

TREEPS-RAG: Tree-based Process Supervision for Reinforcement Learning in Agentic RAG

Tianhua Zhang^{♡*}, Kun Li^{♡*}, Junan Li^{♡*}, Yunxiang Li[♡],
Hongyin Luo[◇], Xixin Wu[♡], James Glass[◇], Helen Meng[♡]

[♡]The Chinese University of Hong Kong, Hong Kong SAR, China

[◇]Massachusetts Institute of Technology, Cambridge MA, USA

{thzhang, li.kun}@link.cuhk.edu.hk, jli@se.cuhk.edu.hk

Abstract

Agentic retrieval-augmented generation (RAG) formulates question answering as a multi-step interaction between reasoning and information retrieval, and has recently been advanced by reinforcement learning (RL) with outcome-based supervision. While effective, relying solely on sparse final rewards limits step-wise credit assignment and provides weak guidance for intermediate reasoning and actions. Recent efforts explore process-level supervision, but typically depend on offline constructed training data, which risks distribution shift, or require costly intermediate annotations. We present **TREEPS-RAG**, an online, tree-based RL framework for agentic RAG that enables step-wise credit assignment while retaining standard outcome-only rewards. Our key insight is to model agentic RAG reasoning as a rollout tree, where each reasoning step naturally maps to a node. This tree structure allows step utility to be estimated via Monte Carlo estimation over its descendant outcomes, yielding fine-grained process advantages without requiring intermediate labels. To make this paradigm practical, we introduce an efficient online tree construction strategy that preserves exploration diversity under a constrained computational budget. With a rollout cost comparable to strong baselines like Search-R1, experiments on seven multi-hop and general QA benchmarks across multiple model scales show that TREEPS-RAG consistently outperforms both outcome-supervised and leading process-supervised RL methods.

1 Introduction

Recent progress in large-scale generative language models has enabled strong performance across many NLP tasks (Brown et al., 2020; Ouyang et al., 2022). However, reliability remains a key bottleneck: models can hallucinate and often lack verifiable provenance for factual claims, especially in

knowledge-intensive settings such as open-domain question answering (Huang et al., 2025b; Zhang et al., 2025b). Retrieval-augmented generation (RAG) mitigates these issues by grounding generation on externally retrieved evidence (Lewis et al., 2021; Gao et al., 2024; Fan et al., 2024).

Conventional RAG pipelines often rely on single-turn retrieval or relatively static workflows, making them less effective for complex questions that require iterative reasoning or dynamic retrieval intent. This has motivated *agentic RAG* (Singh et al., 2026), where Large Language Models (LLMs) interleave reasoning with information-seeking actions, often through ReAct-style loops (Yao et al., 2023). Beyond prompt-only approaches (Trivedi et al., 2023; Madaan et al., 2023; Gou et al., 2024), recent work increasingly treats agentic RAG as a trainable decision-making problem (Asai et al., 2023; Li et al., 2025a). In particular, outcome-supervised reinforcement learning has emerged as a scalable paradigm for eliciting reasoning and planning behaviors (DeepSeek-AI, 2025), and has also been adapted to train search-enabled agents that learn when and how to retrieve external information (Chen et al., 2025b; Song et al., 2025). Within this line of work, Search-R1 serves as one of the most representative approaches: it optimizes the policy for multi-turn search-generation interactions using trajectory-level rewards determined by final answer correctness (Jin et al., 2025).

Despite the promise of agentic RAG, prevailing methods execute step-by-step decisions yet collapse credit assignment into an outcome-only signal over the entire trajectory (e.g., final answer correctness) (Song et al., 2025; Jin et al., 2025; Luo et al., 2025). Such delayed and sparse supervision exacerbates credit assignment: intermediate decisions are updated as if they were equally responsible for the terminal outcome, even though only a subset of steps are truly decision-critical. This raises our central question: *can process su-*

* Equal contribution.

pervision improve the learning of search-enabled agents beyond outcome-only optimization? A natural approach is to introduce process supervision via step-wise rewards; however, as summarized in [Tran et al. \(2025\)](#), recent step-reward formulations often face two practical limitations: (i) they require explicit step-level annotations to supervise intermediate decisions ([Zheng et al., 2025](#)), and (ii) the assignment of process rewards is often carried out offline (post hoc over collected trajectories), instead of being integrated into online RL rollouts for interactive agents ([Zhang et al., 2025a](#)).

To address these limitations, we introduce **TREEPS-RAG (Tree-based Process-Supervised RAG)**, a tree-structured reinforcement learning approach for agentic retrieval that provides process-level supervision without intermediate annotations or auxiliary reward models. The key idea is to model the agentic RAG rollouts as a tree formulation, where each node corresponds to a complete agent-environment interaction step comprising reasoning, an action, and the resulting observation when applicable. During training, TREEPS-RAG assigns node-level supervision on the fly via Monte Carlo estimation over descendant leaves: it samples multiple continuations from a node, scores terminal outcomes by exact match (EM), and propagates the aggregated outcome signal back as the node’s value for advantage calculation. This yields denser step-wise supervision while preserving standard outcome-based policy optimization. To keep online tree expansion computationally tractable, we further introduce similarity-based pruning over retrieved evidence, pruning nodes whose retrieved document sets are highly overlapping to control the branching factor while maintaining diverse paths. We evaluate TREEPS-RAG on seven QA benchmarks under four backbone LLMs, and observe consistent improvements over competitive baselines across datasets and models, indicating robust benefits from process supervision for agentic RAG.

In summary, our contributions are threefold: (i) we formulate agentic RAG rollouts as online trees whose nodes represent complete reasoning-action-observation interactions, enabling effective exploration at a cost comparable to outcome-based RL; (ii) we derive online, annotation-free process supervision from descendant terminal outcomes without auxiliary reward/value models; and (iii) on seven QA benchmarks, our method consistently outperforms competitive baseline training approaches.

2 Related Work

Agentic RAG Early agentic RAG methods mainly relied on prompt engineering and self-refinement mechanisms ([Madaan et al., 2023](#); [Gou et al., 2024](#)), which include interleaving retrieval with LLM reasoning-based generation iteratively to solve complex knowledge-intensive questions ([Trivedi et al., 2023](#); [Jiang et al., 2023](#); [Shao et al., 2023](#)), and adaptive RAG that encourages LLMs to decide when and how to retrieve ([Jeong et al., 2024](#); [Wang et al., 2025](#)). Recent methods advanced towards trainable agents such as Self-RAG ([Asai et al., 2023](#)) and Auto-RAG ([Yu et al., 2024](#)). Methods ([Jin et al., 2025](#); [Song et al., 2025](#); [Chen et al., 2025b](#); [Sun et al., 2025](#)) applied GRPO ([Shao et al., 2024](#)) or PPO ([Schulman et al., 2017](#))-based reinforcement learning outcome-only supervision to train search-capable agents. While effective, these methods rely on sparse, delayed rewards, which limits fine-grained credit assignment. This motivates the exploration of process-supervised RL such as ReasonRAG ([Zhang et al., 2025a](#)), GiGPO ([Feng et al., 2025](#)) and StepSearch ([Zheng et al., 2025](#)), to provide step-wise guidance during policy learning.

Tree Structure Tree-structured reasoning ([Huang et al., 2025a](#)) has been extensively studied in LLM research, including data synthesis, offline alignment, and enhanced credit assignment. Previous works ([Zhang et al., 2025a](#); [Xie et al., 2024](#)) employ Monte Carlo Tree Search (MCTS) to yield step-level supervision signals or preference pairs for downstream training, such as DPO ([Rafailov et al., 2024](#)). Our tree is constructed online during RL training, not used as an inference-time search procedure. More recent studies exploit rollout trees to densify credit assignment during RL training on math or code reasoning problems, such as TreePO ([Li et al., 2025b](#)), TreeRPO ([Yang et al., 2025c](#)), TreeRL ([Hou et al., 2025](#)) and DART ([Li et al., 2026](#)). TREEPS-RAG likewise employs tree-structured rollouts. However, prior methods primarily target math or code reasoning and construct trees over model-internal textual generation, with branches defined at tokens or selected reasoning positions. Agentic RAG instead exhibits an explicit interaction structure: each node in our formulation represents a complete reasoning-action-observation step involving both the agent and an external retrieval environment. This distinction also changes the source of rollout

redundancy, as textually different search actions may retrieve highly overlapping evidence. We therefore introduce document similarity-based pruning to preserve evidence diversity under a fixed rollout budget. TREEPS-RAG thus extends the online tree-based process-supervised framework to agentic RAG.

3 Methodology

We present TREEPS-RAG, an online tree-based reinforcement learning framework for agentic RAG that enables process-level credit assignment without requiring intermediate annotations, auxiliary reward or value models. Through tree exploration, TREEPS-RAG improves step-wise supervision without compromising the efficiency of outcome-based annotation. We first formalize the conceptual approach: modeling agentic RAG inference during rollout as tree construction (§3.1), introduce an online tree construction strategy (§3.2), and finally detail how to derive process-level supervision (§3.3). Figure 1 and Algorithm 1 illustrate the workflow.

3.1 Tree Modeling of Agentic RAG

Agentic RAG under the ReAct (Yao et al., 2023) paradigm solves a given question q in a sequential, step-wise manner. A language model agent π_θ alternates between reasoning, information-seeking via an external retriever, and answer generation. At each step i , the agent conditions on the current state $[x, (r_1, a_1, o_1), \dots, (r_{i-1}, a_{i-1}, o_{i-1})]$, which consists of the prompted input containing the original question (Figure 4) together with all accumulated interaction history, to produce the next reasoning segment r_i along with an action $a_i \in \{\text{search}, \text{answer}\}$. When a *search* action is generated, the agent issues the query to the retriever and receives an observation o_i , consisting of the top-K retrieved passages. A trajectory $y = [(r_1, a_1, o_1), (r_2, a_2, o_2), \dots]$ terminates when the agent produces an *answer* action or reaches a pre-defined maximum step limit. Upon termination, the agent receives a binary outcome reward, reflecting the correctness of the final answer. The objective of agentic reinforcement learning is to optimize the policy π_θ by maximizing the expected cumulative reward of rollout trajectories.

