MITIGATING ZERO-VARIANCE GROUPS VIA QUERY RECYCLING FOR AGENTIC SEARCH TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

The use of GRPO-style algorithms has become the standard strategy for training LLM search agents under outcome-only rewards. With these algorithms, a query contributes to parameter updates only when its rollout group mixes successes and failures, and both all-correct (too-easy) and all-incorrect (too-hard) groups are zero-variance and waste rollout cost. Existing approaches treat zero-variance as a static property and either discard or pre-filter such groups. We hypothesize and empirically validate that queries flip between zero-variance and signal-bearing states as the policy evolves during training. Building on this intuition, we propose a query recycling strategy which returns zero-variance groups to a mutable pool for future resampling, so that the effective training distribution co-evolves with the policy. Using the proposed technique, a 4B parameter model trained exclusively on synthetic data achieves 71.3 average Pass@1 across seven multi-hop QA benchmarks, outperforming prior systems trained on benchmark-derived supervision. Analysis of recycling patterns shows that recycled queries supply roughly three quarters of the effective batch by the end of training, with contributions split between recovery from policy improvement and policy drift.[1]

1 INTRODUCTION

Agentic search has emerged as a paradigm in which LLMs interleave reasoning with search calls to solve complex information-seeking tasks. Training LLMs as search agents increasingly relies on reinforcement learning (Jin et al., 2025a), with GRPO (Shao et al., 2024) or similar variants being the dominant algorithms. For each query in a training batch, the LLM policy produces a group of K rollouts, and advantages are computed from the reward distribution within the group. When considering outcome-only rewards, a query carries gradient only when its rollout group contains both successes and failures. Hence, queries which the policy always solves (too easy) or for which it always fails (too hard) provide no learning signal.

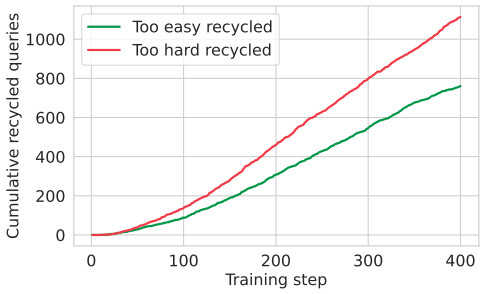


Figure 1: Recycling of zero-variance queries during RL training of a Qwen3-1.7B search agent. A query is recycled if, at some step t, its rollout group was too-easy (all correct) or too-hard (all incorrect), but when resampled at step $t' > t$ the same query was bearing signal.

Prior work treats zero-variance as a static property. For instance, DAPO (Yu et al., 2025) oversamples candidate groups per step until the effective batch is full, and discards the zero-variance candidates. ASearcher (Gao et al., 2025) carries the same approach into search agent training. GRESO (Zheng et al., 2025a) goes further by using past epoch zero-variance statistics to probabilistically skip prompts before rollout, motivated by the observation (in the math reasoning domain) that the zero-variance status is temporally consistent across epochs. The inherent shared assumption across these previous methods is that a query's zero-variance status persists across training steps.

[1] Code and data to be made available at a public GitHub repository.

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.

We instead hypothesize that zero-variance is a transient state, rather than a fixed property of the query. A query's zero-variance status reflects its success probability under the current policy, and the status shifts as the policy evolves. As the policy becomes more capable, previously too-hard queries may become partially solvable and start providing signal. Conversely, as the policy drifts during training, it may lose behaviors it had previously mastered, so that previously too-easy queries again produce mixed outcomes and are needed to counteract this forgetting.

In the LLM search agent setting, we observe that queries flip in and out of zero-variance during training. Figure 1 reports a Qwen3-1.7B run with recycling on a 10k-query pool, tracking queries that were zero-variance at some step t and signal-bearing on resample at a later step $t' > t$. By the time of pool saturation, roughly 20% of unique queries were successfully recycled (i.e., went from zero-variance to signal-bearing). Discarding zero-variance groups, or pre-filtering based on past statistics, eliminates this signal.

We formalize this intuition through dynamic pool management. We instantiate the training pool as a set of weighted queries from which we sample with probability proportional to the weights at each training step. The weights of sampled queries are then updated based on whether they produced gradient signal. Our approach, *query recycling*, is the specific update rule that zeroes the weight of consumed signal-bearing queries while keeping zero-variance ones eligible for resampling. The same framework subsumes GRPO and DAPO as alternative weight-update rules over the same pool.

We conduct experiments in a fully synthetic and sandboxed training setup built on DeepResearch-Gym (Coelho et al., 2025), without relying on the training splits of the considered evaluation benchmarks. We evaluate against standard GRPO and DAPO variants, training under matched rollout-budget regimes, and we find that recycling zero-variance queries is more effective than epoch-based replay of the training pool. Analysis further shows that recycled queries become a dominant source of effective training signal during later stages of GRPO training, with roughly three-quarters of accepted groups originating from previously zero-variance regions of the pool by the end of training. Despite training exclusively on synthetic data, our 4B agent achieves 71.3 average Pass@1 across seven multi-hop QA benchmarks, outperforming prior systems, while a 1.7B variant reaches 66.0, remaining competitive with larger search agents trained with benchmark-derived supervision.

2 Related Work

Search Agent Training. Beyond test-time scaling, which allocates additional inference-time compute (Snell et al., 2024; Muennighoff et al., 2025; Murali et al., 2026), Search-R1 (Jin et al., 2025b) established the now-standard paradigm of training LLM search agents with GRPO (Shao et al., 2024) under sparse outcome-only rewards, interleaving reasoning with multi-turn retrieval. Concurrent work followed the same recipe at various scales and model families (Jin et al., 2025a; Song et al., 2025; Zheng et al., 2025b), and outcome-based GRPO has since become a common starting point for search agent post-training.

Several studies subsequently proposed to augment or replace the sparse outcome signal with intermediate supervision to improve credit assignment and guide search behavior over long trajectories. SmartSearch (Wen et al., 2026) introduces a dual-level process reward that scores each query for novelty and usefulness, while Fathom-DeepResearch (Singh et al., 2025) uses a step-level reward that classifies tool calls according to cognitive behavior and marginal utility. Information Gain-based Policy Optimization (IGPO) (Wang et al., 2026) takes a different approach, deriving dense turn-level rewards from the marginal increase in the policy's probability of producing the correct answer. These approaches provide finer-grained supervision, but require additional reward definitions or auxiliary evaluation mechanisms to assign credit to intermediate search steps.

BehaviorPrime (Jin et al., 2026) instead follows prior work that uses SFT on trajectories generated by stronger teacher models to improve search agent performance (Zeng et al., 2024; Wu et al., 2025a). It further filters teacher trajectories to preferentially retain desirable search behaviors, arguing that this can reduce reliance on dense process rewards and their associated reward-design challenges.

Training with live search APIs introduces additional cost and reproducibility concerns. ZeroSearch (Sun et al., 2025a) addresses these by fine-tuning an LLM to simulate a search engine, while DeepResearchGym (Coelho et al., 2025) provides a static ClueWeb22 snapshot as a reproducible retrieval environment and shows that agents trained in this environment can generalize to

live web search at inference. Our experimental protocol builds on DeepResearchGym and extend its synthetic query generation protocol.

Zero-Variance Mitigation Strategies. Outcome-only GRPO can produce zero-variance groups when all rollouts for a query receive identical rewards, causing group-normalized advantages to collapse to zero and yielding no policy-gradient signal despite incurring rollout cost (Yu et al., 2025). DAPO (Yu et al., 2025) addresses this through dynamic sampling, discarding zero-variance groups after rollout and sampling additional data until the target number of informative groups is reached.

Subsequent methods explore alternative ways of avoiding or exploiting these groups. GRESO (Zheng et al., 2025a) predicts which prompts are likely to produce zero-variance groups using statistics from previous epochs and skips them before rollout. RL-ZVP (Le et al., 2025), in contrast, retains zero-variance groups but assigns their rollouts shaped advantages based on the group's outcome and token entropy.

