ARBOR: ONLINE PROCESS REWARDS VIA A REUSABLE RUBRIC BUFFER FOR SEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based search agents are trained predominantly with outcome-only reward, leaving the search process itself unsupervised. This signal degenerates on outcome-homogeneous groups where all sampled trajectories share the same correctness, yielding zero within-group advantage and no gradient. Existing process supervision either trains a costly scoring model or generates per-query rubrics that are inconsistent across queries. We propose **ARBOR** (**A**daptive **R**ubric **B**uffer for **O**nline **R**eward), a reusable process-reward framework that maintains a rubric memory shared across queries. Query-local drafts induced from contrastive trajectories are admitted, consolidated into cross-query common rubrics, and retired as the policy evolves. A small active subset of common rubrics scores trajectories via sparse pairwise judging, and the resulting scores are added to the base reward, providing process-level gradient even when outcome reward is uniform. ARBOR consistently outperforms GRPO and DAPO baselines on four multi-hop QA benchmarks, raising average LLM-judge accuracy by up to **4.2** points and converting up to **42%** of otherwise-zero-gradient training groups into informative ones.

1 INTRODUCTION

LLM-based agents that interact with external environments through reasoning-and-acting paradigms such as ReAct (Yao et al., 2023) are widely used for complex tasks. Search agents are a representative case, answering multi-hop questions by iteratively reformulating queries, retrieving and filtering external evidence, and synthesizing a final answer (Press et al., 2023; Xi et al., 2025). This interaction substantially improves multi-hop QA over direct answering. Recent systems such as Search-R1 (Jin et al., 2025) and R1-Searcher (Song et al., 2025), together with other studies (Li et al., 2025; Jiang et al., 2025), show that RL has become the dominant training paradigm for search agents. Yet RL training still relies almost exclusively on outcome-only reward, typically final-answer correctness with a format penalty (Shao et al., 2024; Yu et al., 2025), leaving the search process unsupervised.

Trajectories sampled from the same query can follow very different search paths even when they arrive at the same outcome, as illustrated in Figure 1: one may reason carefully through targeted retrieval while another stumbles onto the answer after redundant detours, but final-answer correctness assigns them identical reward. Under group-relative objectives such as GRPO (Shao et al., 2024), identical within-group reward produces zero relative advantage and no policy gradient, so process differences that could inform better search behavior contribute nothing to training. Such outcome-homogeneous groups are far from rare in search-agent RL training (see Section 4.3), making them a major bottleneck for outcome-only reward.

Adding process-level supervision is a natural response, yet existing routes do not fit search agents cleanly. Building a dedicated process reward model (PRM) typically involves preparing process-supervision data and training a separate scoring model (Lightman et al., 2024; Wang et al., 2024; Luo et al., 2024). For search agents, obtaining such supervision through intermediate-state rollouts can further incur substantial search-tool costs. Query-specific LLM-generated rubrics, exemplified by Rubrics-as-Rewards (Gunjal et al., 2026), sidestep scoring-model training but remain tied to individual queries rather than forming shared process criteria across queries. Meanwhile, these criteria are generated in advance and remain fixed during policy training.

These limitations point to three properties that a process reward suited to search-agent RL should possess. First, it should supervise the search process itself, supplying learning signal for the process

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

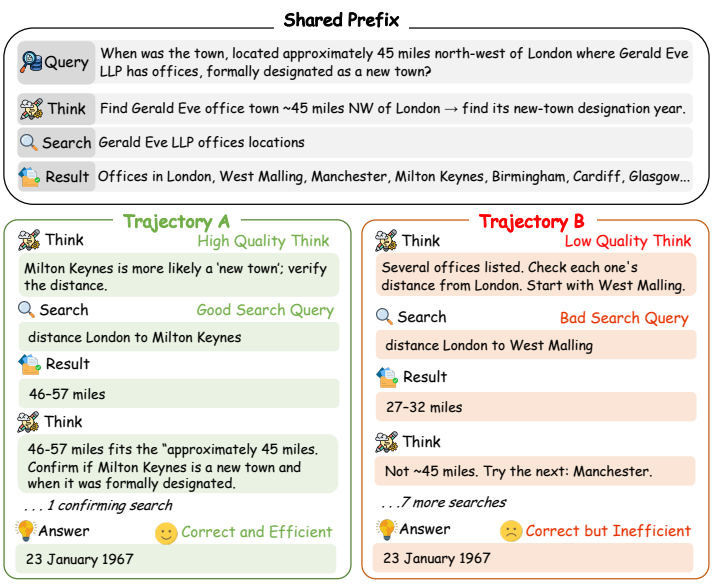


Figure 1: **Process quality divergence under identical outcomes.** Two trajectories from the same query reach the same answer yet differ markedly in search efficiency.

quality that outcome-only reward overlooks; this property matters most on outcome-homogeneous groups, where process supervision is the only available within-group signal. Second, the process criteria should be general, cross-query reusable standards, rather than query-specific rubrics that may conflict across queries. Third, the effectiveness of process criteria decays as the policy's behavior distribution evolves, so the criteria themselves must be updated continuously rather than fixed once.

We propose **ARBOR** (**A**daptive **R**ubric **B**uffer for **O**nline **R**eward), a reusable process-reward framework for search-agent RL training. The core component is a rubric memory consisting of a candidate pool, which holds query-local drafts induced from contrastive trajectories within a query-group, and a common pool, which stores rubrics consolidated into reusable cross-query process criteria. An online lifecycle of admission, consolidation, and retirement consolidates candidate drafts into common rubrics and retires stale ones, so that the common pool provides a unified criterion across queries and evolves with the policy's behavior distribution. Reward shaping invokes only a small active subset of common rubrics. Trajectories within a query-group are scored pairwise under each active rubric, and the resulting scores are added to the base reward. The reusable common pool remains effective even on outcome-homogeneous groups, where it still yields within-group process discrimination that outcome-only reward cannot. Figure 2 shows the overall framework.

Our contributions are as follows: (1) we propose ARBOR, a reusable process-reward framework that provides within-group process supervision even when outcome-only reward yields zero gradient; (2) we design a rubric memory with an online admission, consolidation, and retirement lifecycle that maintains consistent cross-query process criteria and evolves with the policy; and (3) ARBOR consistently outperforms GRPO and DAPO across three Qwen3 scales on four multi-hop QA benchmarks, improving average LLM-judge accuracy by up to **4.2** points and converting up to **42%** of outcome-homogeneous groups into ones with nonzero reward variance.

2 Related Work

2.1 RL Reward Design for Search Agents

Reward design for RL-trained search agents has evolved along three lines. The dominant paradigm relies on outcome-only reward, using final-answer correctness or F1 plus format penalties, and leaves the search process entirely unsupervised. Search-R1 (Jin et al., 2025), R1-Searcher (Song et al., 2025), and Search-o1 (Li et al., 2025) are representative of this approach. A second line augments outcome reward with task-specific process heuristics such as information gain, path coverage, or

retrieval cost, as in StepSearch (Zheng et al., 2025), Search-P1 (Xia et al., 2026), SIGHT (Zhong et al., 2026), InfoFlow (Luo et al., 2025), and TIPS (Xie et al., 2026b), but these metrics cannot capture qualitative aspects of search strategy. A third line trains a process reward model (PRM) to provide step-level feedback, with supervision constructed in different ways: PRM800K (Lightman et al., 2024) uses human annotation, Math-Shepherd (Wang et al., 2024) and OmegaPRM (Luo et al., 2024) use rollout-value estimation, and PRIME (Cui et al., 2025) infers from outcomes. In the search-agent setting, representative attempts include PPR (Xu et al., 2025) with pre-defined principles and a category-aware PRM, ReasonRAG (Zhang et al., 2025b) with MCTS-constructed step-level annotations followed by process-supervised DPO, and LeTS (Zhang et al., 2025a) with a mixture of stepwise process reward and outcome reward.

Existing work either ignores the quality of the search process entirely, reduces it to quantifiable but semantically shallow domain metrics, or relies on a separately trained verifier. Against this background, ARBOR provides process-level feedback that discriminates within outcome-homogeneous groups, restoring a learning signal where outcome-only reward fails.

2.2 Rubric-Based Reward Signals

While rubric-conditioned evaluators such as Prometheus (Kim et al., 2024) and LLM-Rubric (Hashemi et al., 2024) use predefined rubrics to structure LLM-as-a-judge assessment, recent work applies rubrics directly as RL reward signals. Most methods generate rubrics per query without cross-query sharing, which may cause conflicts across queries. Rubrics-as-Rewards (Gunjal et al., 2026) uses static query-specific checklists as on-policy rewards; similar per-query approaches include (Wang et al., 2025; He et al., 2025; Zhou et al., 2026). Several systems further evolve rubrics or their generators during training (Xu et al., 2026; Sheng et al., 2026; Shao et al., 2026), addressing staleness but still without sharing across queries. Auto-Rubric (Xie et al., 2026a) and OpenRS (Jia et al., 2026) achieve cross-query reusability through general rubric sets, but these are learned offline and do not adapt during RL training.

ARBOR combines cross-query reusability with online adaptation: a persistent rubric buffer consolidates query-local drafts into shared common rubrics and continuously retires outdated ones, keeping process standards consistent across queries and aligned with the evolving policy.

3 Method

3.1 Overview and Problem Setup

We consider RL training for search agents. Given a query q, the policy produces a trajectory τ through multiple rounds of think, search, and observe interactions and finally outputs an answer. At each training step, we sample K trajectories from the same query to form a query-group $\mathcal{G}_q = \{\tau_1, \ldots, \tau_K\}$, where trajectories share the query and environment but differ through policy sampling.

During training, ARBOR maintains a single rubric memory $\mathcal{M}$ shared across all queries. At every step, ARBOR uses a small number of currently active natural-language process rubrics from $\mathcal{M}$ to score the trajectories in $\mathcal{G}_q$ along process dimensions, and the resulting score is additively combined with the existing RL reward as an auxiliary process-level signal for the policy optimizer. $\mathcal{M}$ itself evolves throughout training under an online admission, consolidation, and retirement lifecycle, with its contents updated in step with the policy behavior distribution.

