AVOIDING SEARCH BEHAVIOR HOMOGENIZATION IN AGENTIC RAG WITH QUERY-AND-DOCUMENT TRAJECTORY STEERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic retrieval-augmented generation (RAG) integrates dynamic search tool calling with reinforcement learning with verifiable rewards (RLVR) to overcome the static parametric knowledge of LLMs. However, training a single policy across diverse search patterns induces behavioral conflicts, causing the model to collapse onto an average search pattern, which we call *behavioral homogenization*. Since existing approaches leverage extra compute or memory for explicit planning and self-correction, they incur significant computational inefficiency and fail to foster positive transfer among similar search patterns while disentangling conflicting search patterns. To overcome this challenge, we propose *query-and-document trajectory steering* (QDTS), a framework designed to maximize cooperative learning across complementary search patterns while suppressing conflicting signals with minimal cost, by steering generation with query and document contexts. Specifically, *query-side pre-rollout steering* categorizes queries into structural patterns via a rule-based parser, supplying corresponding class contexts to guide targeted reasoning pathways during both training and inference. Furthermore, *document-side mid-rollout steering* evaluates online retrieval statistics and deploys dynamic mid-execution contexts across multi-turn rollouts to identify search failures and steer subsequent actions. By integrating this two-stage mechanism to generate and optimize agentic search trajectories, QDTS effectively learns an adaptive policy across diverse query patterns, outperforming the state-of-the-art method by 9.2% on average across comprehensive agentic RAG benchmarks.

1 INTRODUCTION

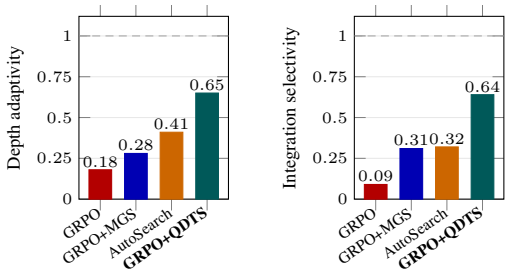
In recent years, *agentic* retrieval-augmented generation (RAG) has emerged as a powerful paradigm for overcoming the limits of static parametric knowledge in large language models (Jin et al., 2025; Zheng et al., 2025; Gao et al., 2025; Song et al., 2025; Deng et al., 2026; Sun et al., 2026). As reasoning unfolds toward an answer, the agent identifies information unavailable from its parametric knowledge, formulates queries to acquire the required information, and initiates retrieval from external databases. The agent then incorporates evidence from the retrieved documents into its reasoning and initiates further searches when necessary. To train agents for such multi-turn search and reasoning, recent studies optimize these interaction trajectories using reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2025; Deng et al., 2026). In RLVR training, rewards are assigned based on final-answer correctness (Jin et al., 2025) or step-wise process evaluations (Zheng et al., 2025; Song et al., 2025), reinforcing retrieval actions and reasoning steps that drive toward an answer.

In practice, search agents should handle *heterogeneous* search demands: performing fewer search steps for simple queries but pursuing deeper search for complex ones, and incorporating evidence from relevant documents but continuing to search when retrieved documents have low relevance. However, jointly training an agent across these demands with RLVR can be challenging because updates that encourage one behavior may conflict with those needed for another, potentially leading to similar search behaviors across different demands. We refer to this failure mode as *behavioral homogenization* and empirically demonstrate it from the query and document perspectives. Figure 1(a) shows that the search depth of jointly trained agents barely increases with the gold depth of the query, while Figure 1(b) shows that they consistently integrate retrieved documents even when their

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.

relevance is low. These behaviors can lead to unnecessary searches for simple queries, premature search termination for complex ones, and hallucinations from integrating low-relevance documents, highlighting the importance of adjusting search behavior to heterogeneous search demands.

To mitigate behavioral homogenization, one could reward search behaviors appropriate to each demand in addition to response quality. However, reward design alone is inherently ***retrospective***: it can only evaluate trajectories already sampled by the policy (Wu et al., 2026a). Consequently, when trajectories that both satisfy the search demand and reach the correct answer are not sufficiently sampled, the reward provides limited signal for reinforcing the desired behavior. This issue is particularly relevant in agentic RAG, where answer correctness and appropriate search behavior need not coincide: a correct trajectory may contain unnecessary searches, whereas a trajectory that follows the intended search pattern may still fail to retrieve or integrate the evidence needed to answer correctly.



(a) Query-side Homogenization. (b) Document-side Homogenization.

Figure 1: **Behavioral homogenization**, scored so 0 is fully homogenized and 1 is fully demand-adaptive (dashed line). **(a)** Slope of the regression of realized depth on gold depth. **(b)** $1 - P(\text{int} \mid \neg\text{gold})/P(\text{int} \mid \text{gold})$. NQ (1-hop) and MuSiQue (2, 3+), 400 questions per bucket.