In the search-agent setting, ASearcher (Gao et al., 2025) applies DAPO-style post-rollout filtering, paying the rollout cost before discarding zero-variance queries. DAVID-GRPO (Han et al., 2026) targets all-failed zero-variance groups by resampling forward from partially sound trajectories, assuming that the current policy may solve the query but failed by chance. In contrast, our approach recycles zero-variance queries directly, allowing them to be sampled again in subsequent training steps without changing the reward function or requiring additional rollout construction.

Synthetic Data for Search Agents. Recent work synthesizes training data for LLM search agents, often distilling trajectories from stronger reasoning models (Sun et al., 2025b; Wu et al., 2025a). Several methods instead target structurally challenging tasks. For instance, WebSailor (Li et al., 2025) samples and fuzzifies entity graphs, while WebShaper (Tao et al., 2025) composes complex QA samples through set-theoretic constructions. In turn, WebExplorer (Liu et al., 2025) and ASearcher (Gao et al., 2025) obfuscate identifying details from factual questions to force longer search chains. WebResearcher (Qiao et al., 2025) escalates complexity through tool augmentation, while ORBIT (Thakur et al., 2026) enforces answer verifiability through self and external checks.

3 Query Recycling for Search Agents

For our study, we consider the ReAct-style search agent design of AReal's Tongyi DeepResearch implementation (Fu et al., 2025; Li et al., 2026; Yao et al., 2023). We define a `search` tool, which allows the model to issue up to three parallel queries per turn. This tool directly returns cleaned text passages from relevant documents to the agent. The agent iteratively searches and reasons over the retrieved results until it produces a final answer or reaches a predefined step budget.

The `search` tool is backed by two retrieval sources depending on the setting. During training, queries are served by the DeepResearchGym API (Coelho et al., 2025; Ning et al., 2026) over a ClueWeb22 document snapshot (Overwijk et al., 2022), which gives a reproducible and sandboxed retrieval environment. During inference, we replace the backend used during training with SERPER (Serper, 2025) live web search, allowing us to test whether policies trained in DeepResearchGym can transfer to live web retrieval.

The search agent training is then done in two phases. First, the base model goes through an SFT phase to expose the model to search trajectories (Section 3.1). Then, we apply our proposed RL method, which recycles queries with zero-variance groups for future sampling (Section 3.2).

3.1 Supervised Fine-Tuning on Search Trajectories

We first apply a short supervised fine-tuning stage to provide a suitable initialization for reinforcement learning. Starting directly from the base model produces mostly invalid or unproductive search trajectories, making it difficult for GRPO to learn useful agentic behaviors from task rewards. We therefore warm-start the policy with SFT before applying reinforcement learning, following prior work (Jin et al., 2026). We run the previously described agent with a GPT-4.1-mini backbone as the teacher, and collect one trajectory per query. We retain trajectories that follow the required action format and end with the correct answer for SFT. The same SFT checkpoint is used as the starting point for every configuration in our experiments, isolating the effect of the RL strategy.

3.2 Reinforcement Learning with Query Recycling

Standard GRPO. We train the SFT checkpoint with GRPO following the AReaL implementation (Fu et al., 2025). Given a query q, the behavior policy produces a group of K rollouts, $G_t(q) = \{o_1, \ldots, o_K\}$, where t denotes the training step at which the group is sampled. Each rollout receives a binary outcome reward $r_i(q) = R(o_i, q) \in \{0, 1\}$, indicating whether the final answer is correct. We then compute group-normalized rollout advantages from the outcome rewards:

$$A_i(q) = \frac{r_i(q) - \mu_G}{\sigma_G + \epsilon_{\text{norm}}}, \qquad \mu_G = \frac{1}{K}\sum_{j=1}^{K} r_j(q),$$

where σ_G is the standard deviation of the rewards in the group, and ϵ_{norm} is a small numerical constant. The rollout-level advantage $A_i(q)$ is assigned to every valid token in rollout o_i. The policy is optimized with the following objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{q,G}\left[\frac{1}{\sum_{i=1}^{K} L_i}\sum_{i=1}^{K}\sum_{\tau=1}^{L_i} \min\left(\rho_{i,\tau}\, A_i, \quad \text{clip}(\rho_{i,\tau}, 1-\epsilon, 1+\epsilon)\, A_i\right)\right],$$

where L_i is the number of valid tokens in the rollout o_i, and where the per-token importance ratio $\rho_{i,\tau}$ between the current policy π_θ and the behavior policy $\pi_{\theta_{old}}$ is defined as follows:

$$\rho_{i,\tau} = \frac{\pi_\theta(o_{i,\tau} \mid q, o_{i,<\tau})}{\pi_{\theta_{\text{old}}}(o_{i,\tau} \mid q, o_{i,<\tau})}.$$

Dynamic Query Pool Management. Under the previous setting, a query q contributes gradient signal only when its rollout group contains both successful and unsuccessful trajectories (i.e., $0 < \sum_{i=1}^{K} r_i(q) < K$). Let $p_t(q)$ denote the success probability of query q at training step t:

$$p_t(q) = \mathbb{E}_{o\sim\pi_{\theta_t}(\cdot|q)}\left[R(o, q)\right]. \tag{1}$$

Assuming the K rollouts in a group are conditionally independent given q, the probability that the group is zero-variance is given by:

$$P_{\text{zv}}(q; \pi_{\theta_t}) = p_t(q)^K + (1 - p_t(q))^K. \tag{2}$$

Evaluating a query with a finite rollout group therefore yields a stochastic observation of whether the query is zero-variance under the current policy. Observing a zero-variance group at step t does not imply that the query is intrinsically uninformative, nor that it will remain so under future policies. Since $P_{\text{zv}}(q; \pi_{\theta_t})$ depends on the evolving policy, the zero-variance status is fundamentally non-stationary. A query observed as all-failed under an earlier policy may later become partially solvable as capabilities improve, while a previously all-correct query may re-enter the signal-bearing region as the policy drifts. We exploit this non-stationarity through dynamic query pool management, formalizing the training pool as a set of N weighted queries:

$$\mathcal{Q}^{(t)} = \{(q_i, w_i^{(t)})\}_{i=1}^{N}, \qquad w_i^{(0)} = 1,$$

where $w_i^{(t)} \in \{0, 1\}$ is the sampling weight of q_i at step t, which is updated iteratively. At each training step, we draw a candidate set C_t of kB distinct queries from $\mathcal{Q}^{(t)}$, with each query drawn with probability proportional to $w^{(t)}$, where B is the training batch size and $k \geq 1$ is an oversampling factor. We then generate K rollouts per candidate under π_{θ_t} and identify $\tilde{S}_t$, i.e. the set containing all the signal-bearing queries:

$$\tilde{S}_t = \left\{ q \in C_t \;\middle|\; 0 < \sum_{i=1}^{K} r_i(q) < K \right\}. \tag{3}$$

From $\tilde{S}_t$, we form the gradient update batch $S_t \subseteq \tilde{S}_t$ of size $\min(|\tilde{S}_t|, B)$, so that the effective batch size never exceeds B. Different training strategies correspond to different choices of the oversampling factor k and different update rules for the query weights $w_i^{(t)}$.

Query Recycling. Our strategy zeroes the weight of the consumed signal-bearing queries S_t, effectively removing them from the pool:

$$w_q^{(t+1)} = \begin{cases} 0 & \text{if } q \in S_t \ , \\ w_q^{(t)} & \text{otherwise} \ . \end{cases}$$

Queries that do not contribute gradient signal, including those with zero-variance groups, as well as excess signal-bearing candidates not selected into S_t, remain eligible for sampling under future policies. Thus, the effective training distribution co-evolves with the policy π_{θ_t} rather than following a fixed multi-epoch traversal of the query pool.

Oversampling and recycling address complementary inefficiencies. Increasing k raises the probability of observing signal-bearing queries within a training step, while recycling reallocates future sampling mass toward queries that have not yet contributed gradient updates. We use both mechanisms jointly throughout our experiments.