3.2 Contrastive Local Rubric Induction

ARBOR induces query-local draft rubrics from each query-group $\mathcal{G}_q$ through contrastive induction. Let $F_1(\tau)$ denote the token-level F1 against the gold answer, and let $\mathcal{G}_q^-$ contain trajectories scoring strictly below τ^+. We select the best, worst, and hard-negative anchors as:

$$\tau^+ = \arg\max_{\tau \in \mathcal{G}_q} F_1(\tau), \qquad \tau^-_{\text{worst}} = \arg\min_{\tau \in \mathcal{G}_q} F_1(\tau), \qquad \tau^-_{\text{hard}} = \arg\max_{\tau \in \mathcal{G}_q^-} F_1(\tau). \tag{1}$$

These anchors define two complementary contrasts. The pair $(\tau^+, \tau^-_{\text{worst}})$ exposes large-scale success-failure differences and surfaces critical process deviations, whereas $(\tau^+, \tau^-_{\text{hard}})$ provides a finer-grained

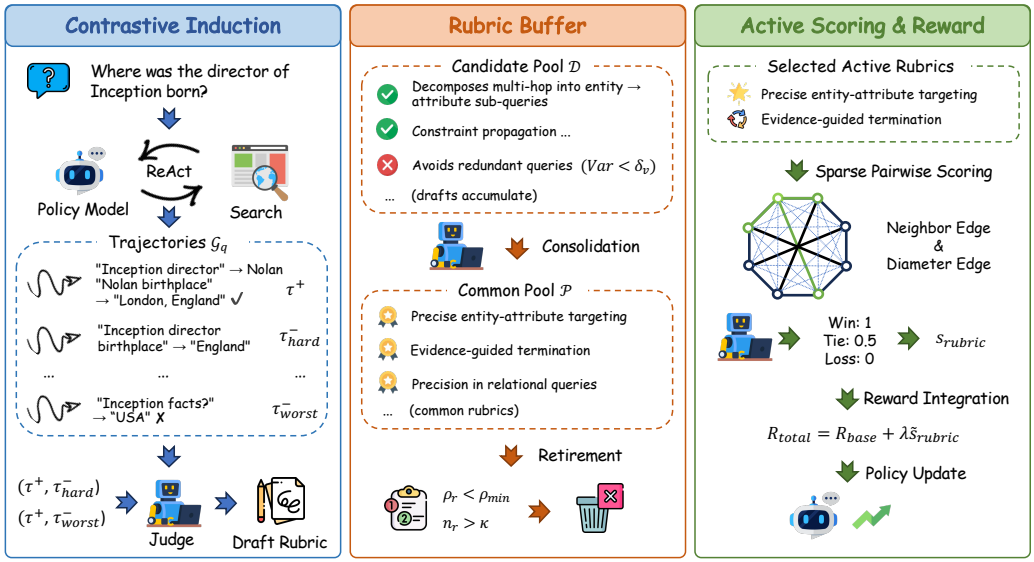


Figure 2: **Overview of ARBOR.** (a) Contrastive induction extracts query-local draft rubrics from trajectories within a query-group. (b) The rubric buffer $\mathcal{M}$ admits drafts into a candidate pool $\mathcal{D}$, consolidates them into a common pool $\mathcal{P}$, and retires stale rubrics, forming an online admission–consolidation–retirement lifecycle. (c) At each step, two active common rubrics are selected and used to score trajectories via sparse pairwise scoring, and the centered scores augment the base reward.

comparison between trajectories of similar correctness and isolates more subtle process differences. Both pairs are provided jointly to an external LLM, which summarizes a small set of natural-language process rubrics as query-local drafts. This design captures both coarse and fine process distinctions while reducing induction from exhaustive $O(K^2)$ pairing to constant cost.

The induction prompt is restricted to search behavior rather than answer content. The LLM is instructed to identify process behaviors that causally separate successful trajectories from failed ones, focusing on search strategy and reasoning process, such as query formulation, evidence use, and stopping judgments. Each induced rubric specifies both what a high-scoring response does and what a low-scoring response does along the dimension. The induction prompt appears in Appendix E.1.

Groups with uniform F1 scores skip the induction stage, as they provide no correctness contrast from which to infer new rubrics. Common rubrics already saved in $\mathcal{M}$, however, can still score these groups under Section 3.4, providing within-group discrimination when the outcome signal collapses. Corner cases are detailed in Appendix A.1.

The resulting drafts enter the memory pipeline of Section 3.3 and can affect future training only after surviving admission and consolidation.

3.3 Rubric Memory and Lifecycle

ARBOR maintains a single rubric memory $\mathcal{M} = (\mathcal{D}, \mathcal{P})$ shared across all queries. The candidate pool $\mathcal{D}$ temporarily stores the query-local drafts induced in Section 3.2, awaiting abstraction into reusable standards. The common rubric pool $\mathcal{P}$ stores rubrics that have been distilled from $\mathcal{D}$ and serves as the signal source for reward shaping in Section 3.4. Throughout training, $\mathcal{M}$ evolves through three mechanisms, namely admission, consolidation, and retirement, keeping its contents updated in step with the policy behavior distribution.

Admission. Each draft rubric scores the trajectories in its source group via the pairwise judging procedure of Section 3.4. Let s_i^r denote the score of trajectory τ_i under draft r. A draft is admitted to $\mathcal{D}$ only if its scores satisfy two conditions:

$$\mathrm{Var}_{i \in \mathcal{G}_q}(s_i^r) \geq \delta_v, \qquad \mathrm{Pearson}\big(\{s_i^r\}, \{F_1(\tau_i)\}\big) \geq \rho_{\min}. \tag{2}$$

The variance condition rules out drafts that fail to discriminate on the samples they were induced from. The correlation condition requires that rubric scores are aligned with outcome correctness, rejecting drafts that penalize correct behavior. Although outcome correctness does not fully characterize process quality, it provides an observable signal of task success, which process supervision ultimately aims to improve. Drafts failing either condition are discarded immediately.

Consolidation. After each rollout batch, ARBOR sends all candidate drafts together with the existing common rubrics in $\mathcal{P}$ to an external LLM for consolidation. The LLM is instructed to identify recurring process patterns across the candidates and abstract them into cross-query general standards. Existing rubrics in $\mathcal{P}$ are provided as context so that the LLM only produces standards covering new dimensions. Each output rubric is further deduplicated against $\mathcal{P}$ by sentence-embedding similarity as a safeguard. After consolidation, $\mathcal{D}$ is cleared. The consolidation prompt is provided in Appendix E.2.

Retirement. Common rubrics are not assumed to remain valid for the entire training run. As the policy evolves, a rubric that once discriminated good from bad may lose its power because the policy has uniformly mastered or failed the corresponding behavior. ARBOR tracks two long-term signals for each common rubric $r \in \mathcal{P}$, corresponding to the same two dimensions checked at admission stage but accumulated over the rubric's entire active lifetime. The first is a consecutive low-variance count n_r, the number of consecutive activations on which the within-group score variance under r falls below δ_v, measuring whether r still discriminates among trajectories. The second is a cumulative Pearson correlation ρ_r between the per-trajectory scores under r and the trajectory F1 across all activations of r, measuring whether r aligns with correct behavior. Either n_r exceeding its tolerance or $\rho_r < \rho_{\min}$ triggers removal from $\mathcal{P}$.

In addition, when consolidation attempts to write a new common rubric into a $\mathcal{P}$ that has already reached its capacity limit, the entry with the lowest cumulative within-group variance is replaced, provided its tracked statistics have matured over a sufficient number of activations. This prevents newly admitted rubrics from being evicted before their quality signals stabilize.

3.4 Process Scoring and Reward Shaping

At each training step, ARBOR selects a small active subset of common rubrics from $\mathcal{P}$, scores the trajectories in $\mathcal{G}_q$ under those rubrics via pairwise process comparison, and integrates the resulting scores into the reward after within-group centering. Because common rubrics are cross-query process standards independent of any specific query, they can discriminate among trajectories even in outcome-homogeneous groups where the base reward is identical for all samples, sustaining process-level supervision when the outcome signal collapses.

Active selection. At each step, ARBOR activates two common rubrics from $\mathcal{P}$. One slot is held by the rubric with the highest cumulative correlation ρ_r, serving as the primary process signal; the other rotates among the next strongest candidates by a least-recently-used policy. Activating only two rubrics avoids the linear blowup of judge cost and the within-group signal dilution that would result from scoring under the full pool. Rotation ensures that non-top rubrics still accumulate ρ_r and n_r, keeping the retirement mechanism of Section 3.3 operational. When $\mathcal{P}$ is empty, the step falls back to the base reward, so the cold-start phase introduces no unverified rubric signal.

Group-stage process scoring. Under each active common rubric, ARBOR performs within-group pairwise comparisons among the trajectories in $\mathcal{G}_q$ via an external LLM judge that returns *win*, *tie*, or *loss*. A full round-robin would require $O(|\mathcal{G}_q|^2)$ judge calls, which is prohibitively expensive. ARBOR therefore sorts the trajectories by base reward and builds a sparse connected graph with $O(|\mathcal{G}_q|)$ edges using two kinds of edges. As illustrated in Figure 2, neighbor edges link adjacent trajectories in the sorted order, providing fine-grained local comparisons between similarly ranked trajectories; diameter edges connect distant trajectories, providing high-contrast pairs for more decisive judgments. Each pairwise call randomizes the presentation order of the two trajectories, to mitigate positional bias of the LLM judge. *Wins*, *ties*, and *losses* are scored as 1, 0.5, and 0, respectively. The score of τ_i under a rubric is the mean over its incident edges, and averaging across the active rubrics yields the composite process score s_i^{rubric}.

