2.5-32B-Instruct model. In each block of X/Y , X indicates the pair amounts of Search-R2 outperforms Search-R1, while Y indicates the pair amounts of Search-R1 outperforms Search-R2.

Methods	General QA				Multi-Hop QA			
	NQ †	TriviaQA *	PopQA *	HotpotQA †	2WikiMultiHopQA *	Musique *	Bamboogle *	Average
Evidence Groundedness	24/1	27/0	20/1	24/4	8/6	7/2	25/3	19.3/2.4
Information Density	37/4	28/1	35/4	40/9	37/10	32/10	46/6	36.4/6.3
Non-Redundancy Efficiency	35/3	28/0	32/3	36/7	34/7	20/8	39/5	32.0/4.7
Query Timing Quality	16/0	17/0	9/1	9/2	5/1	30/0	13/2	14.1/0.9
Trajectory Coherence	35/3	29/1	32/4	34/7	30/6	23/6	36/4	31.3/4.4
Uncertainty Handling	10/0	20/1	8/1	10/2	0/5	3/2	11/1	8.9/1.7

Query Timing Quality captures whether searches are issued at appropriate moments and whether the queries are well-formed. High scores correspond to timely searches with precise, informative queries, whereas low scores indicate poorly timed searches or vague and uninformative query formulations.

Trajectory Coherence measures the global consistency of the reasoning process. A coherent trajectory maintains alignment between early hypotheses, retrieved evidence, and final conclusions, while incoherent trajectories exhibit logical drift, contradictions, or premature commitment to incorrect assumptions.

Uncertainty Handling evaluates how the model responds to incomplete or ambiguous information. High-scoring trajectories appropriately acknowledge uncertainty, seek additional evidence, or hedge conclusions when warranted, whereas low scores indicate overconfident conclusions unsupported by sufficient evidence.

I.2 Detailed Results

Complementing the analysis in Section 5.6, Table 9 presents the full trajectory quality comparison across seven datasets. Across the six evaluation rubrics, the frequency with which Search-R2 outperforms Search-R1 is significantly higher than the reverse on most datasets. This confirms that our Actor-Refiner collaboration mechanism effectively facilitates the generation of higher-quality search-integrated reasoning trajectories.

I.3 Evaluation Prompt

To enhance the reproducibility, we provide the prompt for trajectory quality comparison in Table 10.

J Pseudocode for LLM Response Rollout with Multi-Turn Search

We provide the pseudocode for the standard search-integrated reasoning (original Search-R1) in Algorithm 2.

Algorithm 2 LLM Response Rollout with Multi-Turn Search

Require: Input x , policy π_{θ} , search engine Λ , budget B .

Ensure: Final response \hat{y} .

```

1:  $\hat{y} \leftarrow \emptyset$ ,  $b \leftarrow 0$ 
2: while  $b < B$  do
3:   Generate  $\hat{y}_b$  until  $</\text{search}>$ ,  $</\text{answer}>$ , or EOS.
4:    $\hat{y} \leftarrow \hat{y} + \hat{y}_b$ 
5:   if  $<\text{search}>$  in  $\hat{y}_b$  then
6:     Extract query  $\lambda$ ; Retrieve  $I = \Lambda(\lambda)$ 
7:      $\hat{y} \leftarrow \hat{y} + <\text{information}>I</\text{information}>$ 
8:   else if  $<\text{answer}>$  in  $\hat{y}_b$  then
9:     return  $\hat{y}$ 
10:  end if
11:   $b \leftarrow b + 1$ 
12: end while
13: return  $\hat{y}$ 

```

K Local Process Reward Implementation Details

Local Process Reward (r_{process}) quantifies the *information density* of the retrieved evidence, ensuring that the model is incentivized to perform efficient, non-redundant, and relevant searches. We compute the process reward using a

SYSTEM: You are a STRICT evaluator for trajectory quality comparison in search -integrated reasoning. You will compare two trajectories (A and B) for the SAME question.

Rules:

- Do NOT use outside knowledge. Judge only from what the trajectories show.
- Score EACH trajectory independently on each rubric dimension using 0/1/2:
0=poor, 1=acceptable/mixed, 2=strong.
- If A is slightly better than B on a dimension, assign A a higher score even if both are acceptable.
- Avoid giving identical scores unless the two trajectories are truly indistinguishable on that dimension.
- After scoring, choose the overall winner (A or B). Output 'Tie' ONLY if nearly identical in outcome AND process.
- A/B are just labels.

Return a SINGLE valid JSON object only (no markdown), following the schema exactly.