Connections to Existing Methods. The proposed method is in fact a generalization of existing methods. Both GRPO and DAPO can be formalized within this framework, arising from different choices of k and weight updates. For example, GRPO fixes $k = 1$ and consumes all sampled candidate queries (C_t), irrespective of gradient signal:

$$w_q^{(t+1)} = \begin{cases} 0 & \text{if } q \in C_t \ , \\ w_q^{(t)} & \text{otherwise} \ . \end{cases}$$

When $\sum_{i=1}^{N} w_i^{(t)} = 0$, the pool is reset by setting $w_i \leftarrow 1$ for all queries, recovering standard multi-epoch iteration over a static dataset.

DAPO (Yu et al., 2025) dynamically increases k within a training step until $|\tilde{S}_t| = B$. All sampled candidates are then consumed using the same update rule as GRPO. This guarantees a full effective batch at the cost of unbounded rollout computation per step. If we bound the oversampling factor k in DAPO (which is common in frameworks such as VeRL (Sheng et al., 2025) and NeMO-RL (NVIDIA, 2025)), we avoid the unbounded resampling loop of DAPO while allowing the effective batch size to vary when $|\tilde{S}_t| < B$. The weight update is otherwise identical to GRPO.

4 Experimental Setup

This section starts by providing an overview of the synthetic training queries that were generated for both supervised fine-tuning and reinforcement learning (Section 4.1), afterwards establishing the evaluation datasets, baselines, and evaluation protocol (Section 4.2).

4.1 Training Data and Details

We generate a synthetic pool of 25k multi-hop search queries grounded in ClueWeb22, providing a controlled training environment for agentic search. The pool combines 9k query-answer pairs from DeepResearchGym (Coelho et al., 2025) with 16k newly generated queries from three generators, each targeting a different retrieval strategy: *webgraph* queries chain documents along a random walk over the hyperlink graph, *iterative-search* queries extend chains through dense retrieval beyond hyperlink adjacency, and *comparative* queries require retrieving several independent entities before aggregating over them.

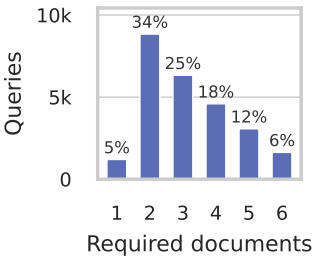


Figure 2: Distribution of the synthetic query pool according to the number of required supporting documents.

An LLM-based validation stage discards questions that are malformed or answerable from common knowledge, and the resulting queries require between one and six supporting documents (distribution depicted in Figure 2). We split this pool into 15k queries for supervised fine-tuning and 10k disjoint queries for reinforcement learning. Generation details, representative examples, and per-source training dynamics are given in Appendix A.

As detailed in Section 3, we train Qwen3-1.7B and Qwen3-4B independently in two stages. First, we perform Supervised Fine-Tuning (SFT) on synthetic search trajectories to instill baseline retrieval behaviors. Then, we apply the Reinforcement Learning (RL) procedure described in Section 3.2, with batch size $B = 32$, group size $K = 4$, and oversampling factor $k \in \{1, 2\}$. Setting $k = 1$ without recycling reduces to standard **GRPO**, while $k = 2$ without recycling corresponds to **Bounded-DAPO**. Our full method uses $k = 2$ with recycling. All configurations share a matched rollout budget of 84.5k trajectories, with $k = 1$ executed for 660 gradient steps and $k = 2$ for 330 steps. For the non-recycling baselines, this is equal to two epochs over the entire training pool.

4.2 Resources and Evaluation Methods

We evaluate agent performance using the BehaviorPrime setup (Jin et al., 2026). For multi-hop question answering, we use 2WikiMultiHopQA (Ho et al., 2020), Bamboogle (Press et al., 2023), HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022), Natural Questions (Kwiatkowski et al., 2019), PopQA (Mallen et al., 2023), and TriviaQA (Joshi et al., 2017). For web navigation, we consider GAIA (Mialon et al., 2024), HLE (Phan et al., 2025), and WebWalkerQA (Wu et al., 2025b). We use GPT-4.1-mini as the judge for all benchmarks.

Inference uses Qwen3 (Yang et al., 2025) default decoding parameters (temperature 0.6, top-p 0.95, and top-k 20). We report Pass@1 averaged over eight runs. We compare against SearchR1 (Jin et al., 2025b), DeepResearcher (Zheng et al., 2025b), R1Searcher (Song et al., 2025), BehaviorPrime (Jin et al., 2026), ASearcher-Web-7B (Gao et al., 2025), ORBIT (Thakur et al., 2026), and SmartSearch (Wen et al., 2026). For the first four, we report the results from the BehaviorPrime study, which evaluates these methods using the same LLM-judge protocol we adopted. For ASearcher-Web-7B, ORBIT, and SmartSearch, we use their publicly released code and model weights to run inference on the same dataset splits considered in our experiments, using SERPER as the search engine, and evaluate the resulting answers with the same LLM-judge protocol.

5 Results and Discussion

This section first compares query recycling against prior agentic search systems and matched-budget RL baselines, on seven multi-hop QA benchmarks and on harder long-horizon benchmarks (Section 5.1). We then analyze how recycling shapes training, examining the composition of the effective batch and query pool, the role of group size, and transfer across LLM backbones (Section 5.2).

5.1 Empirical Results

Agentic Search Results. Table 1 reports Pass@1 results on the seven MHQA datasets. Our full pipeline reaches 71.3 average Pass@1 with Qwen3-4B, outperforming prior systems trained on benchmark-derived supervision despite using only synthetic training queries. The 1.7B variant reaches 66.0 average, remaining competitive with substantially larger search agents. Our setup transfers zero-shot along two axes: from synthetic training queries to different benchmark distributions, and from the sandboxed DeepResearchGym retrieval environment used during training to live

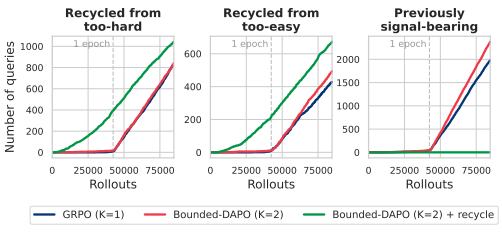


Figure 3: Cumulative queries seen during training on Qwen3-1.7B, decomposed by initial state: recycled from too-hard (left), recycled from too-easy (middle), and revisits to previously signal-bearing queries (right).

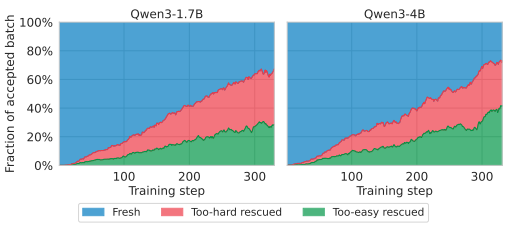


Figure 4: Composition of the effective batch (i.e. signal-bearing, S_t) over training steps on Qwen3 1.7B and 4B. *Fresh* queries are drawn from the pool for the first time.