To address this limitation, we introduce *context-based trajectory steering*, a simple concept that uses demand-specific contexts to steer trajectory generation toward the desired search behavior. Unlike reward-based learning, context-based trajectory steering is ***prospective***: it conditions the policy on the search demand before and during trajectory generation, eliciting relevant behaviors already acquired during pre-training. Rather than waiting for a desired trajectory to emerge and then rewarding it, the context steers generation toward the desired search pattern from the outset. This conditioning also separates the learning signals associated with different search demands. Questions sharing the same demand are trained under the same context and thus reinforce a common search pattern, whereas questions with different demands are conditioned on different contexts, reducing conflicting behavioral supervision under an otherwise identical conditioning state.

To instantiate context-based trajectory steering, we develop **Q**uery-and-**D**ocument **T**rajectory **S**teering (**QDTS**), a framework that steers agentic RAG at two complementary levels. QDTS targets the two sources of homogenization illustrated in Figures 1(a) and 1(b) through two forms of trajectory steering: *query-side pre-rollout steering* differentiates how search should unfold for queries with different search demands, while *document-side mid-rollout steering* differentiates how the agent should proceed at retrieval steps with different retrieval outcomes.

Query-side Pre-rollout Steering To address the *query-side homogenization* in Figure 1(a), query-side steering conditions each query on its *search depth* and *structural pattern* (e.g., 2-hop / bridge). Search depth specifies how far the search should proceed, while the structural pattern specifies how it should unfold, such as resolving an intermediate entity before searching for the target. We initially optimize these contexts before RL training using prompt optimization (Agrawal et al., 2026) to effectively elicit relevant behaviors already acquired by the base policy during pre-training.

Document-side Mid-rollout Steering To address the *document-side homogenization* in Figure 1(b), document-side steering conditions each search step on its *retrieval state*, inferred from query-document relevance and inter-document density. This context distinguishes retrieval outcomes that would otherwise appear identical to the agent, such as a successful retrieval, a potential miss, or an overly broad result. The agent can therefore adapt its next action to the observed retrieval state—by refining the query or proceeding to the next dependency—rather than following the same trajectory regardless of retrieval quality. The contexts are optimized with the same procedure as query-side.

In summary, our QDTS framework improves average exact match (EM) by 3.7 percentage points (9.2% relative) over the strongest baseline trained under identical conditions. Extensive ablations and qualitative analyses further demonstrate that our framework enables efficient learning of a search-pattern-adaptive policy for agentic RAG. The source code is available at https://anonymous.4open.science/r/QDTS-E3D8.

2 Related Work

Agentic RAG Development Agentic RAG systems transition standard passive retrieval into active, multi-turn tool engagement (Schick et al., 2023; Yao et al., 2022). RLVR for training agentic RAG systems have then emerged with answer-only reward (Jin et al., 2025; Deng et al., 2026). To further enhance agentic RAG, existing research has primarily pursued three major paradigms. The first direction is sophisticated reward design which moves beyond sparse final-outcome signals to evaluate intermediate trajectory steps by employing process supervision or multi-objective rewards, which includes scoring individual retrieval relevance (Zheng et al., 2025; Zhang et al., 2026), penalizing redundant tool calls (Sun et al., 2026), or rewarding structural formatting compliance (Song et al., 2025). The second direction scales *inference-time computation*. Self-reflective methods interleave generation with structural critique, either through learned reflection tokens (Asai et al., 2024) or through tool-interactive verification (Gou et al., 2024), while corrective methods attach an external relevance evaluator that triggers retrieval correction (Yan et al., 2024), The third direction scales *parameters or modules*, such as integrating dynamic planning components (Jeong et al., 2024) or multi-agent systems (Chen et al., 2026).

Search-Depth Pathologies in Agentic RAG The search-depth bias has been reported in multiple papers independently. HipRAG (Wu et al., 2026b) discovers high over-search and under-search rates for RL-trained search agents including Search-R1, and mitigate them with gated hierarchical process rewards. SAAS (Tang et al., 2026) target over-search through self-aware reinforcement learning. Autosearch (Sun et al., 2026) and CuSearch Shen et al. (2026) address depth adaptivity through reward shaping and depth-stratified curriculum rollout sampling respectively. These results confirm that a single RLVR-trained policy mis-sets retrieval depth in both directions, and that the effect is not specific to one training recipe. Where these methods intervene on the reward or on the sampling schedule, we intervene on the conditioning of the policy itself, cost-efficient by leaving the reward sparse and without additional rollouts or external LLM usage.

3 Methodology: QDTS

3.1 Problem Formulation of Agentic RAG

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ denote a large background corpus of raw documents. We consider a query-answering dataset $\mathcal{S} = \{(x_i, y_i, \mathcal{D}_i^*)\}_{i=1}^N$, where each query-answer pair (x_i, y_i) is anchored to a ground-truth subset of supporting documents $\mathcal{D}_i^* \subseteq \mathcal{D}$. Retrieval is performed by a frozen retriever $\mathcal{R}_k$, which returns the k documents of $\mathcal{D}$ nearest to a sub-query q in embedding space.