Tree Representation of Rollouts We model the agentic RAG rollout process as a tree rooted at the input prompt x . Each agentic step i , comprising the reasoning-action pair together with retrieved

observation when applicable, i.e., (r_i, a_i, o_i) , is represented as a node s_i in the tree. Nodes associated with *search* actions allow subsequent exploration, serving as non-leaf nodes, while each node with *answer* action forms a leaf node. In this way, a complete trajectory y , composed of a sequence of successive actions, corresponds to a root-to-leaf path in the tree. Under this formulation, step-wise reasoning process naturally maps to parent-child relationships—expanding from a node amounts to sampling continuations conditioned on the prefix represented by the path from root to itself:

$$(r_i, a_i) \sim \pi_\theta(\cdot \mid x, (r_1, a_1, o_1), \dots, (r_{i-1}, a_{i-1}, o_{i-1})). \quad (1)$$

Why Tree Modeling? This tree-structured formulation is particularly well-suited for agentic RAG. First, the sequential nature of agentic RAG admits a direct mapping of steps separated by actions to tree nodes. Modeling an entire agentic step as a node matches the granularity at which process supervision is needed. More crucially, under this formulation, a tree structure offers a principled mechanism for addressing the absence of step-level quality annotations: by treating the path to a given step node as a shared prefix and exploring multiple continuations from that point, the utility of the corresponding step can be estimated empirically from the outcomes of its descendant trajectories. This perspective closely aligns with Monte Carlo-style estimation, where the value of a state is inferred from the returns of sampled rollouts originating from it. Under this view, the tree is not merely a data structure, but an active computational tool that enables exploration and step-wise credit assignment for agentic RAG.

3.2 Practical Tree-structured Rollout

While the tree formulation introduced in §3.1 offers a principled view of agentic RAG rollouts, naively expanding tree nodes in an exhaustive way quickly becomes computationally intractable due to the exponential growth of the tree. To make tree-based exploration feasible under realistic training budgets, we propose an efficient online tree construction strategy that preserves the advantages of tree modeling, i.e., shared prefixes and multi-branch exploration, while maintaining a rollout cost comparable to conventional outcome-based RL methods such as Search-R1 (Jin et al., 2025). Specifically, our tree construction is governed by two core parameters: (1) a target rollout budget N ,

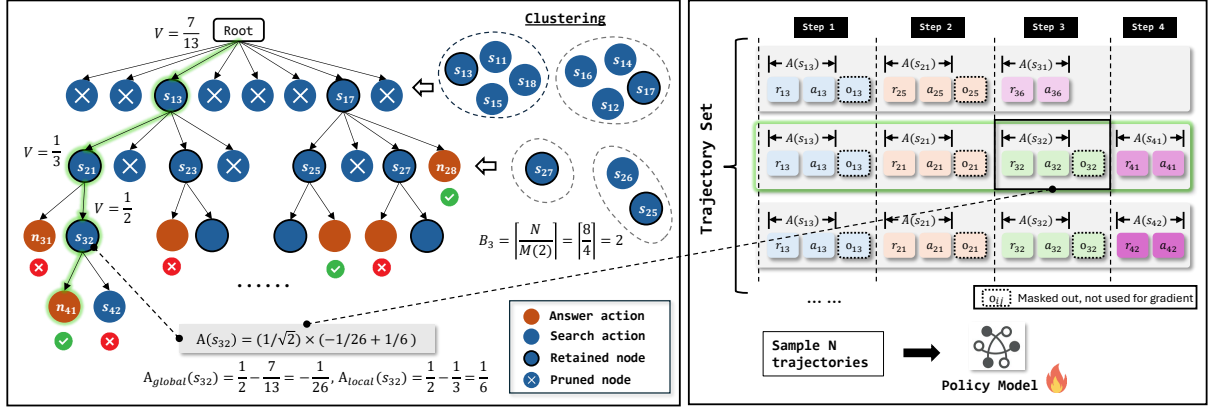


Figure 1: Overview of TREEPS-RAG. **Left:** Online tree construction and process supervision with $N = 8$, $D = 4$, $N_{\text{retain}} = 2$. Retained parent nodes are expanded with B_d continuations. Similarity-based clustering is applied among sibling *search* children to prune redundant nodes (crossed out) and retain $\hat{N}_{\text{retain}}$ diverse candidates (solid circles). Node values (V) are estimated via Monte Carlo averaging over descendant leaf outcomes to compute process advantages. **Right:** Tree-to-trajectory transformation for policy optimization. Root-to-leaf paths are collected into a set (one example highlighted in green). Process advantages are uniformly assigned to all model-generated tokens within a step (r_i, a_i), while observation tokens (o_i) are masked out. For fair comparison, only N trajectories are randomly sampled during training to maintain comparable to standard outcome-based RL methods.

chosen to match the token cost of rollout in those methods that conventionally sample N trajectories in parallel, and (2) a local retention budget N_{retain} , which controls the maximum number of non-leaf child nodes retained per non-leaf node.

Initialization The tree is initialized with the input prompt as the root s_{root} . The tree is expanded in a layer-wise manner, and let $\mathcal{M}(d)$ denote the set of non-leaf nodes at depth d that serve as parents for expansion at the next step. Initially, $\mathcal{M}(0) = \{s_{\text{root}}\}$. The expansion from depth $d - 1$ to d follows a generate-then-prune paradigm.

Child Node Generation At depth d , for each node $s_p \in \mathcal{M}(d - 1)$, the π_θ generates B_d new children with Eq. (1) in parallel. This forms a local children set $\mathcal{C}(s_p) = \{s_p^1, \dots, s_p^{B_d}\}$, where each child corresponds to a sampled continuation step conditioned on s_p and all its predecessors. The branching factor B_d is dynamically computed as:

$$B_d = \lceil N / |\mathcal{M}(d - 1)| \rceil. \quad (2)$$

This depth-wise allocation ensures that the total number of sampled nodes at each layer remains approximately N , i.e., $\forall d \in [1, D], |\mathcal{M}(d - 1)| \times B_d \approx N$.¹ As a result, the overall token cost remains comparable to sampling N independent trajectories in conventional rollout mechanisms.

¹Due to the existence of *answer* actions and the ceiling operation, this number is not strictly fixed to N but fluctuates around it, ensuring the overall computational cost remains comparable to sampling N independent trajectories.

Child Node Pruning Among the children of a given parent, each node associated with the *answer* action immediately forms a leaf and defines a complete trajectory corresponding to the root-to-itself path. For the remaining *search* children, denoted as $\mathcal{C}_{\text{search}}(s_p)$, we select a subset of size: $\hat{N}_{\text{retain}}(s_p) = \min(N_{\text{retain}}, |\mathcal{C}_{\text{search}}(s_p)|)$. These retained children across all parents collectively constitute the next-layer parent set $\mathcal{M}(d)$ (Lines 14, 15 in Algorithm 1). According to Eq. 2, the branching factor at the next depth B_{d+1} is inversely proportional to $|\mathcal{M}(d)|$. Therefore, to maintain a sufficient B_{d+1} for each parent in the next layer, we need to control the size of $\mathcal{M}(d)$ through selective retention. An ideal pruning strategy is to prioritize diversity among $\mathcal{C}_{\text{search}}(s_p)$. Since sibling nodes share an identical prefix, highly similar intermediate steps are likely to induce similar continuations in subsequent layers, resulting in redundant chains. Such redundancy reduces effective exploration under a limited rollout budget. Therefore, we adopted a similarity-based pruning strategy over $\mathcal{C}_{\text{search}}(s_p)$ to retain $\hat{N}_{\text{retain}}(s_p)$ child nodes.

Similarity-based Pruning Our premise is that semantic intent of a *search* action can be captured by the information it retrieves. Hence, for any two sibling *search*-action nodes, s_i and s_j , let P_i and P_j denote the top- K retrieved passage sets associated with their generated search queries. We quantify their similarity using the **Jaccard Similar-**

ity: $J(i, j) = \frac{|P_i \cap P_j|}{|P_i \cup P_j|}$. A larger value near 1 indicates redundant information-seeking, while a value near 0 suggests exploration of distinct knowledge. Armed with this metric, we perform pruning independently for children under each parent. Specifically, for the candidate set $\mathcal{C}_{\text{search}}(s_p)$, we compute pairwise Jaccard distances $\delta(i, j) = 1 - J(i, j)$, and apply **hierarchical clustering** to partition the nodes into $\hat{N}_{\text{retain}}(s_p)$ clusters. From each cluster, a single representative node is retained for expansion to the next layer, while the remaining nodes are pruned. By pruning for semantic redundancy, we ensure that our fixed computational budget is allocated to exploring broader search space.

3.3 Process Supervision

Upon constructing the tree structure of rollout, we transform the sparse outcome rewards into dense, step-wise advantages.

Value Estimation Following prior agentic-RAG RL work (Jin et al., 2025), we adopt the rule-based outcome reward with exact match score for each leaf node s_{leaf} in the tree:

$$V(s_{\text{leaf}}) = \text{EM}(a_{s_{\text{leaf}}}, a_{\text{gold}})$$

where $V(s_{\text{leaf}}) = 1$ if the extracted answer $a_{s_{\text{leaf}}}$ from s_{leaf} matches the ground truth exactly, and 0 otherwise. For each non-leaf node s_{nf} , rather than learning a parametric value function, we employ Monte Carlo (MC) estimation to back-calculate the value from its descendant leaves. Formally, let $\mathcal{L}(s_{nf})$ be the set of all descendant leaves of s_{nf} . The value $V(s_{nf})$ is defined as:

$$V(s_{nf}) = \frac{1}{|\mathcal{L}(s_{nf})|} \sum_{s \in \mathcal{L}(s_{nf})} V(s).$$

Intuitively, $V(s_{nf})$ reflects the estimated probability that continuing from step s_{nf} will eventually lead to a correct answer under current policy.