Model	Size	k	Rollouts	2Wiki	Bamb	HQA	MSQ	NQ	PopQA	TriQA	Avg
Prior Work (training data includes MHQA benchmark splits)											
Search-R1	7B	–	–	47.9	57.6	63.0	27.5	60.0	47.0	76.2	54.2
R1-Searcher	7B	–	–	65.8	65.6	53.1	25.6	52.3	43.4	79.1	55.0
DeepResearcher	7B	–	–	66.6	72.8	64.3	29.3	61.9	52.7	85.0	61.8
SmartSearch	3B	–	–	78.7	74.4	64.1	35.4	67.2	51.6	85.7	65.3
BehaviorPrime	1.7B	–	–	73.8	70.4	67.0	30.7	73.6	**56.6**	86.9	65.6
ORBIT	4B	–	–	75.2	74.4	65.6	35.9	68.4	55.0	86.3	65.8
ASearcher-Web	7B	–	–	76.4	71.2	68.0	36.1	70.1	54.7	87.5	66.3
Ours (full synthetic training over ClueWeb22 and evaluation using SERPER)											
Qwen3	1.7B	–	–	37.4	44.4	44.1	16.3	54.5	41.8	68.6	43.9 ± 0.40
+ SFT	1.7B	–	–	54.8	47.2	54.4	24.6	67.6	52.0	86.0	55.2 ± 0.33
+ GRPO	1.7B	1	84.5k	70.9	68.8	66.6	36.9	69.9	54.4	86.9	64.9 ± 0.21
+ B-DAPO (no recycle)	1.7B	2	84.5k	69.5	67.2	66.9	36.9	69.5	53.5	85.2	64.1 ± 0.20
+ B-DAPO (recycle too-easy)	1.7B	2	84.5k	68.9	73.2	66.7	37.6	71.5	53.6	87.6	65.6 ± 0.27
+ B-DAPO (recycle too-hard)	1.7B	2	84.5k	69.1	70.4	66.1	37.3	70.7	53.7	87.0	64.9 ± 0.32
+ B-DAPO (recycle all)	1.7B	2	84.5k	71.7	72.4	67.7	38.0	70.1	54.2	87.8	66.0 ± 0.17
Qwen3	4B	–	–	57.9	61.2	57.6	29.8	72.0	55.6	84.2	59.8 ± 0.24
+ SFT	4B	–	–	68.9	64.4	64.5	33.8	73.1	55.7	88.5	64.1 ± 0.19
+ GRPO	4B	1	84.5k	77.7	77.2	73.0	40.7	69.9	53.1	89.1	68.7 ± 0.17
+ B-DAPO (no recycle)	4B	2	84.5k	77.4	77.7	72.4	**41.8**	73.7	54.2	**90.8**	69.7 ± 0.17
+ B-DAPO (recycle all)	4B	2	84.5k	**79.1**	**84.0**	**74.4**	41.3	**74.4**	55.8	89.9	$\mathbf{71.3} \pm 0.20$

Table 1: Pass@1 results over seven different multi-hop QA benchmarks. All our RL configurations start from the SFT checkpoint and use a matched rollout budget, equivalent to training for two epochs over the query pool for the GRPO and DAPO variants. Averaged results over multiple runs are reported as mean $\pm$ standard error.

SERPER search at inference time. At a matched rollout budget, query recycling improves over both two-epoch GRPO and Bounded-DAPO. To isolate the contribution of each tail of the zero-variance distribution, we additionally train the 1.7B model while recycling only too-hard or only too-easy queries. Both partial variants improve over Bounded-DAPO without recycling, but recycling both tails performs best overall.

Recycling Mechanism. Multi-epoch GRPO and Bounded-DAPO follow a fixed revisiting schedule: in a second epoch every query is sampled again, regardless of whether it produced signal in the first pass. Recycling replaces this with a status-dependent allocation, concentrating subsequent rollouts on queries that previously produced zero-variance groups. Figure 3 decomposes the unique queries each method extracts signal from according to their initial state. The inflection at one epoch separates the two regimes. Without recycling, the second epoch is dominated by revisits, i.e. both GRPO and Bounded-DAPO accumulate rollouts on previously consumed queries. Recycling never revisits and instead recovers additional too-hard and too-easy queries under the same total rollouts.

Long-horizon Reasoning. Going beyond multi-hop question answering, Table 2 shows similar trends on harder web-navigation and long-horizon reasoning benchmarks. Evaluating our recycle-based 1.7B and 4B Qwen models, we see that performance remains competitive despite the fact that training relied exclusively on English synthetic data, and evaluation used open-web environments. The main exception is WebWalker, whose multilingual setting significantly differs from our English-only training distribution.

Model	k	Recycle	GAIA	HLE	WW
Prior work					
BehaviorPrime-1.7B	–	–	21.4	7.8	**37.2**
SmartSearch-3B	–	–	19.4	9.8	26.8
ASearcher-Web-7B	–	–	22.3	7.0	21.5
Ours (Qwen3 + SFT + B-DAPO w/ recycling)					
1.7B	2	✓	21.4	8.0	35.4
4B	2	✓	**24.8**	**10.4**	36.3

Table 2: Pass@1 on GAIA, HLE, and WebWalker.

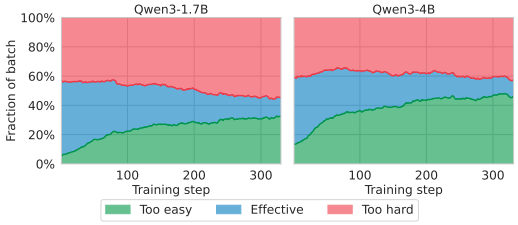


Figure 5: Composition of the sampled batch (i.e. all candidates, C_t) under recycling, for Qwen3-1.7B (left) and Qwen3-4B (right).

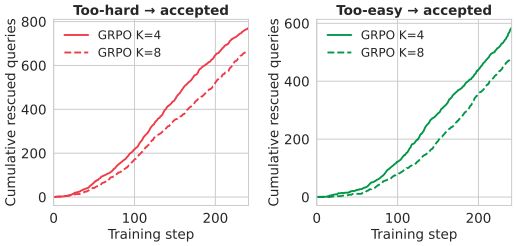


Figure 6: Cumulative distribution of recycled queries during the training of Qwen3-1.7B at GRPO group sizes of K=4 and K=8.

5.2 Ablations and Analysis

This section examines how query recycling shapes training. We analyze the composition of the effective batch and of the sampled pool over training, how GRPO group size affects zero-variance classification, and how the gains transfer across LLM backbones. Further analyses are deferred in appendix: synthetic query generation and per-query-type training dynamics (Appendices A and B), a relaxation of the binary query weighting scheme (Appendix C), recycling under process rewards (Appendix D), and the robustness of the LLM judge (Appendix E).

Recycled Queries Dominate the Effective Batch. Figure 4 shows the composition of the effective training batch during recycling-based training. Early in training, most signal-bearing groups come from freshly sampled queries. As training progresses, recycled queries increasingly dominate the effective batch. By the final steps, rescued too-hard and too-easy queries together account for three quarters of accepted groups. Both rescue populations grow throughout training and neither saturates within our horizon, suggesting that informative queries continue to emerge from previously zero-variance regions. Too-hard and too-easy rescues appear at comparable scale, indicating that recycling captures delayed capability acquisition, and re-entry from previously mastered behavior.

Pool Difficulty Evolves with Model Capacity. Figure 5 tracks the composition of the sampled query pool under recycling throughout training. For the Qwen3-1.7B model, the too-easy fraction increases steadily to approximately 35%, while the effective fraction gradually declines, and the too-hard region remains dominant throughout training. This suggests that the smaller Qwen3 model converts part of the pool into mastered queries, but model training still leaves a large fraction of the queries persistently beyond the model's capability.

In contrast, the 4B model exhibits a different dynamic, with a relatively steady too-hard fraction, and a too-easy one growing to nearly 50%. These trends are consistent with a stronger policy progressively converting initially unsolved queries into solvable and eventually mastered ones, producing a clear migration towards the too-easy regions of the pool. This suggests that zero-variance difficulty is policy-relative. Under an otherwise identical training setup, the larger model converts a substantial fraction of previously unsolved queries into effective or mastered ones. This further indicates that the too-hard pool reflects capability limitations rather than a synthetic sampling noise floor.

Group Size as a Difficulty Estimator. The per-group rollout count K controls both rollout cost and the granularity of zero-variance classification. From Equation 2, larger K concentrates the zero-variance pool around the extreme tails of the difficulty distribution, while smaller K admits more intermediate-difficulty queries that happen to produce all-same outcomes under finite sampling.

Figure 6 reflects this effect empirically. Compared to K=8, the noisier K=4 setting produces approximately 17% more too-hard rescues and 23% more too-easy rescues, consistent with more queries being temporarily assigned to zero-variance regions due to sampling noise, and later re-entering the effective set after resampling. Despite this noisier classification, the final averaged accuracy over the 7 MHQA datasets at K=4 matches K=8 at half the rollout cost, making K=4 the more favorable compute trade-off in our setting.