USER:

QUESTION: {question}

TRAJECTORY A (JSON): {traj_a}

TRAJECTORY B (JSON): {traj_b}

Rubric dimensions: evidence_groundedness, information_density,
non_redundancy_efficiency, query_timing_quality, trajectory_coherence,
uncertainty_handling

Output JSON schema (MUST be valid JSON):

```
{
  "winner": "A" | "B" | "Tie",
  "confidence": 0-100,
  "scores": {
    "A": {
      "evidence_groundedness": 0|1|2,
      "information_density": 0|1|2,
      "non_redundancy_efficiency": 0|1|2,
      "query_timing_quality": 0|1|2,
      "trajectory_coherence": 0|1|2,
      "uncertainty_handling": 0|1|2
    },
    "B": {
      "evidence_groundedness": 0|1|2,
      "information_density": 0|1|2,
      "non_redundancy_efficiency": 0|1|2,
      "query_timing_quality": 0|1|2,
      "trajectory_coherence": 0|1|2,
      "uncertainty_handling": 0|1|2
    }
  },
  "reasons": [ "reason1", "reason2", "reason3" ]
}
```

Decision procedure:

- 1) Score A and B independently on all rubric dimensions.
- 2) Decide winner primarily by outcome quality, if similar, decide by process quality. 3) Use 'Tie' only if truly indistinguishable.

Notes: - reasons: at most 3 short bullet strings.

Table 10: Prompt for trajectory quality comparison.

```

SYSTEM:
You are evaluating numbered collections of retrieved documents to determine
their usefulness in answering a given question, where each collection is
enclosed within <collection_x> and </collection_x> tags containing
documents that belong to that collection. Given the question and its
correct answer, mark each collection as "useful" (yes) or "not useful" (no)
based on these criteria:
(1) A collection is useful if it contains information or clues that help
    identify the correct answer, even partially.
(2) A collection is not useful if it's completely irrelevant to the question.
(3) A collection is not useful if it merely duplicates information from
    previous collections without adding new insights, even if that information
    would otherwise be relevant. After evaluating all collections strictly
    according to these criteria and the provided information, report the total
    count of collections marked as useful.

USER:
Question: {question}

Answer: {answer}

{M} collections of the tool responses:

<collection_1>
Doc 1: {retrieved_info_1_top_1}
Doc 2: {retrieved_info_1_top_2}
.....
Doc k: {retrieved_info_1_top_k}
</collection_1>

.....
<collection_{M}>
.....
Doc k: {retrieved_info_{M}_top_k}
</collection_{M}>

Provide your answer strictly in the format:
Final Answer: number

```

Table 11: LLM Judge Prompt for Chunk Utility u_i . The system prompt enforces strict criteria for relevance and non-redundancy.

density-based approach evaluated by an external LLM judge (DeepSeek-R1-Distill-Qwen-7B in our experiments). Inference is performed via the vLLM framework with greedy decoding parameters: temperature set to 0.0, top-p at 0.95, a repetition penalty of 1.0, and a maximum token limit of 3,000. The evaluation procedure proceeds as follows:

1. *Collection Grouping.* For a given reasoning trajectory y , we identify all search tool invocations. The top- k documents returned by a single search query are grouped into a single *collection*, denoted as c_i . If a trajectory contains M search actions, we have a set of collections $C = \{c_1, \dots, c_M\}$.
2. *Judge Evaluation.* We construct a prompt provided in Table 11 containing the user question, the ground truth answer, and the chronological list of collections. The judge evaluates each collection c_i against three strict criteria:
 - **Useful** ($u_i = 1$): The collection contains information or clues that help identify the correct answer, even partially.
 - **Not Useful** ($u_i = 0$): The collection is completely irrelevant.
 - **Redundant** ($u_i = 0$): The collection merely duplicates information from previous collections ($c_{1\dots i-1}$) without adding new insights, even if the information is relevant.
3. *Density Computation.* The judge outputs the total count of useful collections. The process reward is

calculated as the ratio of useful collections to total search actions:

$$r_{\text{process}}(y) = \frac{1}{M} \sum_{i=1}^M u_i. \quad (16)$$

4. *Outcome Gating.* To prevent "reward hacking" where an agent maximizes retrieval scores without solving the task, the process reward is applied only when the final answer is correct. The total reward $R(y)$ is defined as:

$$R(y) = r_{\text{outcome}}(y) \cdot (1 + r_{\text{process}}(y)), \quad (17)$$

where $r_{\text{outcome}}(y)$ is the binary Exact Match (EM) score.

L Meta-Refiner Prompt

You are a meticulous meta-thinker. Review the numbered ASSISTANT_STEP entries and identify the earliest flawed step. Return a single integer between 0 and {max_steps} where 0 means all steps are acceptable.

ASSISTANT_CONTEXT: <assistant turns before current rollout>

USER: <user message>

ASSISTANT_STEP_1: <assistant turn 1>

TOOL: <tool output triggered by step 1 (if any)>

ASSISTANT_STEP_2: <assistant turn 2>

TOOL: <tool output triggered by step 2 (if any)>

...

Problematic step index (0 = no issue):

Table 12: Prompt for Meta-Refiner.

We provide the prompt for Meta-Refiner in Table 12.