An **agentic RAG** system operates as a finite-horizon Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}_{\text{MDP}}, \mathcal{A}, \mathcal{P}, R, T)$, with a horizon T, where:

- **State** $s_t \in \mathcal{S}_{\text{MDP}}$ represents the context history at step t, initialized by the user query $s_0 = x$.
- **Action** $a_t \in \mathcal{A}$ denotes an executable step, such as sub-query formulation, document retrieval, reasoning, or answering. We write a_{search} for issuing a further retrieval and a_{int} for integrating the evidence gathered so far, since the choice between the two is what determines search behavior.
- **Transition** $\mathcal{P}(s_{t+1} \mid s_t, a_t)$ appends the action and the environment's response to the state. For a search carrying a sub-query q, the state is updated as $s_{t+1} = s_t \oplus q \oplus \mathcal{R}_k(q)$; the agent chooses the sub-query while the passages are supplied by the retriever, and the transition proceeds independently of their relevance to q.
- **Reward** $R(\tau)$ is a scalar trajectory return assembled from a set of objectives $\mathcal{R} = \{R_1, \ldots, R_M\}$ evaluated over $\tau = (s_0, a_0, \ldots, s_T, a_T)$. $R(\tau) = \sum_{m=1}^M w_m R_m(\tau)$, with $w_m \geq 0$. It covers the process-reward baselines, though by default $\mathcal{R}$ is a single terminal correctness objective.

The agentic policy $\pi_\theta(a_t \mid s_t)$, parameterized by θ, aims to maximize the expected return over $\mathcal{S}$:

$$J(\theta) = \mathbb{E}_{(x,y,\mathcal{D}^*)\sim\mathcal{S}}\big[\mathbb{E}_{\tau\sim\pi_\theta(\cdot|x)}[R(\tau)]\big]. \tag{1}$$

Let $p_\theta(\tau \mid x) = \prod_t \pi_\theta(a_t \mid s_t)\,\mathcal{P}(s_{t+1} \mid s_t, a_t)$ be the trajectory distribution induced by π_θ. Equation 1 averages over questions, but the choice between a_{search} and a_{int} is made after retrieval, not at s_0, so we also use the step-level marginal $d^{\pi_\theta}(s \mid x) \propto \sum_\tau p_\theta(\tau \mid x) \sum_t \mathbb{1}\{s_t = s\}$ over the intermediate states that an episode from x visits; both depend on θ.

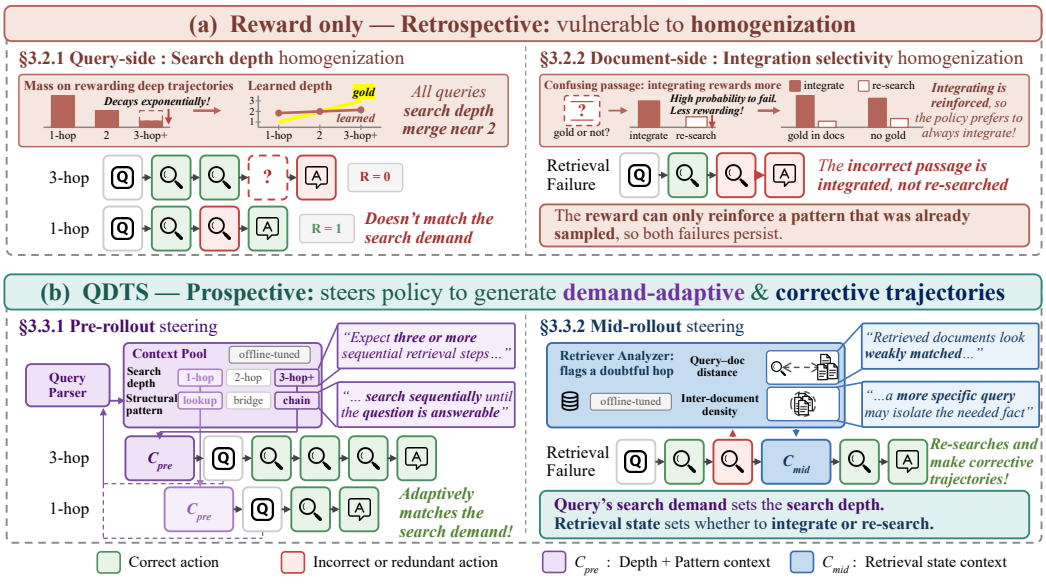


Figure 2: **How behavioral homogenization occurs and how QDTS resolves it.** (**a**) Under reward-only training, the realized search depth can become nearly independent of the query's gold depth, and retrieved passages are integrated irrespective of whether they contain the gold evidence. (**b**) QDTS conditions the policy at two points. A class context C_{pre} is prepended before the rollout, so that the search depth is steered by the query's search demand. A retrieval-state context C_{mid} is appended after each doubtful retrieval, so that the agent re-searches the entity rather than integrating the passage.