Model	Size	k	Rollouts	2Wiki	Bamb	HQA	MSQ	NQ	PopQA	TriQA	Avg
Ours (full synthetic training over ClueWeb22 and evaluation using SERPER)											
Qwen3	4B	–	–	57.9	61.2	57.6	29.8	72.0	55.6	84.2	59.8 ± 0.24
+ SFT	4B	–	–	68.9	64.4	64.5	33.8	73.1	55.7	88.5	64.1 ± 0.19
+ B-DAPO (no recycle)	4B	2	84.5k	77.4	77.7	72.4	**41.8**	73.7	54.2	**90.8**	69.7 ± 0.17
+ B-DAPO (recycle all)	4B	2	84.5k	**79.1**	**84.0**	**74.4**	41.3	**74.4**	55.8	89.9	**71.3** ± 0.20
LlaMA3.2	3B	–	–	39.7	38.2	41.4	17.6	59.6	45.8	70.1	44.6 ± 0.14
+ SFT	3B	–	–	61.5	64.4	63.5	33.0	72.4	56.4	88.3	62.7 ± 0.21
+ B-DAPO (no recycle)	3B	2	84.5k	71.0	76.0	68.7	36.9	70.0	55.2	86.8	66.4 ± 0.17
+ B-DAPO (recycle all)	3B	2	84.5k	70.9	75.6	71.5	40.0	72.7	55.2	89.7	67.9 ± 0.20
SmolLM3	3B	–	–	26.6	25.2	22.5	10.3	23.9	20.5	33.8	23.3 ± 0.20
+ SFT	3B	–	–	60.7	62.8	63.4	34.6	70.0	53.7	87.6	61.8 ± 0.14
+ B-DAPO (no recycle)	3B	2	84.5k	70.7	76.8	69.0	37.9	68.1	52.0	86.2	65.8 ± 0.18
+ B-DAPO (recycle all)	3B	2	84.5k	70.5	77.2	70.6	36.2	70.8	53.2	88.1	66.7 ± 0.21

Table 3: Pass@1 results over seven multi-hop QA benchmarks, comparing multiple LLM backbones (Qwen3, LlaMA3.2, and SmolLM3). All RL configurations start from the SFT checkpoint and use a matched rollout budget. Averaged results over multiple runs are reported as mean ± standard error.

Query Recycling Under Different LLM Backbones. The previous results focus on Qwen3 models of two different sizes. To assess whether our findings generalize across model families, Table 3 compares Qwen3-4B with Llama 3.2-3B (Grattafiori et al., 2024) and SmolLM3-3B (Bakouch et al., 2025), all trained with the same recipe. Query recycling improves over Bounded-DAPO without recycling for every backbone. It is also worth noting that the training pipeline narrows the gap between backbones considerably. For instance, SmolLM3 starts 36.5 points below Qwen3 but ends within 4.6 points after SFT and recycling-based RL. Qwen3-4B remains the strongest backbone overall, although it is also the largest model in this comparison.

6 Conclusions and Future Work

In this paper, we studied query recycling as a train-time modification for LLM search agents, returning queries with zero-variance rollout groups to the active data pool for resampling at later training steps. We paired this training strategy with a fully synthetic training pipeline, where queries are grounded in ClueWeb22 and training runs in a sandboxed retrieval environment, with no access to benchmark training splits. A Qwen3-4B agent trained with this approach reaches 71.3% average Pass@1 across seven multi-hop QA benchmarks, outperforming prior systems trained on benchmark-derived supervision while transferring zero-shot to live web search at inference time. A Qwen3-1.7B variant reaches 66.0% and remains competitive with larger search agents.

Our analysis shows that the zero-variance status is policy-relative across training steps and model scales. The 4B model converts close to half of the pool into too-easy queries by the end of training, while the 1.7B model leaves the too-hard region dominant throughout. Too-easy queries account for 40% of all rescues, so recoverable signal arrives from policy drift alongside policy improvement. Methods that intervene only on all-failed groups would miss this contribution.

Query recycling is also compatible with denser supervision signals and with advantage shaping over zero-variance groups. For future work, we plan to study these combinations in detail. A fixed pool also constrains the training horizon (i.e., as the policy improves, the signal-bearing fraction of the pool shrinks). Generating new queries online, matched to the policy's current capability, would relax this constraint. Although we study search agents, query recycling only assumes group-normalized advantages over a pool of prompts, and could in principle be applied to other domains with verifiable rewards, such as code generation. Whether zero-variance status is similarly transient in these domains remains an open question. Yet another direction for future work concerns extending recycling to deep research systems that perform long-form report generation, where reward design becomes the harder problem.

AI USE STATEMENT

In this work, we used generative AI tools for synthetic data generation and code generation. Additionally, we used generative AI tools for improving the readability of the manuscript. We have reviewed all AI-assisted work. Specifically, code quality was enforced by reviewing and testing, and the manuscript was revised by all authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

By using large pre-trained language models, we acknowledge the risks associated with the presence of inherent biases embedded within the models, which may inadvertently perpetuate or amplify societal biases present in the training data.

REPRODUCIBILITY STATEMENT

All the evaluation benchmarks and base LLMs used in our experiments are publicly available for research use. We will open-source the code that allows for reproduction of the results, as well as model checkpoints and synthetic data.

REFERENCES

Elie Bakouch et al. SmolLM3: Smol, multilingual, long-context reasoner, 2025. URL https://huggingface.co/blog/smollm3. Accessed September 16, 2026.

João Coelho, Jingjie Ning, Jingyuan He, Kangrui Mao, Abhijay Paladugu, Pranav Setlur, Jiahe Jin, Jamie Callan, João Magalhães, Bruno Martins, and Chenyan Xiong. DeepResearchGym: A Free, Transparent, and Reproducible Evaluation Sandbox for Deep Research. *ArXiv*, abs/2505.19253, 2025.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. AReaL: A Large-Scale Asynchronous Reinforcement Learning System for Language Reasoning. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond Ten Turns: Unlocking Long-Horizon Agentic Search with Large-Scale Asynchronous RL. *ArXiv*, abs/2508.07976, 2025.

Aaron Grattafiori et al. The Llama 3 Herd of Models. *ArXiv*, abs/2407.21783, 2024.

Hojae Han, Heeyun Jung, Jongyoon Kim, and Seung-won Hwang. Can David Beat Goliath? On Multi-Hop Reasoning with Resource-Constrained Agents. *ArXiv*, abs/2601.21699, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *International Conference on Computational Linguistics (COLING)*, 2020.

Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan Ö. Arik, and Jiawei Han. An Empirical Study on Reinforcement Learning for Reasoning-Search Interleaved LLM Agents. *ArXiv*, abs/2505.15117, 2025a.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. *ArXiv*, abs/2503.09516, 2025b.

Jiahe Jin, Abhijay Paladugu, and Chenyan Xiong. Beneficial Reasoning Behaviors in Agentic Search and Effective Training Methods to Obtain Them. In *Findings of the Annual Meeting of the Association for Computational Linguistics, (ACL)*, 2026.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Annual Meeting of the Association for Computational Linguistics, (ACL)*, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. 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. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association of Computational Linguistics*, 7, 2019.

Thanh-Long V. Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No Prompt Left Behind: Exploiting Zero-Variance Prompts in LLM Reinforcement Learning via Entropy-Guided Advantage Shaping. *ArXiv*, abs/2509.21880, 2025.

Baixuan Li et al. Tongyi DeepResearch Technical Report. *ArXiv*, abs/2510.24701, 2026.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. WebSailor: Navigating Super-human Reasoning for Web Agent. *ArXiv*, abs/2507.02592, 2025.

Junteng Liu, Yunji Li, Chi Zhang, Jingyang Li, Aili Chen, Ke Ji, Weiyu Cheng, Zijia Wu, Chengyu Du, Qidi Xu, Jiayuan Song, Zhengmao Zhu, Wenhu Chen, Pengyu Zhao, and Junxian He. WebExplorer: Explore and Evolve for Training Long-Horizon Web Agents. *ArXiv*, abs/2509.06501, 2025.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Annual Meeting of the Association for Computational Linguistics, (ACL)*, 2023.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *International Conference on Learning Representations (ICLR)*, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, et al. S1: Simple Test-Time Scaling. *ArXiv*, abs/2501.19393, 2025.

Sidhaarth Murali, João Coelho, Jingjie Ning, João Magalhães, Bruno Martins, and Chenyan Xiong. Beyond Parallel Sampling: Diverse Query Initialization for Agentic Search. *ArXiv*, abs/2606.17209, 2026.