Centering and reward integration. Before reward integration, ARBOR applies a variance filter within each query-group. If an active rubric's within-group variance falls below δ_v, its scores are discarded to avoid injecting uninformative noise. The scores from the filtered active rubrics are then centered within each query-group,

$$\tilde{s}_i^{\text{rubric}} = s_i^{\text{rubric}} - \frac{1}{|\mathcal{G}_q|} \sum_{j \in \mathcal{G}_q} s_j^{\text{rubric}}, \tag{3}$$

and the final reward uses additive shaping,

$$R_i^{\text{total}} = R_i^{\text{base}} + \lambda \cdot \tilde{s}_i^{\text{rubric}}, \qquad R_i^{\text{base}} = \begin{cases} R_i^{\text{F1}} & \text{if format valid} \\ -1 & \text{otherwise} \end{cases}, \tag{4}$$

where R_i^{F1} is the token-level F1 of the final answer and λ is fixed throughout training. Group centering gives the rubric term zero mean within each query-group, and negative centered scores are further attenuated by a factor α, encouraging stronger search behavior without over-penalizing merely less-preferred trajectories. For format-invalid trajectories, the rubric term is forced to zero, preserving the format penalty imposed by R_i^{base}.

4 Experiments

4.1 Experimental Setup

Datasets. Our training follows a two-stage SFT-then-RL procedure. SFT uses a search-tool subset of Tool-Star (Dong et al., 2025) and STILL (Min et al., 2024), filtered to remove Python-tool examples, yielding approximately 16K training examples. RL training uses 2K QA examples randomly sampled from the ARPO Deep Reasoning Tasks (Dong et al., 2026). Evaluation is on four multi-hop QA benchmarks: Bamboogle (Press et al., 2023), HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022) and 2WikiMultiHopQA (Ho et al., 2020). The test split strategy follows the convention of ARPO (Dong et al., 2026). Benchmark and training details are provided in Appendices B.1 and B.2.

Baselines. We use Qwen3-8B (Yang et al., 2025) as the main backbone, with Qwen3-4B and Qwen3-14B for cross-scale verification, comparing three methods on each backbone: (1) TIR Prompting, which injects the search-tool specification but performs no RL fine-tuning; (2) GRPO (Shao et al., 2024), which uses base reward as the sole reward; and (3) DAPO (Yu et al., 2025), which discards outcome-homogeneous query-groups from policy updates, providing a contrast on how to handle the zero-gradient groups that ARBOR exploits. We additionally compare against CaRR/C-GRPO-style (Zhang et al., 2026) and PRInTS-RM (Lee et al., 2026) on Qwen3-8B, adapting query-local rubric rewards and a frozen process reward model, respectively, to our search-agent training setup. Adaptation details and full results are provided in Appendix C.3.

Evaluation Metrics. We report Exact Match (EM), token-level F1, and LLM-judge accuracy with Qwen3-Plus[1] (Alibaba Cloud, 2025) as the judge. All numbers are pass@1 averaged over 5 independent evaluation runs to reduce sampling variance. The LLM-judge prompt is given in Appendix E.4. Detailed implementation settings and external-LLM usage are provided in Appendix B.3.

4.2 Main Results

Table 1 presents the main results across four multi-hop QA benchmarks. ARBOR achieves the best average performance at every model scale. We highlight three observations from the results.

ARBOR consistently improves over outcome-only baselines. ARBOR improves average LLM-judge accuracy over GRPO by **4.0**, **4.2**, and **2.0** points at 4B, 8B, and 14B, and also uniformly outperforms DAPO at all three scales. The improvements are consistent across evaluation metrics, with gains observed not only in LLM-judge accuracy but also in EM and F1. TIR Prompting performs substantially worse than all RL-trained variants, confirming that tool access alone is insufficient without policy optimization. Its non-monotonic scaling is mainly caused by answer-format failures under the strict evaluator, as diagnosed in Appendix C.1.

[1]Model ID: `qwen-plus-2025-04-28`. Accessed from May to August 2026.

Table 1: **Overall performance on 4 knowledge-intensive reasoning benchmarks.** ARBOR achieves the best average EM, F1 and LLM-judge accuracy at all three scales. The top two outcomes in each column are **bolded** and underlined.

Method	Bamboogle			HotpotQA			MuSiQue			2Wiki			Avg.		
	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.
Qwen3-4B															
+ TIR Prompting	46.4	55.3	61.1	30.4	38.5	46.2	9.2	16.2	19.8	32.8	42.3	47.2	29.7	38.1	43.6
+ GRPO	61.6	69.8	71.0	44.9	54.5	57.4	**17.0**	**26.9**	26.5	60.9	68.3	67.6	46.1	54.9	55.6
+ DAPO	63.7	72.7	74.2	44.2	53.9	57.0	16.4	25.9	25.2	60.2	67.7	68.1	46.1	55.1	56.1
+ ARBOR	**65.1**	**73.0**	**76.2**	**45.4**	**55.6**	**59.7**	**17.0**	25.7	**27.8**	**64.2**	**72.4**	**74.6**	**47.9**	**56.7**	**59.6**
Qwen3-8B															
+ TIR Prompting	31.2	38.1	41.3	15.7	20.2	22.9	6.2	9.8	11.6	9.7	13.8	15.1	15.7	20.5	22.7
+ GRPO	61.0	69.4	72.2	44.4	55.0	57.8	17.4	26.7	26.6	**64.8**	71.0	71.1	46.9	55.5	56.9
+ DAPO	61.9	70.1	73.1	45.8	54.9	57.7	17.0	25.8	26.2	63.8	69.7	69.6	47.1	55.1	56.7
+ ARBOR	**65.3**	**72.9**	**76.5**	**47.8**	**58.6**	**62.9**	**18.3**	**26.8**	**29.5**	64.2	**73.3**	**75.3**	**48.9**	**57.9**	**61.1**
Qwen3-14B															
+ TIR Prompting	42.1	50.3	49.9	27.8	34.6	39.2	8.6	16.0	17.4	26.1	31.9	34.4	26.2	33.2	35.2
+ GRPO	61.9	71.5	72.8	48.1	58.4	62.3	**19.4**	**29.9**	**31.6**	**69.8**	76.2	76.4	49.8	59.0	60.8
+ DAPO	65.8	74.1	74.9	46.2	57.5	61.1	17.2	27.1	27.3	67.6	73.4	72.9	49.2	58.0	59.1
+ ARBOR	**67.5**	**75.8**	**78.6**	**48.6**	**59.1**	**63.6**	19.1	28.7	**31.6**	**69.8**	**76.3**	**77.3**	**51.3**	**60.0**	**62.8**

Table 2: **Reward-homogeneous group fractions during ARBOR training.** Groups are measured on R^{base} (before rubric shaping) and R^{total} (after rubric shaping).

Model	Reward	All-correct↓	All-wrong↓	Mixed-uni.↓	Total↓
Qwen3-4B	R^{base}	26.9%	14.7%	2.9%	44.5%
	R^{total}	22.2%	6.0%	2.2%	30.4%
	Δ_{rel}	−17.5%	−59.2%	−24.1%	−31.7%
Qwen3-8B	R^{base}	23.5%	14.3%	2.9%	40.7%
	R^{total}	16.2%	5.6%	1.6%	23.4%
	Δ_{rel}	−31.1%	−60.8%	−44.8%	−42.5%
Qwen3-14B	R^{base}	30.1%	14.3%	3.5%	48.0%
	R^{total}	23.9%	6.6%	2.0%	32.5%
	Δ_{rel}	−20.6%	−53.8%	−42.9%	−32.3%

Exploiting outcome-homogeneous groups outperforms discarding them. DAPO's gains over GRPO are inconsistent across scales, turning negative at 14B (−1.7 points), while ARBOR consistently outperforms GRPO at all scales. An important difference is how the methods handle outcome-homogeneous groups: DAPO discards them entirely from policy updates, whereas ARBOR retains them and applies reusable process rubrics to provide within-group discrimination. ARBOR's lead over DAPO is uniformly large (**+3.5** to **+4.4** points across all backbones), confirming that exploiting these groups through process reward is more effective than discarding them.

Process supervision translates into semantic answer gains. ARBOR's largest gains appear on LLM-judge accuracy: +4.0, +4.2, and +2.0 points over GRPO at 4B, 8B, and 14B, exceeding the corresponding EM gains (+1.8, +2.0, +1.5) and F1 gains (+1.8, +2.4, +1.0). This pattern is consistent with ARBOR's design: rubrics reward better query formulation, evidence use, and answer synthesis rather than final-answer string overlap. The result suggests that reusable process supervision improves answer quality in ways that are better captured by semantic evaluation than by lexical-overlap metrics alone. To ensure these gains are not evaluator-specific, Appendix C.2 verifies the same trend under DeepSeek-V4-Pro (DeepSeek-AI, 2026) as an independent evaluator.

4.3 No-Gradient Group Rescue

A direct test of the motivating claim from Section 1 is whether rubric scoring provides within-group discrimination on groups where outcome-only reward produces zero gradient. Table 2 reports the

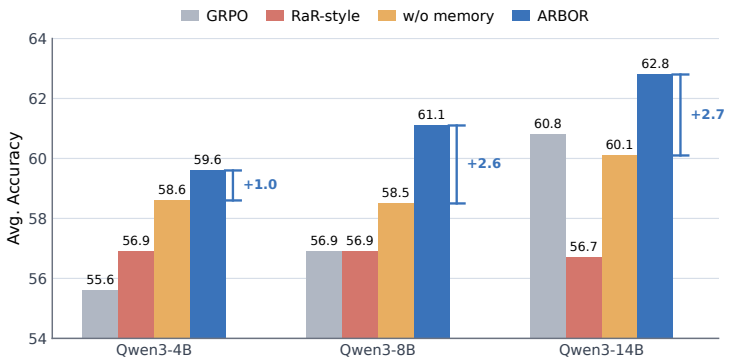


Figure 3: **Effect of reusable rubric memory.** Average LLM-judge accuracy across four benchmarks.

fraction of groups with zero within-group reward variance, measured first under R^{base} alone and then under R^{total}. A drop indicates that rubric scoring introduced reward discrimination that the outcome signal could not provide. Groups are further split by outcome pattern: *all-correct* groups have F1=1 on every trajectory with valid format, *all-wrong* groups have F1=0 on every trajectory including format-invalid cases, and *mixed-uniform* groups share the same partial F1 across all trajectories.