3.2 Why Homogenization Emerges under Retrospective Rewards

The limitation of being retrospective is visible in the gradient of Equation 1,

$$\nabla_\theta J(\theta) = \mathbb{E}_x\Big[\sum_\tau p_\theta(\tau \mid x)\, R(\tau)\, \nabla_\theta \log p_\theta(\tau \mid x)\Big]. \tag{2}$$

The reward enters only through the product $p_\theta(\tau \mid x)R(\tau)$. Thus, a trajectory that the policy rarely produces contributes little even when it receives a high reward. Let $\mathcal{T}^*(x)$ be the trajectories that both meet the search demand of x and answer it correctly, and $\varepsilon_\theta(x) = \sum_{\tau\in\mathcal{T}^*(x)} p_\theta(\tau \mid x)$ be the mass currently placed on them. Their contribution to the update is bounded by

$$\Big\| \sum_{\tau\in\mathcal{T}^*(x)} p_\theta(\tau \mid x) R(\tau) \nabla_\theta \log p_\theta(\tau \mid x) \Big\| \;\le\; \varepsilon_\theta(x)\, R_{\max}\, G, \tag{3}$$

where G is an upper bound on the norm of the score function $\nabla_\theta \log p_\theta(\tau \mid x)$. This bound is finite because rollouts are restricted to a fixed number of turns and the clipped objective limits the magnitude of individual updates (Schulman et al., 2017; Shao et al., 2024). Since $R_{\max}$ and G are constants, the right-hand side vanishes as $\varepsilon_\theta(x)$ vanishes. Consequently, no redesign of the reward can reinforce a behavior to which the current policy assigns negligible probability, because any reward is multiplied by that probability before it reaches the gradient. In the following two subsections, we identify why $\varepsilon_\theta(x)$ becomes small on the query side and on the document side, respectively.

3.2.1 Query-Side: Search Depth Homogenization

Search depth homogenizes because $\varepsilon_\theta(x)$ decreases with the depth required by a query, causing deeper queries to contribute progressively less to learning the policy's stopping behavior. The query side of Figure 2 visualizes how this homogenization behavior is manifested.

Let $H^*(x)$ denote the gold depth of a query x and $H(\tau)$ denote the realized depth in a trajectory. A trajectory in $\mathcal{T}^*(x)$ must complete $H^*(x)$ retrievals, each of which must return the passage that its sub-query targets. If each hop succeeds with probability $p < 1$ under the frozen retriever, then

$$\varepsilon_\theta(x) \;\lesssim\; p^{\,H^*(x)}. \tag{4}$$

By Equation 3, a query's contribution to the policy update decays exponentially with its gold depth.

In addition, the policy cannot distinguish queries of different depth at the point where the decision is made. The state s_h reached after h searches contains all previously issued sub-queries and their retrieved passages, but it does not contain $H^*(x)$ explicitly. A shared stopping rule is therefore applied to every query that reaches depth h. We define the continuation advantage $\delta(x,h) := \mathbb{E}_{s_h\sim d^{\pi_\theta}(\cdot|x)}[A^{\pi_\theta}(s_h, a_{\text{search}}) - A^{\pi_\theta}(s_h, a_{\text{int}})]$, which is positive for $h < H^*(x)$ and negative for $h \geq H^*(x)$. The shared stopping rule is updated according to the mass-weighted aggregate,

$$\bar{\delta}_\theta(h) \;=\; \mathbb{E}_x\big[w_\theta(x,h)\,\delta(x,h)\big], \qquad w_\theta(x,h) \propto d^{\pi_\theta}(s_h \mid x). \tag{5}$$

By Equation 4, these weights favor shallow queries. Beyond a certain depth $\bar{H}$, the aggregate therefore becomes negative even for queries whose $\delta(x,h)$ is positive, and the realized depth $H(\tau)$ concentrates around $\bar{H}$ regardless of the value of $H^*(x)$. Appendix A shows that this concentration reinforces itself and eventually produces exactly zero gradient on deep queries.

Modifying the reward cannot resolve this failure, because the aggregate is formed over states that the reward can no longer reach. What is needed instead is a signal that separates queries according to $H^*(x)$ and becomes available from the initial state, so that every subsequent stopping decision in the rollout is conditioned on it.

3.2.2 Document-side: Integration Selectivity Homogenization

Integration selectivity homogenizes because the policy often cannot determine whether the gold passage is among the retrieved documents. At such a state, a corrective re-search has a low probability of reaching the reward, whereas integration is rewarded more often, and the update therefore favors integration regardless of coverage, as illustrated on the document side of Figure 2(a).