Jingjie Ning, João Coelho, Yibo Kong, Yunfan Long, Bruno Martins, João Magalhães, Jamie Callan, and Chenyan Xiong. Agentic search in the wild: Intents and trajectory dynamics from 14m+ real search requests. In *International Conference on Research and Development in Information Retrieval, (SIGIR)*, 2026.

NVIDIA. NeMo RL: A Scalable and Efficient Post-Training Library, 2025. URL https://github.com/NVIDIA-NeMo/RL. Accessed September 25, 2026.

Arnold Overwijk, Chenyan Xiong, and Jamie Callan. ClueWeb22: 10 Billion Web Documents with Rich Information. In *International Conference on Research and Development in Information Retrieval, (SIGIR)*, 2022.

Long Phan et al. Humanity's Last Exam. *ArXiv*, abs/2501.14249, 2025.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Zile Qiao, Guoxin Chen, Xuanzhong Chen, Donglei Yu, Wenbiao Yin, Xinyu Wang, Zhen Zhang, Baixuan Li, Huifeng Yin, Kuan Li, Rui Min, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebResearcher: Unleashing unbounded reasoning capability in Long-Horizon Agents. *ArXiv*, abs/2509.13309, 2025.

Serper. Serper.dev Search API, 2025. URL https://serper.dev. Accessed May 19, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *ArXiv*, abs/2402.03300, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A Flexible and Efficient RLHF Framework. In *European Conference on Computer Systems (EuroSys)*, 2025.

Shreyas Singh, Kunal Singh, and Pradeep Moturi. Fathom-deepresearch: Unlocking long horizon information retrieval and synthesis for slms. *ArXiv*, abs/2509.24107, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-time Compute Optimally Can Be More Effective than Scaling Model Parameters. *arXiv preprint arXiv:2408.03314*, 2024.

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*, abs/2503.05592, 2025.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. ZeroSearch: Incentivize the Search Capability of LLMs without Searching. *ArXiv*, abs/2505.04588, 2025a.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen. SimpleDeepSearcher: Deep information seeking via web-powered reasoning trajectory synthesis. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025b.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebShaper: Agentically Data Synthesizing via Information-Seeking Formalization. *ArXiv*, abs/2507.15061, 2025.

Nandan Thakur, Zijian Chen, Xueguang Ma, and Jimmy Lin. ORBIT: Scalable and Verifiable Data Generation for Search Agents on a Tight Budget. *ArXiv*, abs/2604.01195, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop Questions via Single-hop Question Composition. *Transactions of the Association of Computational Linguistics*, 10, 2022.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information Gain-based Policy Optimization: A Simple and Effective Approach for Multi-Turn Search Agents. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=qkWP6phrvZ.

Tongyu Wen, Guanting Dong, and Zhicheng Dou. SmartSearch: Process Reward-Guided Query Refinement for Search Agents. *ArXiv*, abs/2601.04888, 2026.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhenglin Wang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Robert Tang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebDancer: Towards Autonomous Information Seeking Agency. In *Annual Conference on Neural Information Processing Systems, (NeurIPS)*, 2025a.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. WebWalker: Benchmarking LLMs in Web Traversal. In *Annual Meeting of the Association for Computational Linguistics, (ACL)*, 2025b.

An Yang et al. Qwen3 technical report. *ArXiv*, abs/2505.09388, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*, 2023.

Qiying Yu et al. DAPO: an open-source LLM reinforcement learning system at scale. *ArXiv*, abs/2503.14476, 2025.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agent-Tuning: Enabling Generalized Agent Abilities for LLMs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act Only When It Pays: Efficient Reinforcement Learning for LLM Reasoning via Selective Rollouts. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025a.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling Deep Research via Reinforcement Learning in Real-world Environments. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025b.

Source	Query	Relevant Documents
DRGym	Which three European football clubs did the Surinamese-born Dutch midfielder who later managed AC Milan and was honored as a Nelson Mandela Foundation Legacy Champion represent when he won the UEFA Champions League during his playing career? *Answer:* Ajax, Real Madrid, AC Milan	- Clarence Seedorf - Nelson Mandela
Iterative	What is the name of the game engine utilized by the open-source real-time strategy project to which the creator of Apache Spark contributed water rendering physics during his undergraduate studies? *Answer:* Pyrogenesis	- Databricks - Matei Zaharia - 0 A.D. (video game)
Webgraph	The author of the 1991 Booker Prize-shortlisted novel *Time's Arrow* also wrote a 1975 parody of country-house mysteries. What is the title of the novel for which that author's father won the Booker Prize in 1986? *Answer:* The Old Devils	- Dead Babies (novel) - Time's Arrow (novel) - Kingsley Amis
Comparative	Among the King of Qin who became the first emperor of China, the Macedonian king who founded Alexandria, the Egyptian pharaoh whose nearly intact tomb was discovered by Howard Carter, and the last emperor of the Qing Dynasty who abdicated in 1912, who was the youngest at the time of their first accession to their respective thrones? *Answer:* Puyi	- Alexander the Great - King Tutankhamun - Qin Shi Huang - Puyi

Table 4: Example queries from each generation source, shown together with the corresponding gold answers and supporting documents.

A Query Generation for Multi-turn Search Trajectories

Our synthetic data pool contains the 9k query-answer pairs from the DeepResearchGym synthetic dataset (Coelho et al., 2025), which are grounded in ClueWeb22 documents. We augment this initial set with 16k additional queries produced by three different methods, grounded in the same document corpus. Each method aims to induce a different search strategy: webgraph queries probe hop-following along document adjacency, iterative-search queries train retrieval-driven chaining beyond the hyperlink structure, and comparative queries induce parallel retrieval over independently sourced entities. All generation methods use `gemini-3-flash-preview` as the generator LLM.

Webgraph-based Queries. From a root Web document sampled uniformly from the corpus, we construct a subgraph by iteratively selecting either an in-link or an out-link of the current node, extending the random walk until reaching n documents. The resulting documents are passed to the generator LLM in a single call, which produces one multi-hop question whose answer is a single entity, date, or number, and that cannot be resolved from any single document. The hop structure is determined by hyperlink adjacency, and no retrieval is performed.

Iterative Search Queries. This method also constructs iterative chain questions, but replaces hyperlink traversal with dense retrieval, allowing it to reach documents that are not directly connected in the Web hyperlink graph. Starting from a root Web document, the generator iteratively produces follow-up retrieval queries whose results are incorporated into progressively harder multi-hop questions, extending the chain until n documents are reached. Intermediate questions from the same trajectory can therefore provide valid queries at lower hop counts.

Comparative Search Queries. The previous two methods build sequential chains in which each hop depends on the previous one. Comparison questions instead require retrieving several entities before aggregation, introducing a parallel retrieval structure. Starting from a root document, the

LLM generator first identifies a measurable attribute of its main entity and generates a query targeting a comparable entity of the same type. The retrieved documents are then incorporated one at a time, with the generator updating the comparison question and producing a follow-up query for another comparable entity, until n entities are gathered.

Query Filtering. All candidate queries pass an LLM-based validation stage that discards questions that are answerable from common knowledge or malformed. In the case of iterative search queries, when constructing a question with n target documents, we retain the intermediate version at every $n' < n$, so a single trace yields multiple valid questions of increasing complexity. Failed traces retain the last valid intermediate version, introducing a bias toward lower hop counts.

Figure 7 shows the final distribution of questions according to the corresponding generation source. From the resulting data pool, we fix two disjoint subsets used throughout all experiments: 15k QA pairs for supervised fine-tuning and 10k for reinforcement learning. Table 4 shows examples of queries generated by the aforementioned methods.

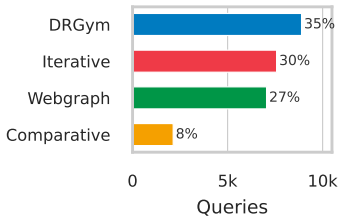


Figure 7: Distribution of the synthetic query pool according to the generation source.

B Training Dynamics Across Synthetic Query Sources

Figures 8 and 9 respectively decompose the training dynamics of the Qwen3-1.7B recycle run according to the query generation source and the required document count, showing that the generation strategies induce distinct search behaviors.