Advantage Calculation Inspired by recent work in tree-based credit assignment (Hou et al., 2025; Tran et al., 2025), we define process-level supervision in terms of advantages using these estimated node values, capturing how beneficial a specific step is relative to appropriate baselines.

Global advantage measures a step’s quality relative to the overall performance on the question:

$$A_{\text{global}}(s) = V(s) - V(s_{\text{root}}),$$

where $V(s_{\text{root}})$ is the average expected rewards over all trajectories in the tree for the query.

Local advantage isolates the relative contribution of a step with respect to its immediate predecessor:

$$A_{\text{local}}(s) = V(s) - V(p(s)),$$

where $p(s)$ denotes the parent of s . This reflects whether the transition to s_i improves or degrades the expected reward, and a greater $A_{\text{local}}(s)$ implies that s is more likely than its sibling nodes to produce a correct outcome.

The final process advantage of a step is the normalized sum of its global and local advantages:

$$A(s) = \frac{1}{\sqrt{|\mathcal{L}(s)|}} (A_{\text{global}}(s) + A_{\text{local}}(s)).$$

The scaling factor $|\mathcal{L}(s)|^{-1/2}$ is to prevent overfitting by down-weighting the advantage of non-leaf nodes based on the size of their subtree, as these nodes may appear in multiple rollout trajectories and thus be repeatedly computed.

Policy Optimization To derive experience for updating the policy, we need to convert the tree into a set of trajectories. Specifically, each leaf node corresponds to the end of a complete rollout. Backtracking from the leaf to the root yields a full agentic trajectory, as a sequence of steps. From the collection of all such paths, we randomly sample N trajectories to form the experience set $Y = \{y_1, \dots, y_N\}$ for updating the policy. Crucially, the tokens within y inherit the advantages of their corresponding nodes $A(s)$. This produces a token-level signal that reflects the step-wise credit induced by the tree, in contrast to trajectory-level uniform signals used in GRPO-based methods. With these token advantages, we can update the policy using standard policy gradient methods. In this work, we use a PPO-style (Schulman et al., 2017) training objective as Eq. 3.²

4 Experiments

4.1 Experimental Settings

Datasets & Metrics Following Jin et al. (2025), we evaluate TREEPS-RAG on seven QA benchmarks, categorized as follows: (1) three single-hop QA datasets, including NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA

²Following prior work (Jin et al., 2025; Song et al., 2025), we apply a loss mask to the observation tokens (retrieved passages).

Methods	Multi-hop QA				General QA		
	HotpotQA	2Wiki	MusiQue	Bamboogle	TriviaQA	PopQA	NQ
Qwen2.5-7B-Instruct							
Rejection Sampling [†] (Jin et al., 2025)	0.331	0.296	0.123	0.355	0.592	0.380	0.360
Search-R1-GRPO [†] (Jin et al., 2025)	0.386	0.346	0.162	0.400	0.623	0.427	0.429
Search-R1-PPO [†] (Jin et al., 2025)	0.370	0.414	0.146	0.368	0.610	0.397	0.393
ReasonRAG [†] (Zhang et al., 2025a)	0.384	0.436	0.128	0.360	-	0.415	-
StepSearch [†] (Zheng et al., 2025)	0.386	0.366	0.226	0.400	-	-	-
GiGPO [†] (Feng et al., 2025)	0.416	0.436	0.189	0.408*	0.647	0.461	0.464
Ours	0.507	0.482	0.239	0.520	0.676	0.485	0.521
Qwen2.5-3B-Instruct							
Rejection Sampling [†] (Jin et al., 2025)	0.240	0.233	0.059	0.210	0.488	0.332	0.294
Search-R1-GRPO [†] (Jin et al., 2025)	0.331	0.310	0.124	0.232	0.565	0.391	0.397
Search-R1-PPO [†] (Jin et al., 2025)	0.324	0.319	0.103	0.264	0.545	0.378	0.341
ReasonRAG [†] (Zhang et al., 2025a)	0.300	0.266	0.069	0.136	-	0.329	-
StepSearch [†] (Zheng et al., 2025)	0.345	0.320	0.174	0.344	-	-	-
GiGPO [†] (Feng et al., 2025)	0.369	0.370	0.126	0.304*	0.595	0.424	0.420
Ours	0.462	0.456	0.215	0.440	0.636	0.460	0.476
Qwen3-8B (non-thinking mode)							
Search-R1-GRPO	0.468	0.503	0.218	0.520	0.670	0.462	0.452
Search-R1-PPO	0.452	0.499	0.213	0.504	0.674	0.450	0.461
ReasonRAG*	0.388	0.450	0.142	0.392	-	0.274	-
GiGPO	0.411	0.450	0.175	0.432	0.662	0.431	0.458
Ours	0.487	0.514	0.238	0.520	0.687	0.477	0.489
Qwen3-4B-Instruct-2507							
Search-R1-GRPO	0.474	0.517	0.225	0.536	0.675	0.462	0.447
Search-R1-PPO	0.454	0.507	0.215	0.480	0.662	0.451	0.433
GiGPO	0.424	0.428	0.184	0.472	0.643	0.467	0.449
Ours	0.480	0.541	0.233	0.536	0.680	0.488	0.476

Table 1: **Main Results.** [†] indicates results cited from the original work. The best performance is set in **bold**. * denotes specially reproduced results with details in Appendix C.4. We do not re-implement StepSearch on Qwen3, because it relies on prior process-based annotation and the provided annotation is specific to Qwen2.5.

(Mallen et al., 2023), and (2) four multi-hop QA datasets, including HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), Bamboogle (Press et al., 2023), and MuSiQue (Trivedi et al., 2022). We use Exact Match (EM) as the primary evaluation metric across all benchmarks.

Baselines We evaluate TREEPS-RAG against a set of competitive and representative training baselines. We include (1) Search-R1 (Jin et al., 2025), our primary baseline, that trains models using GRPO and PPO under outcome-only supervision. We further compare against several recent methods that incorporate step-level supervision: (2) StepSearch (Zheng et al., 2025) leverages step-wise PPO by training on MuSiQue dataset augmented with intermediate subquestion-level annotations, which are generated by GPT-4o to provide process-level rewards. (3) ReasonRAG (Zhang et al., 2025a) constructs an offline process-supervised dataset via Monte Carlo Tree Search (MCTS) exploration and subsequently applies DPO to optimize the policy on this fixed dataset. (4) GiGPO

(Feng et al., 2025) augments episode-level GRPO by introducing an anchor-based grouping mechanism over repeated environment states, enabling step-level credit assignment during training. See implementation details in Appendix C.

We conduct comparisons using Qwen2.5 (Yang et al., 2025b) and Qwen3 (Yang et al., 2025a) backbone models. For the Qwen2.5 series, we cite baseline results from the original papers because their associated training data and supervision protocols are integral to the methods and are not directly interchangeable. We use 12K randomly sampled training instances from HotpotQA and NQ with rollout group size $N = 8$ to train TREEPS-RAG, while the original works of Search-R1 and GiGPO train models with $N = 5$ on full HotpotQA and NQ training split (170K). Thus, the cited baselines use substantially more training data and a larger overall RL experience budget. To provide controlled evidence under Qwen2.5, Appendix D.1 reports experiments with Qwen2.5-3B Base in which all compared methods use identical training data, roll-

Methods	HotpotQA	2Wiki	MusiQue	Bamboogle	TriviaQA	PopQA	NQ	Avg
Search-R1-GRPO	0.474	0.517	0.225	0.536	0.675	0.462	0.447	0.477
Ours	0.480	0.541	0.233	0.536	0.680	0.488	0.476	0.491
w/o process advantage	0.463	0.523	0.216	0.472	0.674	0.470	0.454	0.467
w/o process advantage & similarity pruning	0.430	0.509	0.211	0.464	0.677	0.463	0.450	0.458
w/o local advantage	0.478	0.526	0.232	0.496	0.677	0.478	0.477	0.481
w/o global advantage	0.467	0.523	0.227	0.504	0.678	0.481	0.469	0.478

Table 2: Ablation study on the effectiveness of each component, with Qwen3-4B-Instruct-2507.

out group size and optimization configuration. For Qwen3, we re-implement some of the baselines, as their original papers do not report the results based on Qwen3. A unified setting with the same 12K training instances, rollout group size $N = 8$, is used across all methods. See Appendix C.4 for more details.

4.2 Main Results

As shown in Table 1, **TREEPS-RAG consistently outperforms all competitive training baselines across seven QA benchmarks**, achieving the best average performance with Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-8B and Qwen3-4B-Instruct-2507. The performance gains remain stable across four backbone models on both in-distribution and out-of-domain benchmarks.

Compared to outcome-supervised RL baselines Search-R1, TREEPS-RAG achieves consistent improvements, demonstrating the benefit of our process supervision. This improvement will be discussed in detail in §5.1 and 5.2. Under the controlled Qwen2.5-3B Base setting in Appendix D.1, TREEPS-RAG achieves an average EM of 38.34, compared with 33.02 for Search-R1-GRPO and 30.61 for Search-R1-PPO. This shows that the gains of TREEPS-RAG are not tied to instruction-tuned backbones.

Beyond outperforming outcome-only baselines, TREEPS-RAG also surpasses existing process-supervised approaches. ReasonRAG relies on process supervision annotated via MCTS prior to training; its inferior performance relative to RL-based methods underscores the advantage of the online RL paradigm in mitigating the distribution shift (Shenfeld et al., 2025; Chen et al., 2025a). Among RL methods, StepSearch utilizes process supervision from GPT-4o-generated labels. In contrast, our method achieves superior performance without any external labeling. This demonstrates that effective process supervision can be derived from outcome-only annotations via tree structures, eliminating the need for costly step-level annotations. TREEPS-

RAG consistently outperforms GiGPO—which enables stepwise credit assignment through anchor-based grouping—indicating that our tree structure yields superior process supervision.

5 Analysis

We validate TREEPS-RAG from several key aspects: the effectiveness of rollout mechanisms, advantage assignment and tree scale, and then provide comparative results against Search-R1 on qualitative performance, training latency and training dynamics under matched settings.