Rubric scoring reduces all three types at every scale. The effect concentrates on *all-wrong* groups, where the relative reduction reaches **54–61%**. These groups represent queries the policy has not yet learned to solve, and rubric scoring directly helps here by distinguishing more promising search strategies from less promising ones, providing within-group discrimination that the base reward cannot. The reduction on *all-correct* groups is smaller (**18–31%**), as the active rubrics are primarily induced from success-failure contrasts and less attuned to the subtler process differences among trajectories that all succeed. Overall, rubric scoring introduces nonzero within-group reward variance in **32–42%** of outcome-homogeneous groups, enabling policy learning on these groups.

4.4 Effect of Reusable Rubric Memory

To isolate the contribution of reusable rubric memory, Figure 3 compares ARBOR with two no-memory rubric variants. RaR-style, adapted from Rubrics-as-Rewards (Gunjal et al., 2026), uses static query-specific rubrics generated from the question and reference answer to score rollouts of the corresponding query. The w/o memory variant uses ARBOR's contrastive induction but discards each rubric after scoring its source query-group. Experimental details and full results are in Appendix C.4.

No-memory rubric feedback alone does not explain ARBOR's gains. RaR-style is unstable, improving over GRPO at 4B (+1.3 points), matching it at 8B, and falling below it at 14B (-4.1 points). The w/o memory variant is stronger than RaR-style at all scales, indicating that contrastive process rubrics are more useful than reference-conditioned instance checklists. Nevertheless, ARBOR outperforms w/o memory by **1.0**, **2.6**, and **2.7** points and RaR-style by **2.7**, **4.2**, and **6.1** points at 4B, 8B, and 14B. These margins show that the benefit comes not only from rubric-shaped reward, but from filtering and reusing local observations across query-groups. Appendix C.5 further shows that combining common and query-local rubrics does not help improve performance.

4.5 Role of Outcome Signal

Beyond the base reward, F1 serves two distinct roles in ARBOR: its trajectory ranking determines the contrastive anchors used for induction, while its correlation with rubric scores guides rubric admission, maintenance, and selection. We examine these roles separately on Qwen3-8B, changing one component at a time relative to default ARBOR. Table 3 reports the per-benchmark results.

Effect of outcome-correlation filtering. Removing F1 correlation from rubric admission, maintenance, and selection, while retaining variance-based filtering, reduces average accuracy from 61.1 to 59.4. The ablated variant still outperforms GRPO by 2.5 points, showing that rubric feedback

Table 3: **Role of outcome signal in ARBOR.** LLM-judge accuracy when removing F1-correlation from rubric management or replacing F1-based anchor ranking with MC estimates.

Variant	Bamboogle	HotpotQA	MuSiQue	2Wiki	Avg.
GRPO	72.2	57.8	26.6	71.1	56.9
w/o F1-correlation	73.3	61.5	28.8	74.0	59.4
w/ MC anchors	75.2	61.2	29.3	75.0	60.2
ARBOR	**76.5**	**62.9**	**29.5**	**75.3**	**61.1**

Table 4: **Top-5 most reused common rubrics during training.** Uses counts scored query-groups.

Common rubric	Uses
Precise Entity-Attribute Targeting with Canonical Framing	101
Evidence-Sufficiency-Guided Termination with Cross-Validated Convergence	84
Precision in Relational Query Formulation	55
Constraint-Integrated Entity-Attribute Binding	54
Constraint-Consistent Interpretation of Question Semantics and Intent	54

remains useful without correlation-based management. Nevertheless, the 1.7-point drop indicates that F1 correlation improves rubric management by aligning process supervision with task performance.

Choice of anchor metric. To test whether anchor selection benefits from a rollout-based value estimate, we replace F1-based ranking with Monte Carlo (MC) accuracy estimated from four independent continuations per trajectory after the final search result. This changes only contrastive-anchor selection; rubric management and the base reward continue to use F1. The MC-anchor variant achieves an average accuracy of 60.2, compared with 61.1 for default ARBOR. Thus, MC estimation provides no observed performance benefit despite requiring additional rollouts and introducing sampling noise, supporting F1 as a simpler and more efficient anchor-selection signal.

4.6 Common-Rubric Reuse

We assess cross-query reuse by counting the query-groups scored by each common rubric during training. As detailed in Appendix C.6, the top 5% of rubrics account for 20% of all scoring events and the top 20% account for 44%, showing that a small core is repeatedly reused across diverse query-groups. Table 4 lists the most reused rubrics, whose generic descriptions contain no query-specific entities or facts. Appendix D gives a concrete consolidation example.

We further test whether the learned rubrics capture useful process-quality signals through test-time reuse. Appendix C.7 shows that the frozen rubrics select better test-time trajectories without reference answers or adaptation, improving Best-of-5 accuracy on all four benchmarks and the average by 4.5 points. Together, these results show that common rubrics encode useful, query-general criteria for distinguishing higher-quality search trajectories, supporting their effectiveness as process supervision.

5 Conclusion

In this work, we introduced ARBOR, a reusable process-reward framework designed to address the limitations of outcome-only rewards in search-agent RL. ARBOR induces natural-language process rubrics from contrastive trajectories, consolidates them into a reusable cross-query memory, and manages the memory through an admission–consolidation–retirement lifecycle. This design jointly achieves process-level supervision, cross-query consistency, and co-evolution with the policy. Experiments across three model scales and four multi-hop QA benchmarks show that ARBOR consistently improves over outcome-only RL baselines, while ablations and reuse analyses confirm the value of reusable rubric memory. These results suggest that process knowledge in agent training need not be ephemeral: when accumulated into reusable criteria that co-evolve with the policy, it provides a persistent and growing source of supervision that outcome signals alone cannot offer.

AI USE STATEMENT

Generative AI tools were used in two roles in this work. First, Qwen3-Plus was used as an experimental component for rubric induction, consolidation, pairwise trajectory judging, and final-answer evaluation, as described in the paper. Second, generative AI tools assisted with manuscript preparation, including English-language editing, translation, and LaTeX formatting.

All AI-assisted outputs were reviewed by the authors. The reported results were produced by the experimental pipelines described in the paper, and the numerical results, citations, and scientific claims were manually verified by the authors. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work improves search-agent training by adding process-level supervision and does not introduce new ethical concerns. All training and evaluation data come from publicly available benchmarks, and we will release our code and models after the review process to support transparency and reproducibility.

REPRODUCIBILITY STATEMENT

We support reproducibility by providing the ARBOR formulation and reward construction in Section 3.1. Appendix B documents the benchmark splits, SFT and RL settings, implementation details, and hyperparameters, while Appendix E reproduces all prompts used by the framework and evaluator. All datasets and base models are publicly available and cited in the paper. We will release our code and trained models after the review process.

REFERENCES

Alibaba Cloud. Qwen-Plus Model Card, 2025. URL https://www.aliyun.com/benefit/scene/qwen-plus. Model id: qwen-plus-2025-04-28. Accessed: 2026-05.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/98ed250b203d1ac6b24bbcf263e3d4a7-Paper-Conference.pdf.

DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-Star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*, 2025. URL https://arxiv.org/abs/2505.16410v1.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng

Dou. Agentic Reinforced Policy Optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TX4k7BF6aO.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean M. Hendryx. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=c1bTcrDmt4.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. Llm-rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13806–13834, 2024. URL https://aclanthology.org/2024.acl-long.745/.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G. Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Gonugondla, Hunter Lang, Yue Yu, Yundi Qian, Maryam Fazel-Zarandi, Licheng Yu, Amine Benhalloum, Hany Awadalla, and Manaal Faruqui. Advancedif: Rubric-based benchmarking and reinforcement learning for advancing llm instruction following, 2025. URL https://arxiv.org/abs/2511.10507.

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. URL https://aclanthology.org/2020.coling-main.580/.

Ruipeng Jia, Yunyi Yang, Yuxin Wu, Yongbo Gai, Siyuan Tao, Mengyu Zhou, Jianhe Lin, Xiaoxi Jiang, and Guanjun Jiang. Open rubric system: Scaling reinforcement learning with pairwise adaptive rubric. *arXiv preprint arXiv:2602.14069*, 2026. URL https://arxiv.org/abs/2602.14069v2.

Pengcheng Jiang, Xueqiang Xu, Jiacheng Lin, Jinfeng Xiao, Zifeng Wang, Jimeng Sun, and Jiawei Han. s3: You don't need that much data to train a search agent via RL. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21599–21617, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1095. URL https://aclanthology.org/2025.emnlp-main.1095/.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Seungone Kim, Jay Shin, yejin cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, S Shin, Ryan, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 29927–29962, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/803485352e61e3ebf41221e4776c9fd4-Paper-Conference.pdf.

Jaewoo Lee, Archiki Prasad, Justin Chen, Zaid Khan, Elias Stengel-Eskin, and Mohit Bansal. PRInTS: Reward modeling for long-horizon information seeking. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34120–34138, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1574. URL https://aclanthology.org/2026.acl-long.1574/.

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. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025. URL https://aclanthology.org/2025.emnlp-main.276/.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Kun Luo, Hongjin Qian, Zheng Liu, Ziyi Xia, Shitao Xiao, Siqi Bao, Jun Zhao, and Kang Liu. Infoflow: Reinforcing search agent via reward density optimization. *arXiv preprint arXiv:2510.26575*, 2025. URL https://arxiv.org/abs/2510.26575v1.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision, 2024. URL https://arxiv.org/abs/2406.06592.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems, 2024. URL https://arxiv.org/abs/2412.09413.

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. URL https://aclanthology.org/2023.findings-emnlp.378/.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16. IEEE, 2020. URL https://doi.org/10.1109/SC41405.2020.00024.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, Hannaneh Hajishirzi, and Pang Wei Koh. Dr tulu: Reinforcement learning with evolving rubrics for deep research, 2026. URL https://arxiv.org/abs/2511.19399.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300v3.

Leheng Sheng, Wenchang Ma, Ruixin Hong, Xiang Wang, An Zhang, and Tat-Seng Chua. Reinforcing Chain-of-Thought Reasoning with Self-Evolving Rubrics. *arXiv preprint arXiv:2602.10885*, 2026. URL https://arxiv.org/abs/2602.10885v1.

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. URL https://arxiv.org/abs/2503.05592v2.