For instance, in Figure 8, DRGym-ASearcher queries require fewer turns per trajectory than queries from the other sources, consistent with their higher too-easy fraction as shown in Figure 9, especially early in training. Iterative-search and webgraph queries induce longer trajectories, reflecting their sequential multi-hop construction.

Comparative-search queries follow a different profile in Figure 8. The agent solves them with fewer turns while issuing more search queries per turn, despite the absence of explicit supervision encouraging parallel retrieval. Comparative questions aggregate independently retrievable entities, and the policy dispatches these retrievals in parallel through the search tool interface.

Source-level difficulty in Figure 9 is already separated at the start of training. DRGym-ASearcher contains a large too-easy fraction from the first stage, while iterative-search and webgraph queries concentrate in the too-hard region, particularly at higher document counts.

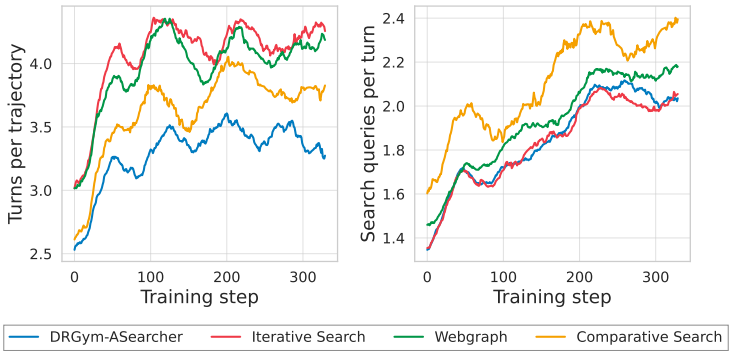


Figure 8: Average trajectory statistics during Qwen3-1.7B RL training under query recycling, grouped by synthetic query source. Left: Average number of turns per search agent trajectory. Right: Average number of search queries issued per turn. DRGym-ASearcher queries require consistently shorter trajectories, while comparative queries exhibit higher query parallelism, reflected in fewer turns but more search calls per turn.

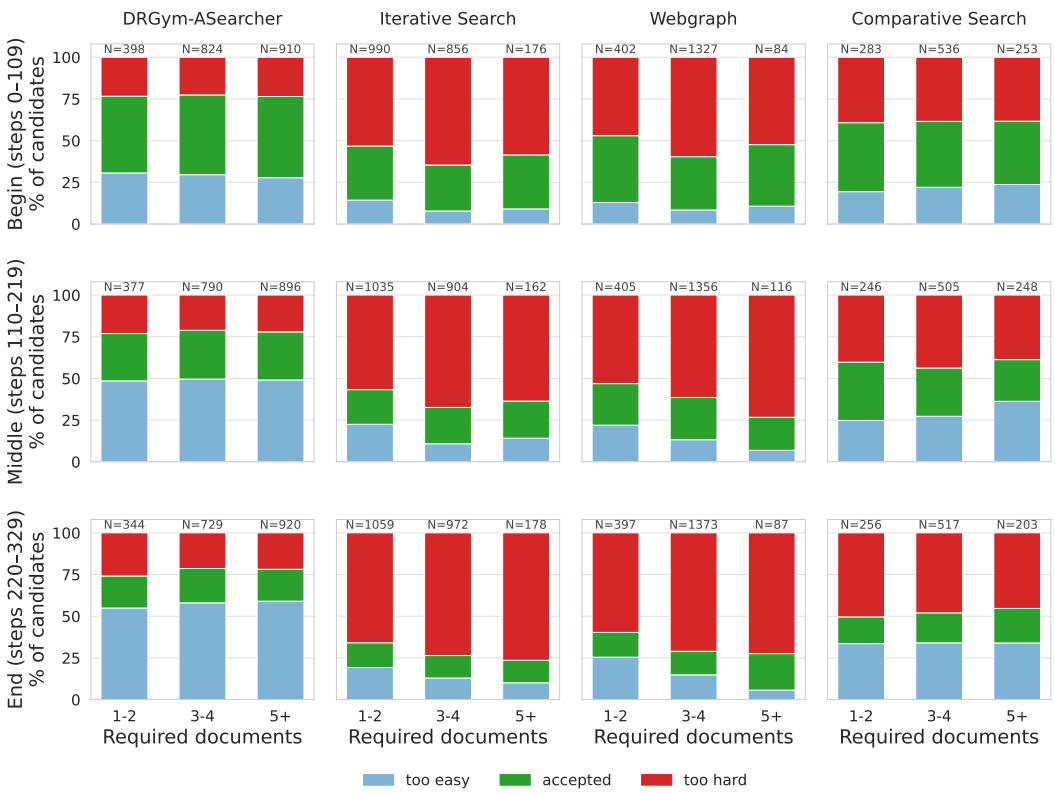


Figure 9: Composition of sampled candidate queries throughout Qwen3-1.7B recycling-based training, stratified by generation source and required document count. Each bar reports the fraction of sampled candidates classified as too-easy, accepted, or too-hard within a training stage (early, middle, or end). Values above bars indicate the number of sampled queries in each subgroup.

As training progresses, the sampled pool shifts toward too-hard queries. However, this shift reflects pool composition rather than policy degradation. Recycling removes accepted queries from the active pool while leaving zero-variance queries eligible for resampling, and thus the remaining candidates concentrate on queries that the current policy does not yet solve consistently. The aforementioned effect is most visible in the middle two columns of Figure 9, where the accepted fraction of iterative-search and webgraph queries decreases while their too-hard fraction expands. Their longer sequential reasoning chains keep these queries near the frontier of the model's capabilities, and they may become signal-bearing once the policy improves enough to solve them.

C Soft Recycling

Section 3.2 instantiates query recycling as a binary weight rule, in which consumed signal-bearing queries are removed from the pool ($w_q^{(t+1)} = 0$), while zero-variance queries are returned at full eligibility ($w_q^{(t+1)} = 1$). Both these branches are nonetheless extreme cases. A query that has contributed once is treated as permanently exhausted, and a query that never leaves the zero-variance region remains eligible for the entire run. We now describe *soft recycling*, i.e. a relaxation that softens both branches. In this case, consumed queries are downweighted rather than removed, and zero-variance queries are retired after a fixed number of unproductive observations.

Let $Z_t \subseteq C_t$ denote the zero-variance candidates at step t, and let $z_q^{(t)}$ count the number of steps up to and including the step t at which q was sampled and observed as zero-variance. The weight update becomes:

Model	Size	k	Rollouts	2Wiki	Bamb	HQA	MSQ	NQ	PopQA	TriQA	Avg
Ours (full synthetic training over ClueWeb22 and evaluation using SERPER)											
Qwen3	1.7B	–	–	37.4	44.4	44.1	16.3	54.5	41.8	68.6	43.9 ± 0.40
+ SFT	1.7B	–	–	54.8	47.2	54.4	24.6	67.6	52.0	86.0	55.2 ± 0.33
+ GRPO	1.7B	1	84.5k	70.9	68.8	66.6	36.9	69.9	**54.4**	86.9	64.9 ± 0.21
+ B-DAPO	1.7B	2	84.5k	69.5	67.2	66.9	36.9	69.5	53.5	85.2	64.1 ± 0.20
+ B-DAPO w/ recycling	1.7B	2	84.5k	**71.7**	**72.4**	**67.7**	**38.0**	**70.1**	54.2	87.8	**66.0** ± 0.17
+ B-DAPO w/ soft recycling	1.7B	2	84.5k	71.5	70.7	67.2	36.5	69.6	52.8	**88.0**	65.2 ± 0.15

Table 5: Pass@1 on seven multi-hop QA benchmarks for the 1.7B model. All RL configurations start from the SFT checkpoint and use a matched rollout budget. Soft recycling uses $\alpha = 0.1$ and a zero-variance budget of $c = 3$. Averages over inference runs are reported as mean ± standard error.

$$w_q^{(t+1)} = \begin{cases} \alpha & \text{if } q \in S_t, \\ 0 & \text{if } q \in Z_t \text{ and } z_q^{(t)} \geq c, \\ w_q^{(t)} & \text{otherwise,} \end{cases}$$

where $\alpha \in [0, 1]$ is the residual weight assigned to consumed queries, and $c \geq 1$ is the zero-variance budget of a query. Queries not drawn at step t keep their previous weight. The binary rule of Section 3.2 corresponds to $\alpha = 0$ and $c = \infty$. Each of the two modified branches targets a different observation from our analysis, and we consider them in turn.