5.1 Effectiveness of Tree-structured Rollout

Impact of Tree-structured Rollout on RL In reinforcement learning, trajectory diversity of rollout is crucial for providing discriminative learning signals. While our shared-prefix branching may intuitively reduce exploration compared to independent parallel sampling (as in Search-R1), we investigate whether this affects learning efficacy. To quantify this, we conduct an ablation by removing the process advantage assignment. In this variant, the policy is updated using standard GRPO advantages, thereby differing from Search-R1-GRPO solely in the rollout mechanism. As in Table 2, Ours w/o process advantage achieves performance comparable to Search-R1-GRPO, despite sampling trajectories from a shared-prefix tree structure. The absence of noticeable performance degradation suggests that **our tree-structured rollout effectively preserves sufficient exploration under a comparable computational budget**.

Similarity-based Pruning Further removing similarity pruning (retaining child nodes randomly) leads to a consistent performance drop on most datasets, with especially pronounced degradation on multi-hop benchmarks. This indicates that the similarity pruning is crucial for our tree-structured rollout to maintain exploration. Appendix D.3 assesses alternative similarity metrics and clustering strategies, confirming that the benefit of similarity

Tree	HotpotQA	2Wiki	MuSiQue	Bamboogle	TriviaQA	PopQA	NQ	Avg
Default (Dynamic Tree)	0.480	0.541	0.233	0.536	0.680	0.488	0.476	0.491
Larger Tree	0.490	0.531	0.239	0.536	0.688	0.487	0.494	0.495

Table 3: Impact of Tree Scale. We compare the default dynamic branching strategy in §3.2 ($B_d = 8, 4, 2, 1$; $N_{\text{retain}} = 2$) against a scaled-up variant ($B_d = 9, 7, 5, 1$; $N_{\text{retain}} = 3$). To ensure a fair comparison, the total number of sampled trajectories is kept constant at $N = 8$ during RL training for both configurations.

pruning is robust to reasonable design choices.

5.2 Effectiveness of Tree-based Process Advantage

Process Advantage Comparing the full method (Ours) with Ours w/o process advantage in Table 2 reveals a consistent and notable improvement across all datasets. This performance lift can be directly attributed to the effectiveness of the process advantage derived from the trees. By providing a fine-grained, step-wise credit assignment signal, **our method enables more precise and reliable policy optimization** than the uniform, trajectory-level feedback of standard GRPO.

Local and Global Advantage Components We further ablate the two components in our process advantage formulation to understand where the step-wise supervision comes from. As shown in Table 2, removing either the local or global advantage consistently lowers the average performance. Their combination provides complementary supervision: the global term calibrates steps to the tree-level expectation, whereas the local term privileges steps exhibiting superior utility against their siblings.

5.3 Effect of Tree Scale

To analyze the effect of tree scale, we increase both the branching factor B_d and the retained child nodes per parent N_{retain} , but sample the same number of trajectories ($N = 8$) as experience from the constructed tree for policy optimization to ensure fair comparison. As shown in Table 3, enlarging the tree leads to an improvement in average performance ($0.491 \rightarrow 0.495$). This is expected as the quality of Monte Carlo estimation for step-wise advantages would benefit from aggregating over more leaf nodes. The bounded gain suggests that our default dynamic branching strategy already provides sufficiently informative descendant outcomes for estimating process advantages.

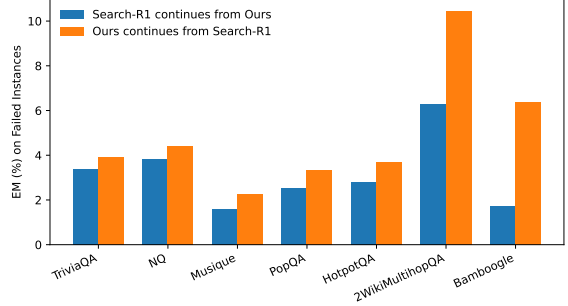


Figure 2: Analysis of continuation-based generation.

5.4 Continuation-Based Robustness Analysis

While final accuracy reflects end-to-end performance, it does not directly reveal the quality of generated steps. In the absence of step-level annotations, we introduce a continuation-based analysis to assess whether our process-supervised training leads to more reliable agentic steps. We compare TREEPS-RAG with Search-R1-GRPO by probing a model’s ability to recover from imperfect prefixes. We collect all test instances where one method fails to produce correct final answers. We truncate these failed trajectories by removing the final *answer* steps, and use the remaining prefix as the input context. The other method is prompted to continue generation from this prefix.

As shown in Figure 2, across all datasets, **TREEPS-RAG consistently yields larger performance gains when continuing from Search-R1-generated prefixes than the reverse setting**. Appendix E provides two comparative cases to show how TREEPS-RAG and Search-R1-GRPO perform under identical prefixes. Across both cases, Search-R1 fails to recover from imperfect prefixes, either by grounding answers to misleading evidence or by prematurely terminating reasoning under incomplete information. In contrast, TREEPS-RAG identifies missing or unreliable evidence, and revises the trajectory through additional verification with rewritten query. This asymmetry provides evidence that our method is more effective at producing high-quality agentic processes, even when

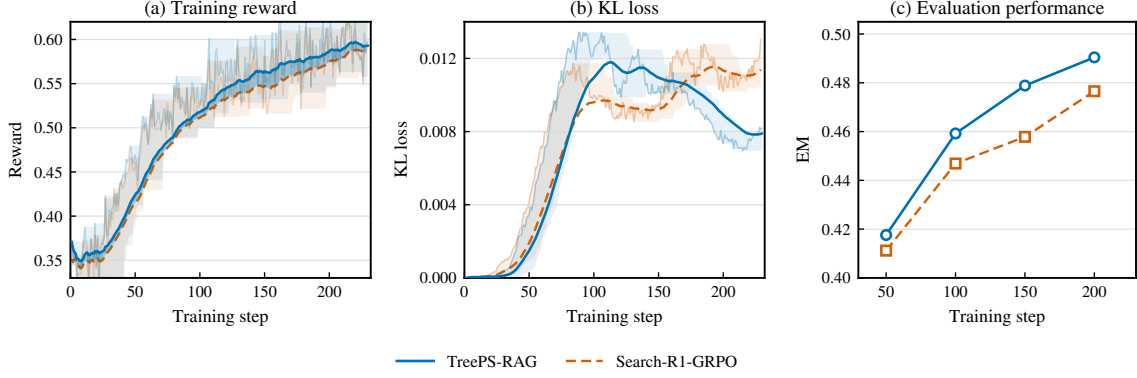


Figure 3: Training dynamics of TREEPS-RAG and Search-R1-GRPO under the matched Qwen3-4B-Instruct-2507 setting. In (a) and (b), shaded regions show the min–max range of raw values over the trailing 20 steps, and thick curves show debiased exponential moving averages with weight $w = 0.95$. Panel (c) reports the macro-average EM over the seven evaluation benchmarks at logged checkpoints.

continuing from imperfect trajectories.

5.5 Training Latency Analysis

TREEPS-RAG is a training method; at inference time, it follows the same procedure as Search-R1 without invoking tree rollout. As a result, both methods demonstrate closely matched inference times. To compare their training latency, we measure the wall-clock time for rollout per step. As shown in Table 4, the runtime gap narrows as training progresses. The larger initial discrepancy arises because Search-R1’s untrained policy tends to generate short trajectories with a small number of steps, whereas TREEPS-RAG forces active exploration via tree construction.³ As training continues, Search-R1’s policy matures, producing longer trajectories. Under the matched Qwen3-8B setting, a complete TREEPS-RAG training run requires approximately 67 wall-clock hours, compared with 61 hours for both Search-R1 and GiGPO, corresponding to an overall training-time overhead of 9.8%. Appendix C.2 provides detailed training settings.

Stage	Search-R1	Ours	Diff. Rate
Global step 0	145.1	198.9	37.1%
Global step 100	315.3	379.7	20.4%

Table 4: Per-step rollout time (seconds) averaged over 5 runs, with batch size of 512.

5.6 Training Dynamics

Figure 3 compares the training dynamics of TREEPS-RAG and Search-R1-GRPO under the

³The proposed similarity-pruning costs < 1 second per four-step trajectory.

unified Qwen3-4B-Instruct-2507 setting. Both methods exhibit an overall increasing reward trend, while TREEPS-RAG achieves a higher late-stage mean reward and higher macro-average EM at every matched evaluation checkpoint. Its average KL loss in the final third of training is also lower (0.00863 vs. 0.01145). Appendix D.5 reports Qwen3-8B results, showing that these trends remain consistent at a larger model scale.

6 Conclusion

We present **TREEPS-RAG**, a tree-based RL framework for agentic RAG that enables step-wise credit assignment without intermediate annotations or auxiliary reward or value models. TREEPS-RAG builds a rollout tree and uses MC estimates from outcome-supervised rollouts to compute process advantages for intermediate steps, while retaining the simplicity of standard outcome-based optimization. Extensive experiments on seven QA benchmarks across various model series and scales show consistent improvements over strong baselines.

Limitations

Despite the promising results, our work has several limitations that need to be addressed in future work. First, due to limited computational resources, we primarily conduct experiments on small- to medium-scale backbone models. Training RL-based methods requires repeated policy rollouts and optimization steps, which makes large-scale training resource-intensive. While our approach is model-agnostic, scaling it to much larger language models would require additional computational resources and further engineering optimization.

tion. Second, our evaluation is limited to English, text-only QA benchmarks. Extending our approach to multilingual or multimodal settings remains an important direction for future work. Finally, while tree-based rollouts introduce additional coordination complexity compared to fully independent sampling, we mitigate this through an efficient on-line tree construction strategy that operates under a comparable rollout budget with standard outcome-supervised approaches. In addition, we implement asynchronous rollout and sampling during RL training, which significantly improves throughput and keeps the overall training cost within a manageable range. Further reducing latency and improving system efficiency is left for future work.

Ethical Considerations

While our proposed RL framework for agentic RAG demonstrates strong performance, the deployment of such a system requires careful attention to content safety. Since the system’s outputs are conditioned on external data, the potential presence of offensive or biased material can lead to unintended harmful generation. We strongly advise users to mitigate potential risks by ensuring that all performance assessments utilize curated, non-toxic benchmarks.