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. URL https://aclanthology.org/2022.tacl-1.31/.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024. URL https://aclanthology.org/2024.acl-long.510/.

Pengkai Wang, Linus, Pengwei Liu, Zhijie Sang, Congkai Xie, and Hongxia Yang. Infimed-orbit: Aligning llms on open-ended complex tasks via rubric-based incremental training, 2025. URL https://arxiv.org/abs/2510.15859.

Yunjia Xi, Jianghao Lin, Yongzhao Xiao, Zheli Zhou, Rong Shan, Te Gao, Jiachen Zhu, Weiwen Liu, Yong Yu, and Weinan Zhang. A survey of llm-based deep search agents: Paradigm, optimization, evaluation, and challenges. *arXiv preprint arXiv:2508.05668*, 2025.

Tianle Xia, Ming Xu, Lingxiang Hu, Yiding Sun, Wenwei Li, Linfang Shang, Liqun Liu, Peng Shu, Huan Yu, and Jie Jiang. Search-P1: Path-Centric Reward Shaping for Stable and Efficient Agentic RAG Training. *arXiv preprint arXiv:2602.22576*, 2026. URL https://arxiv.org/abs/2602.22576v1.

Lipeng Xie, Sen Huang, Zhuo Zhang, Anni Zou, Yunpeng Zhai, Dingchao Ren, Kezun Zhang, Haoyuan Hu, Boyin Liu, Haoran Chen, Zhaoyang Liu, and Bolin Ding. Auto-rubric: Learning from implicit weights to explicit rubrics for reward modeling, 2026a. URL https://arxiv.org/abs/2510.17314.

Yutao Xie, Nathaniel Thomas, Nicklas Hansen, Yang Fu, Li Erran Li, and Xiaolong Wang. TIPS: Turn-level Information-Potential Reward Shaping for Search-Augmented LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=eBMOr6a84z.

Peiran Xu, Zhuohao Li, Xiaoying Xing, Guannan Zhang, Debiao Li, and Kunyu Shi. Hybrid reward normalization for process-supervised non-verifiable agentic tasks. *arXiv preprint arXiv:2509.25598*, 2025. URL https://arxiv.org/abs/2509.25598v1.

Ran Xu, Tianci Liu, Zihan Dong, Tony Yu, Ilgee Hong, Carl Yang, Linjun Zhang, Tao Zhao, and Haoyu Wang. Alternating reinforcement learning for rubric-based reward modeling in non-verifiable llm post-training. *arXiv preprint arXiv:2602.01511*, 2026. URL https://arxiv.org/abs/2602.01511v2.

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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS. In *11th International Conference on Learning Representations, ICLR 2023*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Jiajie Zhang, Xin Lv, Ling Feng, Lei Hou, and Juanzi Li. Chaining the evidence: Robust reinforcement learning for deep search agents with citation-aware rubric rewards. *arXiv preprint arXiv:2601.06021*, 2026. URL https://arxiv.org/abs/2601.06021.

Qi Zhang, Shouqing Yang, Lirong Gao, Hao Chen, Xiaomeng Hu, Jinglei Chen, Jiexiang Wang, Sheng Guo, Bo Zheng, Haobo Wang, and Junbo Zhao. LeTS: Learning to think-and-search via process-and-outcome reward hybridization. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5109–5122, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.257. URL https://aclanthology.org/2025.emnlp-main.257/.

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. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 58701–58729. Curran Associates, Inc., 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/54e1381d0c0598127b90af4c940fd3d9-Paper-Conference.pdf.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21805–21830, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1106. URL https://aclanthology.org/2025.emnlp-main.1106/.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pp. 400–410, 2024. URL https://aclanthology.org/2024.acl-demos.38/.

Wenlin Zhong, Jinluan Yang, Yiquan Wu, Yi Liu, Jianhang Yao, and Kun Kuang. Sight: Reinforcement Learning with Self-Evidence and Information-Gain Diverse Branching for Search Agent. *arXiv preprint arXiv:2602.11551*, 2026. URL https://arxiv.org/abs/2602.11551v1.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, Hengtong Lu, Wei Chen, Yan Xie, and Mingli Song. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general llm reasoning, 2026. URL https://arxiv.org/abs/2508.16949.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An LLM post-training framework for RL Scaling. https://github.com/THUDM/slime, 2025. GitHub repository. Corresponding author: Xin Lv.

A METHOD DETAILS

A.1 CONTRASTIVE INDUCTION CORNER CASES

This section records the rules ARBOR follows when correctness scores within a query-group tie and when the two contrast pairs collapse into one, complementing Section 3.2.

Tie-breaking for anchor selection. When multiple trajectories share the highest F1, τ^+ is the one with the shortest response. For τ^-_{hard} and τ^-_{worst}, the same shortest-response tie-breaker is applied among trajectories sharing the selected lower F1 value.

Collapsed contrast pairs. When only one trajectory falls strictly below τ^+ in F1, τ^-_{hard} and τ^-_{worst} resolve to the same trajectory; in this case only the $(\tau^+, \tau^-_{\text{hard}})$ pair is sent to the LLM. When no trajectory falls below τ^+, the group has no correctness divergence and induction is skipped entirely.

Homogeneous groups. Groups with uniform F1 scores, including all-correct, all-wrong, and mixed-uniform groups, skip induction because no correctness contrast can be formed. Existing common rubrics still score these groups for reward shaping. Induction is also skipped when fewer than two format-valid trajectories exist in the group.

B TRAINING AND EVALUATION DETAILS

B.1 EVALUATION BENCHMARKS

We evaluate on the following four multi-hop QA benchmarks:

- **HotpotQA** (Yang et al., 2018) is a Wikipedia-based multi-hop QA benchmark that requires aggregating evidence across multiple supporting documents to answer a question. It is characterized by two representative question types: reasoning over sentence-level supporting facts across documents, and comparison questions that contrast properties of two entities.
- **2WikiMultiHopQA** (Ho et al., 2020) is a multi-hop QA dataset constructed from both Wikipedia and Wikidata, combining unstructured text with structured knowledge to support multi-step reasoning. It provides explicit evidence and reasoning paths, ensuring that answering each question requires genuine multi-hop inference.
- **MuSiQue** (Trivedi et al., 2022) is designed to resist shortcut solutions by composing connected single-hop questions into multi-hop ones, such that proper multi-hop reasoning is required by construction. Unlike earlier benchmarks that can often be resolved with partial evidence, MuSiQue ensures that each step of the reasoning chain is necessary to reach the correct answer.
- **Bamboogle** (Press et al., 2023) is a hand-crafted benchmark of 125 two-hop compositional questions designed to be difficult for standard search engines to answer directly, while the supporting evidence for each question can be found in Wikipedia. It evaluates models on varied compositional questions that require combining information across multiple retrieval steps.

Following Tool-Star (Dong et al., 2025) and ARPO (Dong et al., 2026), we evaluate on fixed held-out test splits: 200 examples for HotpotQA, 200 for 2WikiMultiHopQA, 200 for MuSiQue, and 125 for Bamboogle.

B.2 SFT AND RL TRAINING

Supervised Fine-Tuning. Starting from Tool-Star (Dong et al., 2025) and STILL (Min et al., 2024), we filter out all examples that invoke Python as a tool and retain only search-tool interactions, yielding approximately 16K training examples. Fine-tuning is conducted with LLaMAFactory (Zheng et al., 2024) using full-parameter optimization, DeepSpeed ZeRO Stage 3 (Rajbhandari et al., 2020), and FlashAttention-2 (Dao, 2024). We train for 3 epochs with a maximum sequence length of 15,000 tokens, an effective batch size of 16, a learning rate of 7×10^{-6} with cosine decay and 10% linear warmup, in BF16 precision.

Reinforcement Learning. RL training runs for 60 rollout steps using Slime (Zhu et al., 2025). We sample 8 trajectories per prompt with a rollout batch size of 32, sampling temperature 0.7, a maximum context length of 36,864 tokens, and a maximum response length of 8,192 tokens. Each trajectory is limited to at most 12 agent turns and 10 search calls. The policy is updated with GRPO with token-level importance sampling (TIS), KL and entropy coefficients set to 0, and symmetric clipping with $\epsilon_{\text{clip}} = 0.2$. Optimization uses Adam with learning rate 1×10^{-6} and weight decay 0.01.

B.3 Implementation Details

All training runs on a single node with 8×H100 GPUs using Slime (Zhu et al., 2025) as the RL framework. The search tool is Google Search, returning 10 results per query. The base reward is token-level F1 with a format penalty. For ARBOR, induction, consolidation, and pairwise judging all call Qwen3-Plus (Alibaba Cloud, 2025) as the external LLM. Consolidation deduplication uses BGE-M3 (Chen et al., 2024) sentence embeddings, and the reward shaping coefficient is fixed to $\lambda = 0.1$. Remaining hyperparameters are listed in Appendix B.4. An ablation of the rubric reward coefficient λ is provided in Appendix C.8.

Table 5 reports external-LLM usage for a complete Qwen3-8B RL training run. Pairwise judging dominates the external-LLM overhead, while induction and consolidation contribute only a small fraction. The total estimated API cost is approximately $36, based on the official Alibaba Cloud Model Studio pricing for Qwen-Plus. The sparse comparison graph and the activation of only two common rubrics per step help keep this cost manageable.

Table 5: External-LLM usage for a complete Qwen3-8B RL training run.

Component	Input (M)	Output (M)	Cost (USD)
Rubric induction	10.23	0.73	$1.34
Pairwise judging	309.84	0.42	$34.54
Rubric consolidation	0.65	0.04	$0.08
Total	**320.72**	**1.19**	**$35.96**

B.4 ARBOR Hyperparameters

Table 6 lists the default ARBOR hyperparameters used in the main experiments. We elaborate on the key design choices below.

Consolidation is performed after each rollout batch using the admitted candidate drafts. $\kappa = 5$ is the maximum number of consecutive low-variance activations a common rubric is allowed before retirement. The deduplication similarity threshold is set to 0.9, above which a newly produced common rubric is considered a duplicate of an existing one under BGE-M3 (Chen et al., 2024) cosine similarity.