Signal-bearing queries selected into the gradient update batch keep weight α instead of being removed, so with $\alpha > 0$ they continue to compete for sampling mass at later steps, at a fraction of the weight of queries that have not yet contributed. This branch follows from the revisiting behaviour of the non-recycling baselines. Figure 3 shows that a large share of the second-epoch rollouts of GRPO and Bounded-DAPO land on previously signal-bearing queries, and thus having a query that produced signal once does not imply that the query is exhausted under a later policy.

Zero-variance queries are returned at full eligibility, as in the binary rule, until they reach the budget c, at which point their weight is zeroed and they leave the pool permanently. Recycling assumes that zero-variance status is transient, but this does not need to hold for every query: some remain beyond the policy for the entire run, and others are mastered early and never return to the signal-bearing region. Under unbounded eligibility, such queries keep drawing rollouts each time they are sampled without ever entering a gradient update, and their share of the pool grows as signal-bearing queries are consumed. The budget c bounds the number of times any single query can incur rollout cost without producing signal, in either tail.

We report the results of a run with $\alpha = 0.1$ and $c = 3$, so that consumed queries remain sampleable at one tenth the weight of fresh queries, and a query is retired after its third zero-variance observation. All other training settings match the Qwen3-1.7B configuration of Section 4. Over the course of training, the budget retires 2,061 queries from the pool as persistently too-hard, and 1,120 queries as persistently too-easy (i.e. roughly a third of the 10k pool is eventually removed, the majority of it from the too-hard tail). Table 5 shows that soft recycling reaches 65.2 average Pass@1, improving over the the non-recycling baselines, but trailing the binary rule. We did not sweep different values for α or c, so we do not rule out gains at other values.

D Query Recycling Under Process Rewards

The experiments in Section 5.1 assume outcome-only supervision, i.e., no intermediate feedback on the individual search calls of a trajectory. This assumption is standard, since verifying intermediate steps generally requires either a process reward model or annotations that may be unavailable at training time. Still, since we control the synthetic generation pipeline, every query carries the list of gold documents from which it was constructed, and we can therefore score each search call against that list without additional supervision. This appendix reports what happens to query recycling when that additional supervision signal is used.

Model	Size	k	Rollouts	Pass@1 2Wiki	Bamb	HQA	MSQ	NQ	PopQA	TriQA	Avg	Trajectory Tokens	Searches
Outcome Reward (full synthetic training over ClueWeb22 and evaluation using SERPER)													
Qwen3	1.7B	–	–	37.4	44.4	44.1	16.3	54.5	41.8	68.6	43.9 ± 0.40	3,253	3.72
+ B-DAPO	1.7B	2	84.5k	69.5	67.2	66.9	36.9	69.5	53.5	85.2	64.1 ± 0.20	3,684	3.41
+ B-DAPO w/ recycling	1.7B	2	84.5k	**71.7**	**72.4**	**67.7**	**38.0**	**70.1**	54.2	87.8	**66.0** ± 0.17	5,583	5.70
Dense Reward (full synthetic training over ClueWeb22 and evaluation using SERPER)													
Qwen3	1.7B	–	–	37.4	44.4	44.1	16.3	54.5	41.8	68.6	43.9 ± 0.40	3,253	3.72
+ B-DAPO w/ process reward	1.7B	2	84.5k	61.6	60.3	60.7	31.1	67.9	50.8	85.7	59.7 ± 0.27	1,776	1.63
+ B-DAPO w/ recycling + process reward	1.7B	2	84.5k	66.6	63.1	61.7	31.7	68.1	52.4	84.7	61.2 ± 0.20	1,961	1.89

Table 6: Pass@1 results on seven multi-hop QA benchmarks for the 1.7B model, comparing outcome-only rewards against process rewards. All RL configurations start from the SFT checkpoint and use a matched rollout budget. Averages over inference runs are reported as mean ± standard error. The two rightmost columns report mean trajectory length at evaluation, in terms of the total number of context tokens, and search calls per question.

We adopt a variant of the reward proposed for SmartSearch (Wen et al., 2026). A search query issued during a trajectory is *good* if it returns at least one gold document of the original question that has not yet been returned earlier in the same trajectory, and *bad* otherwise. Let n^+ and n^- denote the number of good and bad queries in a trajectory. The reward of a rollout o for query q becomes:

$$R(o,q) = \begin{cases} \max\left(0.7,\ 1 - 0.1 \times n^-\right) & \text{if the final answer is correct,} \\ \min\left(0.3,\ 0 + 0.1 \times n^+\right) & \text{otherwise.} \end{cases}$$

Correct answers, which previously received a reward of exactly 1, are penalized for unproductive search calls but never fall below 0.7. In turn, incorrect answers are credited for productive search calls but never rise above 0.3. The constants follow the values used by the original authors.

Results show that reward shaping degrades accuracy in our setting. Table 6 compares both reward functions under matched rollout budgets, and the trajectory statistics show that shaped policies answer after 1.63 and 1.89 search calls on average, against 3.41 and 5.70 under outcome-only rewards, with generated tokens dropping by a comparable factor.

It should nonetheless be noted that query recycling remains effective under shaped rewards. Shaping with the SmartSearch reward reduces zero-variance groups, since rollouts that agree on the final answer can still differ in retrieval quality, but it does not fully remove them. During the course of training training, 2,841 queries produced at least one zero-variance group and were returned to the pool, so roughly a quarter of the dataset would be discarded by a non-recycling method. The per-batch zero-variance rate also grows over training, from 24% to 40%, as rollouts within a group become increasingly similar. Of the re-queued queries, 77.6% were later re-drawn and produced a trainable group. The effect on final accuracy persists, as the identically configured run without query recycling scores 1.5 Pass@1 lower (see results in Table 6).

E LLM JUDGE ANALYSIS

Throughout this work, we follow the evaluation protocol used in the BehaviorPrime study (Jin et al., 2026), which relies on an LLM judge to assess the correctness of answers produced by search agents. While LLM-based evaluation accommodates semantic variations that string-matching metrics cannot capture, it also introduces reproducibility concerns, as judgments may vary across models and across repeated evaluations with the same model. To quantify this variation, we randomly sampled 1000 query-answer pairs from the models and datasets reported in Table 3, originally evaluated using `gpt-4.1-mini`. We then relabeled these pairs using five additional LLM judges: `gpt-4.1`, `gpt-4o`, `gemini-2.5-flash`, `gemini-3.1-pro-preview`, and also `Qwen3-32B`.

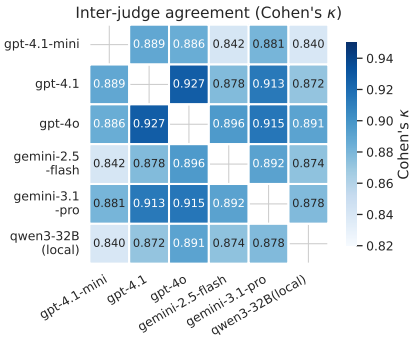


Figure 10: Inter-judge agreement (Cohen's κ) between multiple LLM judges.

Figure 10 reports the pairwise Cohen's κ across the six judges. All values fall in the *almost perfect* agreement range. This high level of inter-judge agreement indicates that correctness labels are largely insensitive to the choice of model for implementing the judge, supporting the robustness of evaluations conducted by the LLM judge protocol.

To isolate run-to-run variability, we relabeled the same pairs five times with our primary judge, `gpt-4.1-mini`. The fraction judged correct is stable, with a standard deviation of 0.2%. At the level of individual pairs, 96% receive an identical label across all runs. Repeated single-judge evaluation therefore introduces a small variation in the reported metrics, which we control during the main experiments by reporting averages over multiple inference passes.