Acknowledgments

This study was supported by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd., a CUHK-led InnoCentre under the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government.

References

- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. [Self-rag: Learning to retrieve, generate, and critique through self-reflection](#). *Preprint*, arXiv:2310.11511.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). *Preprint*, arXiv:2005.14165.
- Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. 2025a. [Retaining by doing: The role of on-policy data in mitigating forgetting](#). *Preprint*, arXiv:2510.18874.
- Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025b. [Research: Learning to reason with search for llms via reinforcement learning](#). *Preprint*, arXiv:2503.19470.
- DeepSeek-AI. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Wenqi Fan, Yajuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. [A survey on rag meeting llms: Towards retrieval-augmented large language models](#). *Preprint*, arXiv:2405.06211.
- Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. [Group-in-group policy optimization for LLM agent training](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. [Retrieval-augmented generation for large language models: A survey](#). *Preprint*, arXiv:2312.10997.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. [Critic: Large language models can self-correct with tool-interactive critiquing](#). *Preprint*, arXiv:2305.11738.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. [Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. 2025. [Treerl: Llm reinforcement learning with on-policy tree search](#). *Preprint*, arXiv:2506.11902.
- Bingning Huang, Tu Nguyen, and Matthieu Zimmer. 2025a. [Tree-opo: Off-policy monte carlo tree-guided advantage optimization for multistep reasoning](#). *Preprint*, arXiv:2509.09284.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025b. [A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions](#). *ACM Transactions on Information Systems*, 43(2):1–55.
- Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. [Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity](#). In *Proceedings of*

- the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.
- Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. [Active retrieval augmented generation](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. [Search-r1: Training llms to reason and leverage search engines with reinforcement learning](#). Preprint, arXiv:2503.09516.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. [Billion-scale similarity search with gpus](#). *IEEE Transactions on Big Data*, 7(3):535–547.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. [TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense passage retrieval for open-domain question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. [Retrieval-augmented generation for knowledge-intensive nlp tasks](#). Preprint, arXiv:2005.11401.
- Kun Li, Zenan Xu, Junan Li, Zengrui Jin, Jinghao Deng, Zexuan Qiu, and Bo Zhou. 2026. [Discovery and reinforcement of tool-integrated reasoning chains via rollout trees](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14265–14283, San Diego, California, United States. Association for Computational Linguistics.
- Yangning Li, Weizhi Zhang, Yuyao Yang, Wei-Chieh Huang, Yaozu Wu, Junyu Luo, Yuanchen Bei, Henry Peng Zou, Xiao Luo, Yusheng Zhao, Chunkit Chan, Yankai Chen, Zhongfen Deng, Yinghui Li, Haitao Zheng, Dongyuan Li, Renhe Jiang, Ming Zhang, Yangqiu Song, and Philip S. Yu. 2025a. [A survey of RAG-reasoning systems in large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12120–12145, Suzhou, China. Association for Computational Linguistics.
- Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, Zheng Zhang, Wei Shen, Qian Liu, Chenghua Lin, Jian Yang, Ge Zhang, and Wenhao Huang. 2025b. [Treepo: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling](#). Preprint, arXiv:2508.17445.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Kun Luo, Hongjin Qian, Zheng Liu, Ziyi Xia, Shitao Xiao, Siqi Bao, Jun Zhao, and Kang Liu. 2025. [Infollow: Reinforcing search agent via reward density optimization](#). Preprint, arXiv:2510.26575.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. [Self-refine: Iterative refinement with self-feedback](#). Preprint, arXiv:2303.17651.
- 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)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). Preprint, arXiv:2203.02155.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. [Measuring and narrowing the compositionality gap in language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn.

2024. [Direct preference optimization: Your language model is secretly a reward model](#). *Preprint*, arXiv:2305.18290.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *Preprint*, arXiv:1707.06347.
- Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. [Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy](#). *Preprint*, arXiv:2305.15294.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *Preprint*, arXiv:2402.03300.
- Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. 2025. [RL’s razor: Why online reinforcement learning forgets less](#). *Preprint*, arXiv:2509.04259.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. [Hybridflow: A flexible and efficient rlhf framework](#). In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys ’25, page 1279–1297, New York, NY, USA. Association for Computing Machinery.
- Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V. Vasilakos. 2026. [Agentic retrieval-augmented generation: A survey on agentic rag](#). *Preprint*, arXiv:2501.09136.
- Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. [R1-searcher: Incentivizing the search capability in llms via reinforcement learning](#). *Preprint*, arXiv:2503.05592.
- Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2025. [Zerosearch: Incentivize the search capability of llms without searching](#). *Preprint*, arXiv:2505.04588.
- Hieu Tran, Zonghai Yao, and Hong Yu. 2025. [Exploiting tree structure for credit assignment in rl training of llms](#). *Preprint*, arXiv:2509.18314.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. [MuSiQue: Multi-hop questions via single-hop question composition](#). *Transactions of the Association for Computational Linguistics*, 10:539–554.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. [Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions](#). *Preprint*, arXiv:2212.10509.
- Hongru Wang, Boyang Xue, Baohang Zhou, Tianhua Zhang, Cunxiang Wang, Huimin Wang, Guanhua Chen, and Kam-Fai Wong. 2025. [Self-DC: When to reason and when to act? self divide-and-conquer for compositional unknown questions](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6510–6525, Albuquerque, New Mexico. Association for Computational Linguistics.
- Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. [Text embeddings by weakly-supervised contrastive pre-training](#). *Preprint*, arXiv:2212.03533.
- Yuxi Xie, Anirudh Goyal, Wenye Zheng, Min-Yen Kan, Timothy P. Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. [Monte carlo tree search boosts reasoning via iterative preference learning](#). *Preprint*, arXiv:2405.00451.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Qwen: An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025b. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. 2025c. [Treerpo: Tree relative policy optimization](#). *Preprint*, arXiv:2506.05183.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William 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*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). *Preprint*, arXiv:2210.03629.
- Tian Yu, Shaolei Zhang, and Yang Feng. 2024. [Autrag: Autonomous retrieval-augmented generation for large language models](#). *Preprint*, arXiv:2411.19443.
- Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. 2025a. [Process vs.](#)

outcome reward: Which is better for agentic RAG reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Chen Xu, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2025b. *Siren’s song in the ai ocean: A survey on hallucination in large language models*. *Preprint*, arXiv:2309.01219.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. *Sglang: Efficient execution of structured language model programs*. *Preprint*, arXiv:2312.07104.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. 2025. *StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization*. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21805–21830, Suzhou, China. Association for Computational Linguistics.

A Prompt Template

Figure 4 shows the input prompt template used during both training and inference. Our template largely follows that of Search-R1 (Jin et al., 2025), with a minimal modification to accommodate Qwen3 models. Specifically, we replace the `<think>` token used in Search-R1 with `<reason>`. This change is motivated by differences in tokenizer across Qwen model families. In the Qwen3 series, `<think>` is treated as a special token associated with the model’s thinking mode. Qwen3-4B-Instruct-2507 is not explicitly trained to use `<think>` as a control signal, while Qwen3-8B uses it to support its pretrained native thinking interface. To ensure consistent behavior across model variants and to avoid entangling our agentic RAG reasoning traces with model-specific thinking modes, we adopt `<reason>` as a replacement.

B Training Objective

Our training objective remains fully compatible with standard policy gradient methods, differing only in that our dense, process-level advantages defined in §3.3 are integrated.

$$\mathcal{J}(\theta) = \mathbb{E}_{x, \{y_i\}_{i=1}^N \sim \pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{\sum_{i=1}^N |y_i|} \sum_{i=1}^N \sum_{t=1}^{|y_i|} \min \left(\frac{\pi_{\theta}(y_{i,t} | x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} | x, y_{i,<t})} A_{i,t}, \text{clip} \left(\frac{\pi_{\theta}(y_{i,t} | x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} | x, y_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) A_{i,t} \right) - \beta D_{\text{KL}}[\pi_{\theta} \| \pi_{\text{ref}}] \right] \quad (3)$$

C Implementation Details

C.1 Datasets

Table 5 shows the data statistics. The 12K training set is used for all our implemented online RL approaches, which include Ours, Search-R1 (GRPO & PPO), and GiGPO.

Dataset	Train Set Size	Test Set Size
HotpotQA	10,000	7,405
2WikimultihopQA	-	12,576
MuSiQue	-	2,417
Bamboogle	-	125
TriviaQA	-	11,313
PopQA	-	14,267
NQ	2,000	3,610
Total	12,000	51,713

Table 5: Number of examples in our training and evaluation sets.

C.2 Training Details

The experiments in this paper are conducted using VeRL (Sheng et al., 2025), an open-source library for LLM reinforcement learning. Rollouts are generated with the SGLang (Zheng et al., 2024) inference engine in an asynchronous setting to accelerate training. The training runs on 8 NVIDIA H20 GPUs. For one complete Qwen3-8B training run under identical settings—12K training instances, rollout group size $N = 8$, batch size 512, and 10 epochs—TREEPS-RAG requires approximately 67 wall-clock hours (536 GPU-hours), whereas Search-R1 and GiGPO each require approximately 61 hours (488 GPU-hours). The hyperparameters for the training TREEPS-RAG are listed in Table 6. The tree depth limit D , i.e., maximum action steps, is a configurable hyperparameter. During

Input Template
<pre> < im_start >user Answer the given question. You must conduct reasoning inside <reason> and </reason> first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by <search> query </search> and it will return the top searched results between <information> and </information>. You can search as many times as your want. If you find no further external knowledge needed, you can directly provide the answer inside <answer> and </answer>, without detailed illustrations. For example, <answer> Beijing </answer>. Question: {input_question} < im_end > < im_start >assistant </pre>

Figure 4: Template for TREEPS-RAG. {input_question} will be replaced with specific question during training and inference. We follow Jin et al. (2025) and only replace <think> with <reason> to adapt for Qwen3 models.

training, we set $D = 4$ to follow the experimental configuration of our primary baseline, Search-R1 (Jin et al., 2025). Adopting this identical setting ensures a fair comparison, demonstrating that our performance gains stem from the proposed tree-based process supervision rather than extended rollout horizons. During inference, we set the maximum number of action steps as 10 for our approach and all reproduced approaches (Search-R1-GRPO/PPO and GiGPO). Appendix D.4 provides a detailed analysis of inference trajectory lengths and their associated performance.