C Additional Experimental Results

C.1 Format Sensitivity of TIR Prompting

The official evaluation used in Table 1 is strict-format gated: a response is scored only if it produces a non-empty prediction and terminates with the required final-answer format. If the termination reason indicates a formatting or finalization failure, all official metrics are set to zero, including EM, F1, and LLM-judge accuracy. This protocol matches the training reward, where finalization is part of the task rather than a post-hoc parsing detail.

To diagnose the non-monotonic TIR Prompting results in the main table, Table 7 separates formatting validity from answer quality. *Official* accuracy follows the strict protocol above, *valid-only* accuracy evaluates only responses that satisfy the required format, and *recovery* is the gap between the two LLM accuracies.

Table 6: Default ARBOR hyperparameters used in the main experiments.

Hyperparameter	Notation	Default
Rubric shaping coefficient	λ	0.1
Admission variance threshold	δ_v	0.05
Correlation threshold	$\rho_{\min}$	0.0
Retirement tolerance	κ	5
Negative score attenuation factor	α	0.25
Maximum common pool size	$\lvert\mathcal{P}\rvert_{\max}$	6
Active common rubrics per step	—	2
Deduplication similarity threshold	—	0.9

Table 7: **Format sensitivity of TIR Prompting.** All values are percentages averaged over four benchmarks and five evaluation runs.

Model	**Valid Rate**	**Official LLM Acc.**	**Valid-only LLM Acc.**	**Recovery**
Qwen3-4B	89.7	43.6	48.4	**+4.8**
Qwen3-8B	45.1	22.7	48.6	**+25.9**
Qwen3-14B	68.5	35.2	52.4	**+17.2**

Table 7 shows that the non-monotonic TIR results are mainly driven by formatting failures. Qwen3-8B and Qwen3-14B have much lower valid rates than Qwen3-4B, which sharply lowers their official LLM-judge accuracies. On valid responses, however, they reach 48.6% and 52.4% LLM-judge accuracy, comparable to or higher than Qwen3-4B. Thus, the apparent weakness of larger TIR backbones primarily reflects unreliable prompt-only finalization under a strict protocol, not weaker search reasoning. We keep the strict official scores in the main table because correct finalization is part of the tool-use task and is also enforced during RL training.

C.2 Cross-Judge Robustness

To check whether the main LLM-judge trends depend on the same model family used by ARBOR's reward construction, we re-evaluate all final answers with DeepSeek-V4-Pro (DeepSeek-AI, 2026) as an independent post-hoc judge. DeepSeek-V4-Pro is not used for rubric induction, consolidation, pairwise scoring, or reward construction. Table 8 compares average LLM-judge accuracy under Qwen3-Plus and DeepSeek-V4-Pro. The independent judge preserves the same ranking: ARBOR remains the best method at all three scales, and its gains over GRPO remain close to the Qwen3-Plus gains.

Overall, DeepSeek-V4-Pro is slightly stricter than Qwen3-Plus: its average scores are lower by 0.4–1.2 points across methods, but the drop is small and uniform. Despite this stricter independent judge, the relative ordering is preserved and ARBOR's gains over GRPO remain nearly unchanged (+3.6, +4.0, and +1.9 points at 4B, 8B, and 14B). Thus, the main LLM-judge conclusion is robust across judge families rather than driven by a Qwen3-Plus-specific preference.

C.3 Additional Reward Baseline Comparisons

We compare ARBOR with CaRR/C-GRPO-style and PRInTS-RM on Qwen3-8B to evaluate alternative rubric-based and learned process rewards. Both baselines use the same SFT initialization, 2K RL training set, Google Search backend, and evaluation protocol as ARBOR (Section 4.1). We adapt their reward mechanisms to this shared setting rather than reproduce their original end-to-end systems.

CaRR/C-GRPO-style adaptation. We adapt CaRR/C-GRPO (Zhang et al., 2026) by pre-generating 5–8 answer-conditioned, query-local rubrics for each training question. Rubric evaluation

Table 8: **Cross-judge robustness of average LLM-judge accuracy.** Qwen3-Plus is the main evaluation judge, while DeepSeek denotes DeepSeek-V4-Pro used only as an independent post-hoc judge. Δ denotes DeepSeek accuracy minus Qwen3-Plus accuracy. Gain over GRPO denotes the average accuracy difference from GRPO under the corresponding judge.

Backbone	Method	Judge Accuracy			Gain over GRPO	
		Qwen3-Plus	DeepSeek	Δ	Qwen3-Plus	DeepSeek
Qwen3-4B	TIR Prompting	43.6	42.4	-1.2	-12.0	-12.7
	GRPO	55.6	55.1	-0.5	0.0	0.0
	DAPO	56.1	55.6	-0.5	$+0.5$	$+0.5$
	ARBOR	**59.6**	**58.7**	-0.9	$+4.0$	$+3.6$
Qwen3-8B	TIR Prompting	22.7	22.0	-0.7	-34.2	-34.6
	GRPO	56.9	56.5	-0.4	0.0	0.0
	DAPO	56.7	55.7	-1.0	-0.2	-0.8
	ARBOR	**61.1**	**60.6**	-0.5	$+4.2$	$+4.0$
Qwen3-14B	TIR Prompting	35.2	34.6	-0.6	-25.6	-25.3
	GRPO	60.8	59.9	-0.9	0.0	0.0
	DAPO	59.1	58.6	-0.5	-1.7	-1.3
	ARBOR	**62.8**	**61.8**	-1.0	$+2.0$	$+1.9$

uses the URLs, titles, and snippets cited by the agent in our Google Search environment. We retain the positive C-GRPO reward composition with a mixture weight of 0.3, while using our strict token-level F1 outcome signal and format protocol. All ARBOR buffer operations and cross-query reuse are disabled. We therefore refer to this baseline as CaRR/C-GRPO-style, a matched adaptation rather than an official reproduction.

PRInTS-RM adaptation. We use the official frozen PRInTS process reward model (Lee et al., 2026), retaining its recursive step-level scoring and trajectory-summary updates. PRInTS originally uses these scores to rank candidate next steps at inference time. For our training comparison, we instead aggregate the scores of completed search steps into a trajectory-level process reward and combine it with the outcome reward within GRPO. ARBOR's rubric induction, consolidation, and retirement are disabled, while the policy-training setup remains unchanged. We denote this training-time adaptation as PRInTS-RM.

Table 9: **Additional reward baseline comparisons on Qwen3-8B.** LLM-judge accuracy (%) on four benchmarks and their average. CaRR/C-GRPO-style and PRInTS-RM are adapted to our search-agent training setting.

Method	Bamboogle	HotpotQA	MuSiQue	2Wiki	Avg.
GRPO	72.2	57.8	26.6	71.1	56.9
CaRR/C-GRPO-style	71.8	62.0	26.2	**77.0**	59.3
PRInTS-RM	72.8	59.2	26.8	71.7	57.6
ARBOR	**76.5**	**62.9**	**29.5**	75.3	**61.1**

Table 9 shows that CaRR/C-GRPO-style achieves the highest accuracy on 2Wiki, while ARBOR performs best on the other three benchmarks and on average, exceeding CaRR/C-GRPO-style by 1.8 points. PRInTS-RM improves average accuracy over GRPO from 56.9 to 57.6, but remains below ARBOR on all four benchmarks, with a 3.5-point average gap. These results support ARBOR's effectiveness relative to the adapted reward baselines in this controlled Qwen3-8B setting.

C.4 Setup and Full Results for Rubric Memory Ablation

Table 10 provides the per-benchmark results for the rubric memory ablation discussed in Section 4.4.

Table 10: **Per-benchmark results for the rubric memory ablation.** Average columns report the mean over the four benchmarks; Section 4.4 discusses the averaged trends.

Method	Bamboogle			HotpotQA			MuSiQue			2Wiki			Avg.		
	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.
Qwen3-4B															
+ GRPO	61.6	69.8	71.0	44.9	54.5	57.4	17.0	26.9	26.5	60.9	68.3	67.6	46.1	54.9	55.6
+ RaR-style	63.0	70.0	72.3	45.1	54.3	57.7	16.9	25.7	27.0	60.2	68.8	70.5	46.3	54.7	56.9
+ w/o memory	64.3	72.5	74.6	45.5	55.1	58.9	18.4	27.1	28.3	62.8	71.0	72.6	47.8	56.4	58.6
+ ARBOR	65.1	73.0	76.2	45.4	55.6	59.7	17.0	25.7	27.8	64.2	72.4	74.6	47.9	56.7	59.6
Qwen3-8B															
+ GRPO	61.0	69.4	72.2	44.4	55.0	57.8	17.4	26.7	26.6	64.8	71.0	71.1	46.9	55.5	56.9
+ RaR-style	60.0	67.8	70.7	43.9	54.5	57.4	17.3	25.5	28.1	65.1	71.3	71.3	46.6	54.8	56.9
+ w/o memory	64.2	72.0	74.2	47.0	57.2	60.9	16.9	25.7	28.0	62.4	69.7	70.8	47.6	56.2	58.5
+ ARBOR	65.3	72.9	76.5	47.8	58.6	62.9	18.3	26.8	29.5	64.2	73.3	75.3	48.9	57.9	61.1
Qwen3-14B															
+ GRPO	61.9	71.5	72.8	48.1	58.4	62.3	19.4	29.9	31.6	69.8	76.2	76.4	49.8	59.0	60.8
+ RaR-style	59.8	68.8	69.4	45.3	55.5	58.7	17.2	26.9	27.8	66.8	71.6	70.9	47.3	55.7	56.7
+ w/o memory	64.5	73.8	76.0	48.7	58.3	62.4	18.4	27.3	27.5	67.9	74.4	74.3	49.9	58.5	60.1
+ ARBOR	67.5	75.8	78.6	48.6	59.1	63.6	19.1	28.7	31.6	69.8	76.3	77.3	51.3	60.0	62.8

RaR-style is our search-QA adaptation of Rubrics-as-Rewards (Gunjal et al., 2026), instantiated as a single-use instance-specific rubric reward. For each training query, the external rubric-generation LLM generates 5–10 reference-conditioned rubric items from only the question and ground-truth answer, without seeing sampled rollout trajectories. During RL, the judge model scores each full agent trajectory under the query-specific rubric set and returns a holistic 1–10 rating; we normalize it to $[0, 1]$, center it within the rollout group, and add it to the base F1/format reward with the same shaping coefficient as ARBOR. The generated rubrics are used only for their source query group and are never reused by later groups.