Consider a state s_t reached immediately after a retrieval, and let $g_t \in \{0,1\}$ indicate whether the retrieved passages cover the sub-claim currently being tested. The optimal action is a_{int} when $g_t = 1$ and a_{search} when $g_t = 0$. We measure how closely the policy follows this rule by its selectivity,

$$S := 1 - \frac{\pi_\theta(a_{\text{int}} \mid g_t = 0)}{\pi_\theta(a_{\text{int}} \mid g_t = 1)}, \tag{6}$$

which is positive only if π_θ depends on g_t. The transition, however, appends $s_{t+1} = s_t \oplus q \oplus \mathcal{R}_k(q)$ regardless of coverage, so g_t can be inferred only from the text of the passages. Let $\mathcal{T}_{\text{corr}}(x)$ denote the trajectories that respond to $g_t = 0$ by issuing another search. Producing such a trajectory requires that the policy first detect the miss and then recover from it, which yields

$$\varepsilon_\theta^{\text{corr}}(x) \;\lesssim\; \pi_\theta(a_{\text{search}} \mid g_t = 0) \cdot p, \tag{7}$$

and Equation 3 applies as before. The detection rate $\pi_\theta(a_{\text{search}} \mid g_t = 0)$ is the only factor that depends on the policy. Because near-miss passages resemble relevant ones and the accumulated context often exceeds the effective capacity of the model, the states with $g_t = 0$ and $g_t = 1$ are frequently indistinguishable to the policy, and integration yields the higher return on average. The update therefore lowers the detection rate, $\varepsilon_\theta^{\text{corr}}(x)$ vanishes in the sense of Equation 3, and S approaches zero. Appendix B gives the argument in full and shows that denser process rewards do not resolve it, because at inference time the policy must still infer g_t from a state that does not encode it. What is required is a signal available at s_t during the rollout and correlated with g_t.

3.3 Method: query-and-document trajectory steering (QDTS)

QDTS addresses behavioral homogenization at its source by conditioning π_θ on a context $c \in \mathcal{C}$ in addition to the state. Equation 3 explains why conditioning, rather than a modified reward, is the appropriate lever. Increasing $\varepsilon_\theta(x)$ by updating θ fails because the gradient that would increase it has already vanished. In contrast, replacing $p_\theta(\tau \mid x)$ with $p_\theta(\tau \mid x, c)$ changes the sampling distribution immediately. The preceding analysis identified two signals that the policy lacks. The first is a signal that separates queries according to $H^*(x)$ before the rollout begins, and the second is a signal that correlates with g_t during the rollout. Because these two signals become available at different times, QDTS consists of two stages, which are illustrated in Figure 2 with example in Appendix C.

3.3.1 Query-side Pre-rollout Steering

This stage provides the query-side signal. A deterministic parser ψ (details in Appendix D) extracts two entity-invariant attributes from a query string, its *search depth*, which correlates with $H^*(x)$, and

its *structural pattern*, which indicates how the hops are composed, and assigns the query to a class $\psi(x)$. The class context $C_{\text{pre}}(x)$ (e.g., for a bridge pattern, "find an intermediate entity, then search for the final attribute") is prepended to s_0, so the stopping decision at depth h is no longer made at a state shared by queries with opposite demands, and Equation 5 is replaced by its within-class counterpart,

$$\bar{\delta}_\theta(h, c) = \mathbb{E}_{x|C_{\text{pre}}(x)=c}\big[w_\theta(x, h)\,\delta(x, h)\big]. \tag{8}$$

Within a class, $H^*(x)$ is approximately constant. The sign of $\delta(x, h)$ is therefore constant as well, and the aggregate tracks this sign instead of averaging over opposite signs.

Conditioning also increases the probability mass on the desired trajectories, i.e., $\varepsilon_\theta(x, C_{\text{pre}}(x)) > \varepsilon_\theta(x)$ with θ unchanged, which is precisely what Equation 3 requires. Rather than assuming that a hand-written context has this property, we optimize for it directly. The context for a class $\psi(x)$ is chosen to maximize the mass that it elicits from the base policy θ_0,

$$c_{\psi(x)} \;\in\; \arg\max_{c}\; \mathbb{E}_{x\sim\mathcal{S}_{\psi(x)}}\big[\varepsilon_{\theta_0}(x, c)\big], \qquad C_{\text{pre}}(x) = c_{\psi(x)}, \tag{9}$$

where $\mathcal{S}_{\psi(x)}$ denotes the set of queries that are assigned to class $\psi(x)$, and the expectation is estimated by the fraction of rollouts under c that answer correctly. We search over candidate contexts using reflective prompt optimization (Agrawal et al., 2026) for a small number of rounds. Because the search is scored against θ_0, it is completed before RL begins, and the resulting contexts are held fixed throughout training. Both the seed strings and the optimized strings are listed in Appendix E.