Hyperparameters	Value
Training batch size	512
Optimizer	AdamW (Loshchilov and Hutter, 2019)
Learning rate	1e-6
Warmup ratio	0.285
Gradient accumulation step	1
Learning rate scheduler	Linear
KL coefficient	0.001
Temperature	1.0
Maximum length (per turn)	512
Total epoch	10
Rollout group size	8
Maximum step	4

Table 6: The hyperparameters used in our RL training.

C.3 Retrieval

We follow the same setting in Search-R1 (Jin et al., 2025), with **2018 Wikipedia** dump (Karpukhin et al., 2020) as the corpus source and E5 (Wang et al., 2024) as the retriever. The retriever is implemented with Faiss (Johnson et al., 2021) to accelerate the retrieval process. The Top-3 passages are returned, constrained to a maximum combined length of 512 tokens.

C.4 Baselines

We assess TREEPS-RAG across multiple backbone language models, including Qwen2.5-3B/7B-Instruct, Qwen3-4B-Instruct-2507, and Qwen3-8B (non-thinking mode). For methods that rely on **online** rollout-based reinforcement learning, including Ours, Search-R1 and GiGPO, we train all models under a unified experimental setting to ensure fair comparison with Qwen3 series. For this setting, we randomly sample 12K training instances from the NQ and HotpotQA training sets, and fix the rollout group size to $N = 8$. NQ and HotpotQA are then used for in-domain evaluation, with the remaining five serving as an out-of-domain test. We cite the results of Search-R1 and GiGPO with Qwen2.5 series from the original source works, with rollout group size $N = 5$ and on the full combination of HotpotQA and NQ training split, i.e., 170K. StepSearch and ReasonRAG rely on specifically-constructed process-supervised datasets, which constitute a core part of their methodological contributions. ReasonRAG generates training **offline** trajectories over 5K questions sampled from PopQA, HotpotQA, and 2WikiMultihopQA. StepSearch augments the MuSiQue dataset with intermediate annotations pre-generated by GPT-4o as labels for process rewards. Due to the differences, it is non-trivial to reproduce them under a fully unified setting. We therefore mainly report their original results. Below we detail two specially reproduced baselines results indicated with * symbol in Table 1.

When reproducing GiGPO using the Qwen2.5 series, we observed an inconsistency in the reported Bamboole evaluation. Specifically, the reported

Algorithm 1 TREEPS-RAG: Online Tree Construction and Process Advantage Calculation

Require: Question q , ground-truth answer a_{gold} , rollout budget N , depth limit D , local retention budget N_{retain} , large language model π_θ , retriever $\mathcal{R}$, passage number K .

Ensure: A pruned tree $\mathcal{T}$ where each node s_i is annotated with a process advantage $A(s_i)$.

▷ *Phase 1: Online Tree Construction*

```
1: Initialize tree  $\mathcal{T}$  with root node  $s_{root}(q)$ 
2:  $\mathcal{M}(0) \leftarrow \{s_{root}\}$  ▷ Initialize the set of nodes to be expanded at depth 0.
3: for  $d = 1$  to  $D$  do
4:    $\mathcal{M}(d) \leftarrow \emptyset$  ▷ Initialize the set of retained nodes for the current depth
5:    $B_d \leftarrow \lceil N/|\mathcal{M}(d-1)| \rceil$  ▷ Update branching factor for expansion
6:   for each parent node  $s_p \in \mathcal{M}(d-1)$  do
7:      $C(s_p) \leftarrow \{s_p^1, \dots, s_p^{B_d}\} \sim \pi_\theta$  ▷ Sample  $B_d$  children from parent  $s_p$ 
8:      $\mathcal{C}_{search}(s_p) \leftarrow \{s_p^i \in C(s_p) \mid a_{s_p^i} = \text{search}\}$  ▷ Identify searchable children
9:      $\mathcal{C}_{answer}(s_p) \leftarrow \{s_p^i \in C(s_p) \mid a_{s_p^i} = \text{answer}\}$  ▷ Identify terminal answer children
10:     $\hat{N}_{retain}(s_p) \leftarrow \min(N_{retain}, |\mathcal{C}_{search}(s_p)|)$ 
11:    for each search child  $s_p^i \in \mathcal{C}_{search}(s_p)$  do
12:       $P_i \leftarrow \mathcal{R}(q_i, K)$  ▷ Retrieve passages for similarity calculation
13:    end for
14:     $\mathcal{C}_{retain}(s_p) \leftarrow \mathcal{F}_{prune}(\mathcal{C}_{search}(s_p), \hat{N}_{retain}(s_p))$  ▷ Similarity-based clustering
15:     $\mathcal{M}(d) \leftarrow \mathcal{M}(d) \cup \mathcal{C}_{retain}(s_p)$  ▷ Add locally retained nodes to the global set
16:     $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{C}_{retain}(s_p) \cup \mathcal{C}_{answer}(s_p)$  ▷ Add retained search and answer children to the tree
17:  end for
18: end for
```

▷ *Phase 2: Process Advantage Calculation*

```
19: for each node  $s_i \in \mathcal{T}$  do
20:   if  $a_{s_i} = \text{answer}$  OR  $\text{depth}(s_i) = D$  then
21:      $r(s_i) \leftarrow \text{EM}(a_{pred}, a_{gold})$  ▷ Assign outcome rewards to leaves
22:   end if
23: end for
24: for each node  $s_i \in \mathcal{T}$  do
25:    $V(s_i) \leftarrow \frac{1}{|\mathcal{L}(s_i)|} \sum_{s_j \in \mathcal{L}(s_i)} r(s_j)$  ▷ Calculate node values
26:    $A(s_i) \leftarrow \frac{1}{\sqrt{|\mathcal{L}(s_i)|}} [A_{global}(s_i) + A_{local}(s_i)]$  ▷ Calculate process advantages
27: end for
```

score does not appear to be normalized by the benchmark size (125), which may be related to the batch-based evaluation procedure. To ensure a fair and consistent comparison, we therefore report our own reproduced results on Bamboogle. For the remaining benchmarks, our reproduced results closely match those reported in the original paper, and we thus directly cite their original scores.

ReasonRAG is trained with a different data source, augmenting the 2018 Wikipedia dump with passages from PopQA, HotpotQA, and 2WikiMultihopQA, whereas our main experiments use 2018 Wikipedia dump only (Appendix C.3). Since the offline preference training trajectories are generated

using BGE⁴ as the retriever with top-3 retrieval, we follow the same retriever during reproduction to avoid discrepancies between training and inference. Using their MCTS-constructed dataset, we are able to reproduce performance comparable to the reported results on Qwen2.5-7B-Instruct. However, when applying the same pipeline to the Qwen3 series, the reproduced performance is less competitive, particularly on PopQA. We therefore report the best results for each dataset obtained after DPO training, based on multiple hyperparameter trials, under both the BGE-indexed augmented corpus and the BGE-indexed 2018 Wikipedia dump. Further analysis suggests that this difference may stem

⁴<https://huggingface.co/BAAI/bge-base-en-v1.5>

Iteration-0 Prompt Used in ReasonRAG

System: You are an assistant for question answering with access to a retrieval tool. Upon receiving a question, your task is to:

- Analyze and Decompose the Question: Break the question into smaller, manageable sub-questions to ensure all aspects are addressed.
- Evaluate Your Knowledge: Assess each sub-question or component:
 - Identify parts you can confidently answer based on your existing knowledge.
 - Pinpoint parts that require additional information or verification through retrieval tools.
- Conciseness: Ensure both queries and answers are concise, using nouns or short phrases whenever possible.
- Respond Format:
 - If your knowledge is sufficient to answer the question, conclude with:
“So the answer is <answer>answer</answer>”
 - If retrieval is necessary to provide a complete answer, conclude with:
“So the next query is <query>query</query>”

User: {input_question}

Table 7: Iteration-0 prompt used in ReasonRAG, which allows the model to answer using its parametric knowledge if deemed sufficient and confident, or to invoke retrieval otherwise.

from behavioral differences between the backbone models. Under the iteration-0 prompt designed in ReasonRAG (Table 7), Qwen2.5-7B-Instruct tends to actively generate search queries and rely on retrieved passages, with only around 3% of PopQA instances attempting to answer directly using parametric knowledge. In contrast, Qwen3-8B (no thinking mode) answers nearly 80% of questions directly in the first iteration without invoking retrieval. While DPO training partially mitigates this tendency, it does not fully shift the model’s behavior given the limited scale of the training data (5K questions produced 13K process steps).

D Supplementary Experiments

D.1 Controlled Base-Model Comparison

To verify that the gains of TREEPS-RAG do not depend on instruction-tuned backbones or mismatched baseline settings, we conduct an additional controlled experiment using Qwen2.5-3B Base. All methods are trained on the same 12K sampled training instances, use the same rollout group size $N = 8$, share the same optimization configuration, and are evaluated with the same protocol.

Table 8 shows that TREEPS-RAG consistently outperforms both Search-R1-GRPO and Search-R1-PPO under the identical base-model setting. This confirms that the improvement is not specific to instruction-tuned models and can be attributed to the proposed tree-based process supervision.

D.2 Ablation on Normalization Exponent

In §3.3, we normalize the process advantage by the size of the descendant leaf set, using $|\mathcal{L}(s)|^{-1/2}$ as the default scaling factor. This normalization is intended to reduce the dominance of high-level non-leaf nodes, which may appear in multiple root-to-leaf trajectories and therefore be repeatedly optimized. To evaluate the sensitivity to this design choice, we vary the exponent q in the generalized normalization term $|\mathcal{L}(s)|^{-q}$ while keeping all other settings unchanged.