The w/o memory variant instead keeps ARBOR's contrastive process-rubric induction from sampled trajectories, but discards each induced rubric after scoring its source query-group. Thus both variants have no reusable memory, while differing in whether their rubrics are reference-conditioned instance criteria or contrastively induced process criteria.

C.5 Common vs. Query-Local Rubrics

We further examine whether newly induced query-local rubrics should be used directly for reward shaping. On Qwen3-8B, we compare three variants: *query-local only* is the existing w/o memory variant, which induces rubrics from the current query-group and discards them after scoring that group, without maintaining a reusable buffer; *common + query-local* computes the process reward using both the active common rubrics and all newly induced query-local rubrics; and *common only* is default ARBOR, which uses only active common rubrics for reward shaping. Common-only ARBOR still induces query-local drafts for admission and later consolidation, but does not use them directly as rewards.

Table 11: **Common and query-local rubric comparison on Qwen3-8B.** LLM-judge accuracy (%) across four benchmarks and their average. Query-local only corresponds to w/o memory, and common only is default ARBOR.

Variant	Bamboogle	HotpotQA	MuSiQue	2Wiki	Avg.
Query-local only	74.2	60.9	28.0	70.8	58.5
Common + query-local	69.6	60.0	28.8	73.0	57.9
Common only (ARBOR)	**76.5**	**62.9**	**29.5**	**75.3**	**61.1**

Table 11 shows that common-only ARBOR achieves the highest average accuracy (61.1), outperforming both query-local only (58.5) and common + query-local (57.9). The combined variant does not

improve over either source alone, supporting the design choice to filter and consolidate query-local drafts before using them for reward shaping.

C.6 Reuse Distribution

Figure 4 plots the cumulative share of scoring events as a function of the fraction of common rubrics ranked by reuse. The curve sits well above the even-distribution diagonal. The top 5% of rubrics cover 20% of all scoring events, and the top 20% cover 44%. This concentration arises from the active selection mechanism in Section 3.4, which preferentially activates rubrics with high cumulative correlation, so that a small core of validated rubrics accumulates most scoring events.

C.7 Test-Time Reuse of Learned Rubrics

We examine whether the reusable process rubrics learned during ARBOR training can also support test-time trajectory selection. Specifically, we freeze the final common-rubric buffer from the Qwen3-8B ARBOR training run and use it to score complete trajectories for Best-of-N selection.

We use the final consolidated common-rubric buffer without manually selecting or rewriting its contents. The buffer contains the following two reusable process criteria: (i) *Precision in Query Formulation Relative to Semantic and Structural Intent*, which evaluates whether search queries faithfully capture the question's semantic constraints and are refined according to evidence gaps; and (ii) *Evidence-Aware Synthesis with Hierarchical Conflict Resolution*, which evaluates whether the agent prioritizes convergent and authoritative evidence, resolves conflicts appropriately, and stops once the key uncertainty is sufficiently resolved.

For each question, we reuse the five independently sampled ARBOR trajectories from our original evaluation. The scorer, Qwen-plus, receives the question, the full observable trajectory, and the two frozen rubric descriptions, without access to the reference answer. Unlike the pairwise comparisons used during training, this experiment uses pointwise scoring: the scorer assigns an integer score from 0 to 10 under each rubric, and the mean of the two scores determines the trajectory's ranking.

We compare three inference settings. **Sample@1** reports the average accuracy of a single sampled trajectory across the five evaluation runs. For **Rubric Bo3**, we select the highest-scoring trajectory within each of the $\binom{5}{3} = 10$ three-candidate subsets for each question and average the resulting correctness across these subsets. **Rubric Bo5** selects the highest-scoring trajectory from all five candidates. The scorer's trajectory scores are reused across candidate subsets.

Table 12: **Test-time reuse of learned rubrics on Qwen3-8B.** Average LLM-judge accuracy (%) under rubric-guided trajectory selection.

Method	Bamboogle	HotpotQA	MuSiQue	2Wiki	Avg.
Sample@1	76.5	62.9	29.5	75.3	61.1
Rubric Bo3	79.0	65.8	32.2	81.5	64.6
Rubric Bo5	**79.2**	**67.0**	**33.5**	**82.5**	**65.6**

Table 12 shows that frozen-rubric selection improves over Sample@1 on all four benchmarks. Average accuracy increases from 61.1 to 64.6 with three candidates and to 65.6 with five candidates, yielding a **4.5** points improvement for Rubric Bo5 over Sample@1. Bo5 also improves over Bo3 on every benchmark. These results support the utility of the learned common rubrics for post-hoc selection among complete ARBOR trajectories, extending their use beyond training-time reward shaping.

C.8 Effect of the Rubric Reward Coefficient

We vary the rubric reward coefficient λ over $\{0.05, 0.1, 0.15, 0.2\}$ on Qwen3-8B while keeping the remaining settings fixed. Figure 5 reports average LLM-judge accuracy across the four benchmarks. The default coefficient $\lambda = 0.1$ achieves the highest average accuracy among the tested values (61.1), and all four settings outperform GRPO (56.9).

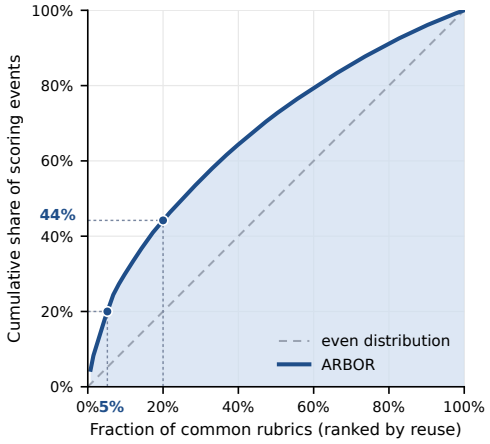


Figure 4: Cumulative share of scoring events covered by the top fraction of common rubrics, ranked by reuse count.

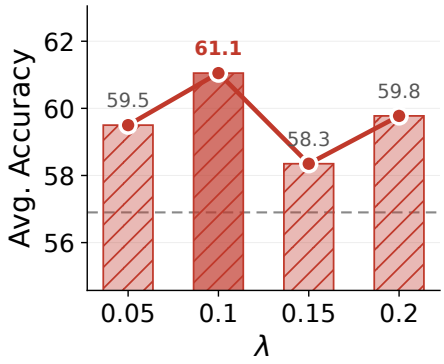


Figure 5: **Effect of the rubric reward coefficient λ on Qwen3-8B.** Average LLM-judge accuracy (%) across four benchmarks. The dashed line denotes GRPO; $\lambda = 0.1$ is the default setting.

D Consolidation Case Study

We illustrate the consolidation process with a concrete example. Table 13 shows three query-local candidate rubrics induced from different queries during RL training. Each candidate describes the same underlying process behavior, resolving the target entity with canonical naming before querying its attributes, but in different phrasings shaped by the specific query context.

The consolidation module receives these and other similar candidates as input, identifies the shared process pattern, and produces the common rubric *Precise Entity-Attribute Targeting with Canonical Framing*. The resulting common rubric unifies three variant phrasings of the same process standard into a single reusable criterion, while the query-specific framing of each candidate is abstracted away. This abstraction is what enables the common rubric to be reused across 101 query-groups throughout training.

E Prompts

This section reproduces the four prompts used in ARBOR. Placeholders inside braces are filled at runtime, and ellipses denote repeated structures over the corresponding lists.

E.1 Contrastive Induction

The induction prompt has a fixed instruction header followed by a per-call payload that fills in the question itself, the optional ground-truth answer, the contrast pairs, and the existing rubrics; the input fields are listed at the end of the header. The full header is reproduced in Table 14.

Table 14: **Contrastive induction prompt.** The prompt induces query-local process rubrics from contrastive trajectory pairs.

Contrastive Induction Prompt

You are evaluating a **Deep Search Agent** — a model that answers questions by iteratively searching the web, reading results, and synthesizing a final answer.
Each response below contains the agent's full trajectory: its reasoning, search queries issued, search results received, and the final answer. **Answer correctness is already scored separately via strict-format-gated F1.**

Contrastive Induction Prompt

Format compliance is part of trajectory quality. A response with malformed tags, missing or unclosed <think>/<search>/<result>/<answer> structure, multiple final answer blocks, missing boxed final answer, or truncation must not be treated as a high-quality positive example, even if a raw answer can be recovered and happens to match the ground truth.

Ideal Search Agent Vision. An excellent Search Agent is defined by the quality of its reasoning and decisions at each step. The ideal agent:

- **Plans deliberately**: Understands the question deeply, identifies what information is needed, and decomposes complex questions into clear sub-goals before searching.
- **Searches precisely**: Formulates targeted queries using specific entity names and relationships to retrieve the right information directly.
- **Searches efficiently**: Avoids redundant queries, unnecessary re-verification of already-established facts, and circular search patterns that waste the search budget without adding new information once the key uncertainty has been resolved.
- **Adapts intelligently**: Searches further when genuine ambiguity, missing constraints, or conflicting evidence remain; synthesizes and answers when sufficient evidence has resolved the key uncertainty.
- **Answers confidently**: Produces a well-reasoned, correctly formatted final answer once the question can be resolved.
- **Finalizes cleanly**: Uses the required tool-call and answer format correctly, keeps tags properly paired and closed, and transitions cleanly from search to a single final answer without leaving unfinished structure.

Your Goal. Identify **search strategy and reasoning process behaviors that CAUSE the agent to reach correct or incorrect answers**. The key question is: what did successful responses do differently in their *process* (not their final answer) that led them to succeed? A good rubric targets a process behavior that, if improved, would lead to more correct answers. A bad rubric merely re-checks whether the answer content is correct (which F1 already does).