3.3.2 Document-side Mid-rollout Steering

This stage provides the document-side signal. An analyzer ϕ inspects the output of the retriever after every search and emits a short context that describes the *retrieval state*. This context (e.g., for a distant query-document state, "a revised search may find better coverage") is appended to the state before the next action is taken,

$$C_{\text{mid},t} = \phi(s_t, \Omega_{\text{stat}}), \qquad \tilde{s}_t = s_t \oplus C_{\text{mid},t}, \qquad a_t \sim \pi_\theta(\cdot \mid \tilde{s}_t), \tag{10}$$

where Ω_{stat} holds the thresholds that determine which retrieval state is reported. The analyzer has no learned parameters and makes no calls to a language model. It uses only the embeddings that the retriever has already computed. For a query embedding $\mathbf{e}_q$ and the embeddings $\{\mathbf{e}_i\}_{i=1}^k$ of the top-k retrieved passages, we compute

$$d_{\text{embed}} = 1 - \tfrac{1}{k}\textstyle\sum_i \cos(\mathbf{e}_q, \mathbf{e}_i), \qquad \rho_{\text{embed}} = \Big[\binom{k}{2}^{-1} \textstyle\sum_{i<j} \big(1 - \cos(\mathbf{e}_i, \mathbf{e}_j)\big)\Big]^{-1}. \tag{11}$$

A large d_{embed} indicates that the retrieval missed and the sub-query should be reformulated, whereas a large ρ_{embed} indicates a dense region of the corpus in which the sub-query should be more specific.

The analyzer thus provides π_θ with an estimate $\hat{g}_t$ of the coverage indicator that the transition does not record. The selectivity S in Equation 6 increases to the extent that $\hat{g}_t$ correlates with g_t. The analyzer also raises the detection rate in Equation 7, which yields $\varepsilon_\theta^{\text{corr}}(x, C_{\text{mid},t}) > \varepsilon_\theta^{\text{corr}}(x)$ with θ unchanged, and the reward once again has corrective trajectories that it can reinforce.

The thresholds Ω_{stat} and the context c_r emitted for each retrieval state r are fit before RL against the base policy, by the same criterion of Equation 9 with the corrective mass in place of ε_{θ_0},

$$\big(\Omega_{\text{stat}}, \{c_r\}\big) \;\in\; \arg\max_{\Omega,\,\{c_r\}}\; \mathbb{E}_{x\sim\mathcal{S}}\big[\varepsilon_{\theta_0}^{\text{corr}}(x; \Omega, \{c_r\})\big], \tag{12}$$

The thresholds are quantiles of each statistic rather than absolute values, so that they transfer across retrievers. The optimized contexts are phrased as observations and possibilities rather than as forcing instructions, so the final decision remains with the policy. Appendix F provides the details, and §4.3 shows that this optimization stage accounts for most of the gain of the mid-rollout stage.

3.3.3 Unified Context-Conditioned Optimization

We train the policy with group-relative optimization (Shao et al., 2024), sampling n rollouts per query under the conditioned policy. Because $C_{\text{pre}}(x)$ is identical across the group, it shifts the sampling distribution without affecting the group-relative baseline; because $C_{\text{mid},t}$ varies with the realized retrievals, it introduces behavioral variance exactly in the states where retrieval has failed. Together, the two stages restore the two quantities, $\varepsilon_\theta(x)$ and $\varepsilon_\theta^{\text{corr}}(x)$, that Section 3.2 showed to vanish, without changing the reward, the rollout budget, or the composition of the advantage group. Appendix G provides the corresponding estimator.

4 EXPERIMENTS

4.1 EXPERIMENT SETTINGS

Datasets and Implementation Details We train on a mixture of single-hop and multi-hop QA datasets—Natural Questions (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022), and 2WikiMultihopQA (Ho et al., 2020)—and evaluate on these datasets as well as PopQA (Mallen et al., 2023) and Bamboogle (Press et al., 2023). For the main experiments and component ablations, we construct a diversity-based training coreset that balances the 18 intent templates $t(q) = [\text{Structural pattern}] \times [\text{Search depth}]$ within each source dataset. We use Qwen2.5-7B-Instruct (Hui et al., 2024) as the policy backbone for all main experiments and ablations to ensure comparability across methods. We report exact match (EM) and token-level F1 against the answers. Appendix H provides further implementation details.

Compared Methods We compare QDTS against representative baselines spanning agentic RAG and multi-task optimization. For outcome-reward models, we include Search-R1 with both PPO and GRPO. For process-reward models, we include StepSearch (Zheng et al., 2025) and ReasonRAG (Zhang et al., 2026). We also include AutoSearch (Sun et al., 2026), which adapts search effort across questions. Furthermore, we consider the methods designed to mitigate multi-task interference at the parameter and gradient levels. At the parameter level, we employ LoRA with class-conditioned low-rank adapters to separate parameter updates across different query types (Xia et al., 2024). At the gradient level, we consider Multi-task Gradient Surgery (MGS) (Yu et al., 2020), which mitigates conflicting updates by projecting conflicting gradients onto each other's normal planes.