As shown in Table 9, removing the normalization term leads to a clear degradation in average performance, confirming that subtree-size normalization is important for stable process supervision. The default exponent $q = 0.5$ achieves the best average result. Although other exponents can improve individual benchmarks, they do not outperform the default setting overall, suggesting that square-root normalization provides a robust balance between preserving useful process signals and mitigating repeated optimization of shared prefix nodes.

D.3 Alternative Similarity and Clustering Strategies

We further evaluate whether the effectiveness of similarity-based pruning depends on a particular similarity metric or clustering algorithm. We compare the default Jaccard similarity over retrieved passage IDs with query-embedding and document-embedding similarities (using E5-base-v2⁵), and compare hierarchical clustering with a K-Means variant.

All variants remain competitive, indicating that the gain of TREEPS-RAG does not hinge on a single similarity metric or clustering choice. The default Jaccard similarity with hierarchical clustering achieves the best average performance while keeping the pruning procedure non-iterative, lightweight, and deterministic for online rollout construction.

⁵<https://huggingface.co/intfloat/e5-base-v2>

Method	HotpotQA	2Wiki	MuSiQue	Bamboogle	TriviaQA	PopQA	NQ	Avg
Ours	38.02	36.02	13.07	32.00	61.89	43.51	43.85	38.34
Search-R1-GRPO	31.49	30.32	7.32	16.80	60.08	42.53	42.60	33.02
Search-R1-PPO	29.18	27.33	5.50	11.20	58.55	42.72	39.81	30.61

Table 8: Controlled comparison on Qwen2.5-3B Base. Scores are EM percentages.

Exponent q	HotpotQA	2Wiki	MuSiQue	Bamboogle	TriviaQA	PopQA	NQ	Avg
0.5 (Ours)	48.00	54.10	23.30	53.60	68.00	48.80	47.60	49.06
0 (No norm)	47.27	52.71	22.71	48.80	67.48	47.62	47.48	47.72
0.75	48.02	50.40	23.71	53.24	67.43	48.10	47.42	48.33
1.5	47.35	54.40	22.67	52.78	67.91	47.85	47.06	48.57

Table 9: Ablation on the normalization exponent q in $|\mathcal{L}(s)|^{-q}$ with Qwen3-4B-Instruct-2507. Scores are EM percentages.

Similarity Metric + Clustering	HotpotQA	2Wiki	MuSiQue	Bamboogle	TriviaQA	PopQA	NQ	Avg
Jaccard + Hierarchical	48.00	54.10	23.30	53.60	68.00	48.80	47.60	49.06
Jaccard + K-Means	47.31	52.98	22.30	50.40	67.92	48.17	46.90	48.00
Query Embedding + Hierarchical	48.35	52.32	23.71	53.60	67.44	47.51	47.31	48.61
Document Embedding + Hierarchical	48.14	53.29	22.42	52.80	67.37	48.18	48.89	48.73

Table 10: Alternative similarity metrics and clustering strategies on Qwen3-4B-Instruct-2507. Scores are EM percentages.

D.4 Inference Trajectory Length and Performance

To examine inference behavior across different trajectory lengths, particularly when inference extends beyond the training depth $D = 4$, we analyze the trajectories generated by TREEPS-RAG with Qwen3-4B-Instruct-2507 over all 51,713 evaluation instances. During inference, the maximum action step is set to 10, where each step corresponds to either a search or an answer action.

Steps	Count	Percentage	EM
3	3,990	7.72%	0.7699
4	46,763	90.43%	0.5087
5–8	957	1.85%	0.5152
No answer	3	0.006%	–

Table 11: Distribution of inference trajectory lengths for Qwen3-4B-Instruct-2507 over all 51,713 evaluation instances. A step denotes either a search or an answer action.

As summarized in Table 11, 51,710 trajectories successfully terminate with a final answer, while only three instances fail to terminate within the maximum limit. Among the successfully terminated trajectories, the maximum observed length is eight steps, corresponding to up to seven search actions before the final answer. Overall, 98.14% of the evaluation instances terminate within the four-step training horizon. For the 1.85% of instances whose

trajectories extend beyond the training horizon, the model achieves an EM of 0.5152, comparable to the EM of 0.5087 for four-step trajectories. These results suggest that when the model decides to search beyond its training horizon, it does so meaningfully to resolve harder queries. Thus, within the evaluated QA benchmarks, we do not observe lower aggregate accuracy when trajectories extend beyond the training depth.

D.5 Additional Training Dynamics

Figure 5 reports the corresponding training dynamics under the matched Qwen3-8B setting. Both methods exhibit steadily increasing training rewards, while TREEPS-RAG achieves a higher late-stage mean reward (0.602 vs. 0.584). It also obtains higher macro-average EM at matched evaluation checkpoints. Its mean raw KL loss over the complete training run is also lower (0.00447 vs. 0.00670). Together with the Qwen3-4B results in Figure 3, these results demonstrate a consistent pattern across model scales: stronger late-stage rewards and benchmark performance.

E Case Study

Case-Study-1 Figure 6 presents a case study for the question “Which post-DC Extended Universe actress will also play a role in what is intended to be the fifth installment of the DC Extended Universe?” Answering this question requires resolving

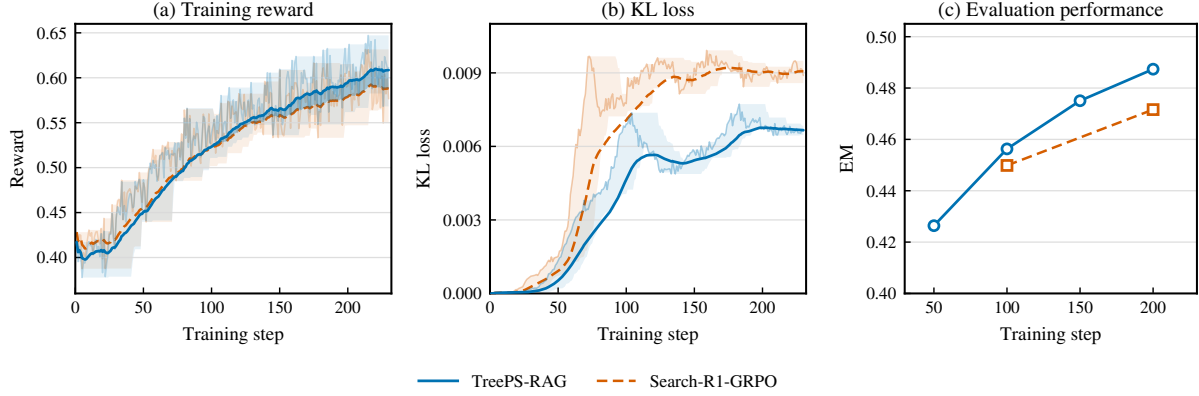


Figure 5: Training dynamics under the matched Qwen3-8B setting. The visualization conventions follow Figure 3.

three distinct pieces of information: (1) identifying which film constitutes the fifth installment of the DC Extended Universe; (2) determining which actresses appear in that installment; and (3) verifying that the actress has previously appeared in a DC film. Under the given reasoning prefix, the model has already established that *Justice League* is the fifth installment of the DCEU, but the retrieved passages contain both relevant evidence and misleading candidate mentions. When continuing from this identical prefix, **Search-R1 incorrectly grounds the answer to a spurious entity**, selecting *Margot Robbie* based on a loosely related passage that mentions her role in *Suicide Squad*, despite the absence of explicit evidence linking her to *Justice League*.

In contrast, **our approach correctly grounds the answer in the retrieved evidence by aligning entity mentions with all required constraints**. Specifically, it identifies that *Gal Gadot* previously starred in *Wonder Woman* and is explicitly listed as part of the cast of *Justice League*. By consistently grounding the final answer in passages that satisfy all three conditions, our method successfully produces the correct answer. This example highlights our model’s improved ability to **locate, verify, and ground answers from retrieved documents even under imperfect reasoning prefixes**, avoiding distracting entity associations that lead to incorrect answers.

Case-Study-2 Figure 7 illustrates another continuation-based case study for the question “Which film has more writers, *Splash* or *Christopher Robin*?” Answering this question requires correctly identifying and comparing the number of credited screenwriters for both films. Under the given reasoning prefix, the model has already established that *Splash* was written by two

screenwriters: *Lowell Ganz and Babaloo Mandel*, but the subsequent search results for *Christopher Robin* fail to provide explicit screenwriter information. When continuing from this identical prefix, **Search-R1 prematurely commits to an answer despite insufficient evidence**, inferring the number of writers for *Christopher Robin* from loosely related information (e.g., composers and musicians) and terminating the trajectory with an incorrect answer. This behavior reflects a failure to verify missing critical information and a tendency to propagate uncertainty into the final decision. In contrast, **our approach explicitly recognizes the incompleteness of the retrieved evidence and initiates an additional verification step via query rewriting**. By reformulating the search query to directly target the screenwriter credits of *Christopher Robin (2018)*, our method successfully retrieves the missing information, confirming that the film was written by *Alex Ross Perry, Tom McCarthy, and Allison Schroeder*. Integrating this newly acquired evidence, our model correctly concludes that *Christopher Robin* has more writers than *Splash*. This example highlights our method’s ability to **reflect on intermediate reasoning states, detect unresolved informational gaps, and recover from incomplete trajectories**, rather than prematurely terminating with an answer.