What to Look For. Examine the provided contrast pairs and identify **specific process behaviors that caused some responses to succeed while others failed** (or that made some stronger responses clearly better-executed than weaker ones). Use the Ideal Search Agent Vision above as a reference for what good process looks like, but ground your rubrics in concrete observations from the given trajectories — not generic best practices. Each rubric should capture a behavior dimension where the given responses **actually differ**, so that scoring produces meaningful variance across samples.

Rules.

- **CRITICAL: Rubrics Must Target Process, Not Answer Content.** Do NOT generate rubrics that verify whether specific facts, names, dates, or relationships in the final answer are correct — this is exactly what F1 already measures.
- **Format as Process Quality.** Treat clean finalization and valid interaction structure as part of process quality. Format-invalid or truncated responses may be used as low-quality anchors when they reveal unfinished reasoning, invalid tool-use structure, or failure to produce a single parseable final answer. However, avoid generating rubrics that only duplicate the strict format checker; prefer criteria that connect format discipline with reasoning, evidence use, search control, and answer finalization.
- **Using Ground Truth.** If a Ground Truth Answer is provided, use it to: (1) understand what information the agent needed to find; (2) identify which responses succeeded vs. failed; (3) then analyze: what **search decisions** (query formulation, result interpretation, search direction) caused the successful responses to find the right information while others did not?
- **Discriminative Power.** Each rubric must produce actual score variance across the contrasted responses. Focus on observable differences in search behavior. If both sides exhibit the same search pattern, do not create a rubric for it.
- **Rubric Format.** Each rubric should describe BOTH what a high-scoring response does AND what a low-scoring response does for the same dimension. This helps the judge accurately place each response on the full spectrum.

Contrastive Induction Prompt

- **Balanced Closure Preference.** Do not treat extra verification as an intrinsic virtue. When the retrieved evidence is already sufficient to resolve the key uncertainty, continuing to search, continuing to doubt, or delaying the final answer should be treated as low-scoring behavior rather than "thoroughness." At the same time, do NOT reward premature closure. If important ambiguity, missing constraints, or conflicting evidence remain unresolved, stopping early and answering too soon should also be treated as low-scoring behavior.

- **Non-Redundancy.** Do not duplicate Existing Rubrics in meaning or scope. If Existing Rubrics already cover the quality differences, return empty lists.

Output. Generate 1–3 rubrics total. Fewer high-quality process rubrics are far better than many generic ones.

```
{ "rubrics": [ { "title": "<concise label>", "description": "<what a high-scoring response does>", "counter_description": "<what a low-scoring response does>" } ] }
```

Inputs.

1. **Question**: The original question the agent is trying to answer.
2. **Ground Truth Answer** (if provided): The correct answer — use to understand what information was needed and which search strategies led to finding it.
3. **Contrast Pairs**: One or two higher-target vs. lower-target trajectory comparisons.
4. **Existing Rubrics**: Previously established rubrics — do not duplicate these.

E.2 CONSOLIDATION

The consolidation prompt is invoked when processing of a rollout batch is complete. Existing common rubrics in $\mathcal{P}$ are passed in as a no-duplicate constraint, and the candidate drafts are grouped by query of origin. The full prompt is shown in Table 15.

Table 15: **Consolidation prompt.** The prompt distills recurring patterns from candidate rubrics into reusable common rubrics while avoiding duplicates.

Consolidation Prompt

You are a rubric consolidation expert. Your task is to distill specific, query-level scoring rubrics into **generalized cross-query standards** that effectively differentiate response quality levels across queries.

Core Principles

1. **Maximize Differentiation**: Each standard should create meaningful scoring differences between strong and weak responses — avoid standards that most responses trivially satisfy or all inevitably fail.
2. **Align with Sound Reasoning**: Standards must reflect the core elements of correct reasoning — logical coherence, evidence accuracy, and conclusion reliability — rather than penalizing minor stylistic differences or irrelevant details.
3. **Dual-Sided Standards**: Each standard should describe BOTH the desired behavior (what good responses do) AND the undesired behavior (what weak responses lack) for the same dimension.
4. **Balanced Closure**: Do not treat early stopping as a generic virtue. Reward timely closure only when the key uncertainty has already been resolved; do not reward premature closure while ambiguity, missing constraints, or conflicting evidence remain.

Task

Identify 1–2 NEW, high-value standards by **abstracting common patterns** from the candidates below.

CRITICAL: Do NOT produce any standard that overlaps in meaning with the Existing Common Standards listed below, even if phrased differently.

Consolidation Prompt

Abstraction Strategy

- Look for **recurring quality patterns** across multiple candidates — both strengths worth rewarding and weaknesses worth penalizing — then merge them into one general standard with both a DO (desired) and AVOID (undesired) description.
- The description must be **cross-query general**: applicable to any query, not tied to specific queries.
- Replace question-specific details with general language. Use parenthetical examples "(e.g., ...)" to clarify, but the core criterion must be broadly applicable.
  - × Too specific: "Correctly identifies that hydroxyzine pamoate is for anxiety while hydrochloride is for itching"
  - ✓ Generalized: "Accurately distinguishes functional or contextual differences between closely related entities (e.g., variant forms, sub-categories, similar-named concepts)"
  - × Too specific: "Fails to verify the founding year of MIT from the search results"
  - ✓ Generalized: "Fails to cross-verify key factual claims against retrieved evidence before drawing conclusions"
- Do NOT create standards that penalize trivial issues unrelated to reasoning quality or answer correctness — stay focused on what genuinely separates strong responses from weak ones.
- If the candidates are already well-covered by Existing Common Standards, return an empty list.

Existing Common Standards (already established — do NOT duplicate these):

- {existing_title_1}
  DO: {existing_description_1}
  AVOID: {existing_counter_description_1}
- {existing_title_2}
  DO: {existing_description_2}
  AVOID: {existing_counter_description_2}
- ...

New Candidates (specific rubrics from recent queries):

Query 1: {query_1}
- {candidate_title_1_1}
  DO: {candidate_description_1_1}
  AVOID: {candidate_counter_description_1_1}
- ...

Query 2: {query_2}
- {candidate_title_2_1}
  DO: {candidate_description_2_1}
  AVOID: {candidate_counter_description_2_1}
- ...

Output

```
{ "new_common_rubrics": [ { "title": "...", "description": "<what a high-scoring response does>", "counter_description": "<what a low-scoring response does>" } ] }
```

E.3 PAIRWISE JUDGE

Pairwise judging is invoked once per edge in the sparse comparison graph (Section 3.4). The system prompt fixes the win/tie/loss output format, and the user prompt fills in the question, both responses, and the criterion derived from the active common rubric. Order randomization is applied at the call site by swapping which response receives the A label. The full prompt is shown in Table 16.

Table 13: **Consolidation case study for reusable rubric memory.** Three query-local candidate rubrics from different queries describe the same entity-attribute targeting pattern in query-specific language, and the consolidation module abstracts them into a reusable common rubric.

Source query	Candidate rubric (title and description)
Did Ashok Chavan and Peppino Tirri share the same nationality?	**Query Precision via Canonical Entity Anchoring.** A high-scoring response formulates its first search query using the canonical, widely attested name format of the target entity rather than vague role descriptors, enabling direct retrieval of authoritative biographic identifiers.
What is the date of death of the performer of song Story Of A Life?	**Entity Disambiguation Before Attribute Search.** A high-scoring response first identifies and confirms the canonical entity using unambiguous, attribute-robust evidence before searching for biographical attributes like date of death—ensuring subsequent searches are grounded in the right person.
The university which David R. Marchant was a member of in 1994 is located in what city?	**Precision-First Entity Identification Before Attribute Verification.** A high-scoring response treats the target as a concrete, disambiguated artifact requiring authoritative identification before querying attributes, prioritizing queries that resolve the entity's canonical identity in early steps.
Consolidated common rubric: *Precise Entity-Attribute Targeting with Canonical Framing*	

Table 16: **Pairwise judge prompt.** The prompt compares two trajectories under one active common rubric and outputs a win, tie, or loss.

Pairwise Judge Prompt

System:
You are an expert judge. You will be given a question (in <question></question> tags) and two responses labeled Response A and Response B (in <response_a></response_a> and <response_b></response_b> tags). You will also be given a scoring criterion (in <criterion></criterion> tags).

Your task is to determine which response better satisfies the criterion. Consider the criterion carefully and compare both responses against it. Strictly judge only the given criterion, and do not use any considerations beyond that criterion.

Format compliance is part of trajectory quality when it affects the criterion. If one response is malformed, truncated, unparseable, or fails to produce a single clean final answer, that should count against it for criteria involving coherent reasoning, valid tool-use/search-result structure, complete synthesis, or clean finalization.

You MUST output JSON in exactly this format: `{"winner": "A"}` or `{"winner": "B"}` or `{"winner": "TIE"}`.
- Output `{"winner": "A"}` if Response A clearly better satisfies the criterion.
- Output `{"winner": "B"}` if Response B clearly better satisfies the criterion.
- Output `{"winner": "TIE"}` if BOTH responses perfectly satisfy the criterion, OR if BOTH fail it entirely, OR if they are virtually indistinguishable in quality regarding this specific criterion.

User:
<question>{query}</question>
<response_a>{response_a}</response_a>
<response_b>{response_b}</response_b>
<criterion>
High-scoring response:

Pairwise Judge Prompt

{rubric_description}

Low-scoring response:
{rubric_counter_description}
</criterion>

E.4 LLM-JUDGE ACCURACY EVALUATION

Final-answer accuracy on the four benchmarks is reported with an LLM judge that compares the predicted answer against the gold answer. The judge is invoked with the prompt in Table 17 and is configured to output the literal string Correct or Incorrect.

Table 17: **LLM-judge accuracy evaluation prompt.** The prompt judges whether a predicted final answer matches the gold answer.

LLM-Judge Accuracy Evaluation Prompt

Given a Question and its Golden Answer, verify whether the Predicted Answer is correct. The prediction is correct if it fully aligns with the meaning and key information of the Golden Answer. Respond with "Correct" if the prediction is correct and "Incorrect" otherwise.
Golden Answer may have multiple options, and matching any one of them is considered correct.

Question: {question}
Golden Answer: {labeled_answer}
Predicted Answer: {pred_answer}