4.2 MAIN EXPERIMENT RESULTS

QA Accuracy Table 1 reports EM and F1 on two single-hop and four multi-hop benchmarks. QDTS achieves the best average performance on both metrics and outperforms both the process-reward baselines and the multi-task baselines on every multi-hop benchmark. The process-reward methods such as AutoSearch gain on some benchmarks but not consistently, and the multi-task methods that separate updates by the parameter or gradient level recover only a small fraction of the performance gap, since neither changes which trajectories are sampled. The gains of QDTS are most pronounced on the multi-hop benchmarks, where both forms of homogenization appear, while single-hop performance remains comparable, which is what a demand-adaptive policy should show.

Table 1: Answer accuracy in EM and F1 compared with the state-of-the-art methods.

	NQ		PopQA		HotpotQA		2Wiki		MuSiQue		Bamboogle		Avg.	
Method	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Search-R1 (PPO)	34.1	44.8	41.7	46.1	35.5	40.6	38.2	44.5	17.8	21.3	36.8	47.6	34.0	40.8
Search-R1 (GRPO)	41.4	50.0	42.1	47.5	39.0	47.5	45.8	53.5	21.7	27.8	37.6	51.2	37.9	46.3
Search-R1 (GRPO + LoRA)	45.6	51.1	42.1	47.8	42.4	49.0	41.8	49.1	18.5	25.7	33.6	47.6	37.3	45.1
Search-R1 (GRPO + MGS)	44.1	48.9	43.1	49.3	46.4	53.2	46.2	53.5	23.6	31.2	39.2	57.0	40.4	48.9
StepSearch	35.5	46.1	36.8	44.0	38.6	50.2	36.6	43.1	22.6	31.2	40.0	53.4	35.0	44.7
ReasonRAG†	40.4	47.6	41.5	46.2	38.4	48.9	43.6	50.4	12.8	20.6	36.0	45.5	35.5	43.2
AutoSearch†	**47.6**	**56.0**	46.3	50.3	42.7	54.2	43.6	50.3	19.8	24.6	41.9	54.9	40.3	48.4
Search-R1 (GRPO + QDTS)	45.8	52.2	**47.1**	**51.5**	**49.8**	**54.3**	**52.5**	**57.9**	**26.8**	**33.1**	**42.4**	**58.5**	**44.1**	**51.3**

†Taken from the original papers to provide higher performance, except ReasonRAG on NQ and AutoSearch on MuSiQue, which those papers do not report and we reproduced.

Behavioral Homogenization Table 2 measures the two types of homogenization on logged rollouts. On the query side, the search depth of GRPO is almost independent of the gold depth, and the multi-task and process-reward baselines recover only part of the dependence. QDTS attains the steepest depth slope and the lowest under-search, at the cost of some over-search, which is natural under an imperfect retriever. *Under-search* is the more critical failure, since a missing hop cannot be recovered. On the document side, QDTS maintains a high integration rate in the presence of a gold passage but substantially reduces it when no gold passage is retrieved, which yields a selectivity about twice that of the next best method. Steering therefore restores both forms of adaptivity missing under outcome reward alone: query-adaptive search depth and retrieval-adaptive integration.

Table 2: **Behavioral homogenization.** Query side: m is the regression slope of realized depth H on gold depth H^*; over-search is $\mathbb{E}[H(\tau)-H^*(x) \mid H(\tau) \geq H^*(x)]$; under-search is $\mathbb{E}[H^*(x)-H(\tau) \mid H(\tau) \leq H^*(x)]$. Document side: $P(\text{int} \mid \cdot)$ is the rate of integrating rather than searching again, conditioned on whether a gold passage is in the retrieved set, and $S = 1 - P(\text{int} \mid \neg g)/P(\text{int} \mid g)$. NQ (1-hop) and MuSiQue (2-hop, 3-hop+) are tested with 400 questions per hop bucket.

	Query side			Document side		
Method	$\boldsymbol{m(H, H^*)} \uparrow$	Over-search $\downarrow$	**Under-search** $\downarrow$	$P(\text{int} \mid g) \uparrow$	$P(\text{int} \mid \neg g) \downarrow$	$\boldsymbol{S} \uparrow$
StepSearch	0.28	0.27	0.37	0.86	0.68	0.21
ReasonRAG	0.35	<u>0.13</u>	0.84	**0.93**	0.70	0.16
AutoSearch	<u>0.41</u>	**0.08**	1.15	0.53	<u>0.36</u>	<u>0.32</u>
Search-R1 (PPO)	0.24	0.95	<u>0.11</u>	0.51	0.43	0.15
Search-R1 (GRPO)	0.18	0.22	0.28	0.85	0.77	0.09
+ LoRA	0.38	0.33	0.36	0.74	0.61	0.16
+ MGS	0.28	0.57	0.33	0.78	0.54	0.31
+ QDTS	**0.65**	0.52	**0.07**	<u>0.88</u>	**0.32**	**0.64**
Ideal	*1.00*	*0.00*	*0.00*	*1.00*	*0.00*	*1.00*

Efficiency Table 3 compares inference and training costs under a common setup. QDTS generates the fewest output tokens at every search depth, as the injected contexts reduce the deliberation required before each action. These contexts incur additional prefill cost, which largely offsets the reduction in decoding for 1-hop queries, resulting in similar latency. For deeper queries, however, the reduction in sequential decoding becomes more substantial, yielding lower latency than GRPO. Training time increases moderately because the contexts are processed in every rollout and the analyzer is invoked after each search, but shorter generations keep the overall overhead comparable to prior methods. Overall, QDTS achieves substantial accuracy gains at only a *small* additional cost.