Question: Which post DC Extended Universe actress will also play a role in what is intended to be the fifth installment of the DC Extended Universe? Gold Answer: Gal Gadot	
Trajectory Prefix (Search-R1) <reason> I need to find out which actress from the DC Extended Universe (DCEU) will also play a role in what is intended to be the fifth installment of the DCEU. This requires identifying both the actress and the specific film, which is meant to be the fifth movie in the DCEU series. I should first determine what the fifth installment of the DCEU is intended to be and then identify which actress from a previous DCEU film will reappear in it.</reason> <search> what is the fifth installment of the DC Extended Universe intended to be </search> <information> Doc 1(Title: "DC Extended Universe") Doc 2(Title: "DC Extended Universe") Doc 3(Title: "DC Extended Universe") </information> <reason> The search results do not clearly identify the fifth installment of the DCEU. The results mention various films like "Green Lantern Corps" and "Batman," but do not specify which one is intended to be the fifth. I need to narrow down which film is the fifth in the DCEU lineup.</reason> <search> which actress from DC Extended Universe will appear in the fifth DCEU film </search> <information> Doc 1(Title: "DC Extended Universe") selected Cathy Yan as director, with Margot Robbie, Sue Kroll, and Bryan Unkless serving as producers for the project. Robbie will reprise her role as Harleen Quinzel / Harley Quinn from ""Suicide Squad"". The film is expected to be the first theatrical R-rated installment in the franchise..... Doc 2(Title: "Wonder Woman (2017 film)") Wonder Woman (2017 film) Wonder Woman is a 2017 American superhero film based on the DC Comics character of the same name, produced by DC Entertainment in association with RatPac Entertainment and Chinese company Tencent Pictures, and distributed by Warner Bros. Pictures. It is the fourth installment in the DC Extended Universe (DCEU). Directed by Patty Jenkins from a screenplay by Allan Heinberg and a story by Heinberg, Zack Snyder, and Jason Fuchs, ""Wonder Woman"" stars Gal Gadot in the title role, alongside Chris Pine, ... Doc 3(Title: "Justice League (film)") Justice League (film) Justice League is a 2017 American superhero film based on the DC Comics superhero team of the same name, distributed by Warner Bros. Pictures. It is the follow-up to 2016's ""Wonder Woman"" and the fifth installment in the DC Extended Universe (DCEU). The film is directed by Zack Snyder </information> <reason> From the search results, I now know that "Justice League" is the fifth installment in the DC Extended Universe (DCEU). The question asks which DCEU actress will also play a role in this fifth installment. I need to identify which actress from a previous DCEU film will reappear in "Justice League."</reason> <search> which DCEU actress will reprise a role in Justice League </search> <information> Doc 1(Title: "Justice League (film)") Justice League (film) Justice League is a 2017 American superhero film based on the DC Comics superhero team of the same name, distributed by Warner Bros. Pictures. It is the follow-up to 2016's ""Wonder Woman"" and the fifth installment in the DC Extended Universe (DCEU). The film is directed by Zack Snyder, written by Chris Terrio and Joss Whedon, and features an ensemble cast that includes Ben Affleck, Henry Cavill, Gal Gadot, ... Doc 2 (Title: "Justice League (film)") after a reshuffling of film production staff at Warner Bros..... Justice League (film) Justice League is a 2017 American superhero film based on the DC Comics superhero team of the same name, distributed by Warner Bros. Pictures. It is the follow-up to 2016's ""Wonder Woman"" and the fifth installment in the DC Extended Universe (DCEU). The film is directed by Zack Snyder Doc 3(Title: "DC Extended Universe") (Title: "DC Extended Universe") selected Cathy Yan as director, with Margot Robbie, Sue Kroll, and Bryan Unkless serving as producers for the project. Robbie will reprise her role as Harleen Quinzel / Harley Quinn from ""Suicide Squad"". The film is expected to be the first theatrical R-rated installment in the </information>	
Search-R1 Continuation (wrong) <reason>The results confirm that "Justice League" is the fifth installment of the DCEU. The actress who will reprise a role in Justice League is Margot Robbie, who plays Harley Quinn from "Suicide Squad." The question asks which post-DCEU actress will also play a role in the fifth installment. However, "post-DCEU actress" may be a misstatement. Given the context, it likely means an actress from a DCEU film who will appear in the fifth installment. Margot Robbie is the correct actress from a previous DCEU film (Suicide Squad) who appears in Justice League.</reason> <answer>Margot Robbie</answer>	
Ours Continuation (correct) <reason> The information confirms that "Justice League" is the fifth installment in the DCEU. The cast includes Gal Gadot, who played Wonder Woman, and other actors. However, the question specifically asks for an actress from a previous DCEU film who will also play a role in the fifth installment. The actress Margot Robbie, who played Harley Quinn in "Suicide Squad," is noted to reprise her role in a film, though it's not explicitly stated to be "Justice League." However, the most prominent actress from a prior DCEU film who appears in the fifth installment is Gal Gadot, as she played Wonder Woman in "Wonder Woman," which is the fourth film in the DCEU, and she appears in "Justice League." Therefore, Gal Gadot is a key actress from a prior DCEU film who appears in the fifth installment. </reason> <answer> Gal Gadot </answer>	

Figure 6: Case study 1 where **TreePS-RAG** is correct while **Search-R1** fails, given the same reasoning prefix. Evidence supporting the correct answer is highlighted with **red bold underlines**, whereas the information related to Search-R1's erroneous answer is marked with **black bold underlines**.

Question: Which film has more writers, Splash or Christopher Robin?
Gold Answer: Christopher Robin

Trajectory Prefix (Search-R1)

<reason> I need to determine which film, "Splash" or "Christopher Robin," has more writers. This requires comparing the number of credited writers for each film. I should first identify the writers associated with each film. Since the question is about the number of writers, I need to find out how many individuals are credited as writers for each film. I will start by gathering information on the writers of "Splash" and then move on to "Christopher Robin". **</reason>**

<search> how many writers are credited for the film Splash **</search>**

<information> Doc 1(Title: "Splash (film)") **Splash (film) Splash is a 1984 American fantasy romantic comedy film directed by Ron Howard, written by Lowell Ganz and Babaloo Mandel,** and starring Tom Hanks, Daryl Hannah, John Candy, and Eugene Levy

Doc 2(Title: "Babaloo Mandel") including a set of triplets. He and long-time writing partner Lowell Ganz have penned numerous high-profile films, including ""Splash"" (1984), ""Parenthood"" (1989), ""City Slickers"" (1991) and ""A League of Their Own"" (1992).....

Doc 3(Title: "Splash (film)") LP and cassette have but has a considerably longer running length..... **</information>**

<reason> From the search results, I see that "Splash" (1984) was written by Lowell Ganz and Babaloo Mandel. This indicates two writers for "Splash". Now I need to find out how many writers are credited for "Christopher Robin". **</reason>**

<search> how many writers are credited for the film Christopher Robin **</search>**

<information> Doc 1(Title: "Christopher Robin (film)") Zanelli and Jon Brion. At an Academy event, songwriter and Disney Legend Richard M. Sherman revealed that the film would feature the iconic ""Winnie the Pooh"" theme, and that he was working on three new songs for the film, titled ""Goodbye Farewell"", ""Busy Doing Nothing"" and ""Christopher Robin"", with the first one being performed by the voice cast, and the last two by Sherman.....

Doc 2(Title: "Christopher Robin") 2018 Disney live-action film ""Christopher Robin"", which is about Christopher Robin as an adult meeting Pooh once again.....

Doc 3(Title: "Christopher Robin (film)") was replaced by Cummings, respectively. Principal photography on the film began in early August 2017, in the United Kingdom, and concluded on November 4, 2017..... **</information>**

Search-R1 Continuation (wrong)

<reason> The search results for "Christopher Robin" do not clearly state the number of writers. However, one result mentions "Zanelli and Jon Brion" in relation to the film. This suggests that at least two writers are credited for "Christopher Robin". Given that "Splash" has two writers (Lowell Ganz and Babaloo Mandel), and "Christopher Robin" also appears to have at least two writers (Zanelli and Jon Brion), the count is tied. However, I need to verify if there are more writers for one of the films. Since the results do not provide a complete list of writers for "Christopher Robin", and only mention two names, and "Splash" is explicitly stated to have two writers, I must conclude that both films have two writers unless additional information shows otherwise. **</reason>**

<answer>Splash**</answer>**

Ours Continuation (correct)

<reason> The search results indicate that "Christopher Robin" (2018) has a film score composed by Jóhann Jóhannsson, who was replaced by Geoff, and that the film features music by Jon Brion and Zanelli. However, the information about writers (screenwriters) is not clearly specified. The information provided focuses on composers and voice actors, not screenwriters. In contrast, "Splash" is explicitly stated to have two credited writers: Lowell Ganz and Babaloo Mandel. There is no clear mention of multiple screenwriters for "Christopher Robin". Therefore, based on available information, "Splash" has a known number of writers (two), while "Christopher Robin" lacks explicit screenwriter credits. Without definitive information on the number of screenwriters for "Christopher Robin", it is reasonable to infer that "Splash" has more clearly documented writers.**</reason>**

<search> who are the screenwriters of the film Christopher Robin (2018) **</search>**

<information> Doc 1(Title: "Christopher Robin (film)") **Christopher Robin (film) Christopher Robin is a 2018 American fantasy comedy-drama film directed by Marc Forster and written by Alex Ross Perry, Tom McCarthy, and Allison Schroeder,** from a story by Greg Brooker and Mark Steven Johnson.....

Doc 2(Title: "Christopher Robin (film)") reported that the studio had hired Marc Forster to direct the film, titled ""Christopher Robin,"" and the project would have ""strong elements of magical realism as it seeks to tell an emotional journey with heartwarming adventure."" On March 1, 2017, **Tom McCarthy was hired to rewrite the existing screenplay.** On April 26, 2017, Ewan McGregor was announced to play the title character while **Allison Schroeder was recruited to do additional work on the script.**

Doc 3(Title: "Christopher Robin (film)") was replaced by Cummings, respectively. Principal photography on the film began in early August 2017, in the United Kingdom, and concluded on November 4, 2017..... **</information>**

<reason> The most recent and specific information confirms that the film "Christopher Robin" (2018) was written by Alex Ross Perry, Tom McCarthy, and Allison Schroeder. This makes a total of three screenwriters. In contrast, "Splash" (1984) has two credited writers: Lowell Ganz and Babaloo Mandel. Therefore, "Christopher Robin" has more writers than "Splash".

</reason>

<answer> Christopher Robin **</answer>**

Figure 7: Case study 2 where TreePS-RAG is correct while Search-R1 fails to get needed information given the same reasoning prefix. Evidence supporting the correct answer is highlighted with **black bold underlines**.