Table 3: **Efficiency comparison.** Latency is wall-clock time per question (batch size 1, H100); tokens denote generated output tokens; training time is measured over 150 GRPO steps on 4×H100.

	Hop1		Hop2		Hop3+		Train
Method	Lat	Tok	Lat	Tok	Lat	Tok	(h)
StepSearch	**13.0**	121	**17.8**	153	21.1	231	19.0
AutoSearch	14.2	123	19.7	176	22.5	262	22.4
Search-R1 (GRPO)	15.2	120	21.3	159	26.0	206	18.4
+ LoRA	14.1	110	19.5	153	**21.0**	221	**16.5**
+ MGS	15.6	120	22.2	164	23.2	240	24.7
+ QDTS	15.2	**96**	19.2	**134**	22.2	**188**	21.5

4.3 Ablation Study and Further Analysis

Ablation on the Components of QDTS We add the three components of QDTS one at a time: the pre-rollout context C_{pre}, the mid-rollout context C_{mid}, and context optimization with GEPA. "+ C_{pre} + C_{mid} + GEPA" is the genuine setup of QDTS. Table 4 shows that C_{pre} alone already improves over Search-R1 (GRPO), while adding C_{mid} on top of it changes little. The gap closes only once the contexts are optimized with GEPA, which gives the best result on every benchmark. The retrieval-fit signal is therefore informative but not directly actionable. It becomes useful only when the injected context is tuned so that the policy's subsequent search decisions follow it.

Table 4: Ablation on the components of QDTS.

	HotpotQA		2Wiki		MuSiQue		Bamboogle		Avg.	
Setting	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Search-R1 (GRPO)	39.0	47.5	45.8	53.5	21.7	27.8	37.6	51.2	36.0	45.0
+ C_{pre}	44.4	51.4	49.0	55.1	25.5	30.7	39.2	53.6	39.5	47.7
+ C_{pre} + C_{mid}	46.0	52.7	47.7	53.2	25.4	**33.5**	38.4	51.9	39.4	47.8
+ C_{pre} + C_{mid} + GEPA	**49.8**	**54.3**	**52.5**	**57.9**	**26.8**	33.1	**42.4**	**58.5**	**42.9**	**51.0**

Ablation on the Class Coarseness We vary how finely questions are partitioned into structural classes. *Single* uses one shared context; *Mid-grained* uses the six structural patterns crossed with three hop buckets (18 classes); *Fine-grained* further splits each class by six answer-entity domains (108 classes in total). Mid-grained is the genuine setup of QDTS. Table 5 shows a clear optimum at mid-grained classes, inferring that class contexts help to the extent that they group questions that have the same demand, not to the extent that they are specific.

Table 5: Ablation of class coarseness of QDTS.

	HotpotQA		2Wiki		MuSiQue		Bamboogle		Avg.	
Context Setting	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Single	43.7	47.7	48.2	55.7	23.1	28.0	38.4	47.1	38.4	44.6
Single + GEPA	45.3	52.2	48.7	54.5	**27.0**	31.3	40.8	54.1	40.5	48.0
Fine-grained + GEPA	42.9	46.1	46.8	50.1	22.2	26.7	36.8	48.6	37.2	42.9
Mid-grained + GEPA	**49.8**	**54.3**	**52.5**	**57.9**	26.8	**33.1**	**42.4**	**58.5**	**42.9**	**51.0**

Ablation on the Backbone, the Retriever, and the Training Coreset QDTS improves over GRPO by a similar margin on three backbones from different families and with a stronger dense retriever, and it is what makes diversity-based coreset selection pay off, since the two sampling strategies perform almost identically without steering and diversity sampling becomes clearly better with it. Appendix I reports the three ablations.

4.4 Case Study

Examples of Conflicts Avoided by QDTS Figure 3 contrasts reasoning traces from GRPO and QDTS. Naive GRPO tends to issue the full question verbatim as a single query and integrate the retrieved passages without resolving intermediate entities, often leading to incorrect answers from off-target evidence. In contrast, the query-side and document-side contexts in QDTS steer the agent away from premature integration, triggering either a re-search with refined keywords for the same target or a next-search that resolves the remaining dependency. These examples illustrate how QDTS intervenes at the point where the search trajectory begins to deviate, rather than allowing the failure to propagate to the final answer. Appendix J presents two further cases.

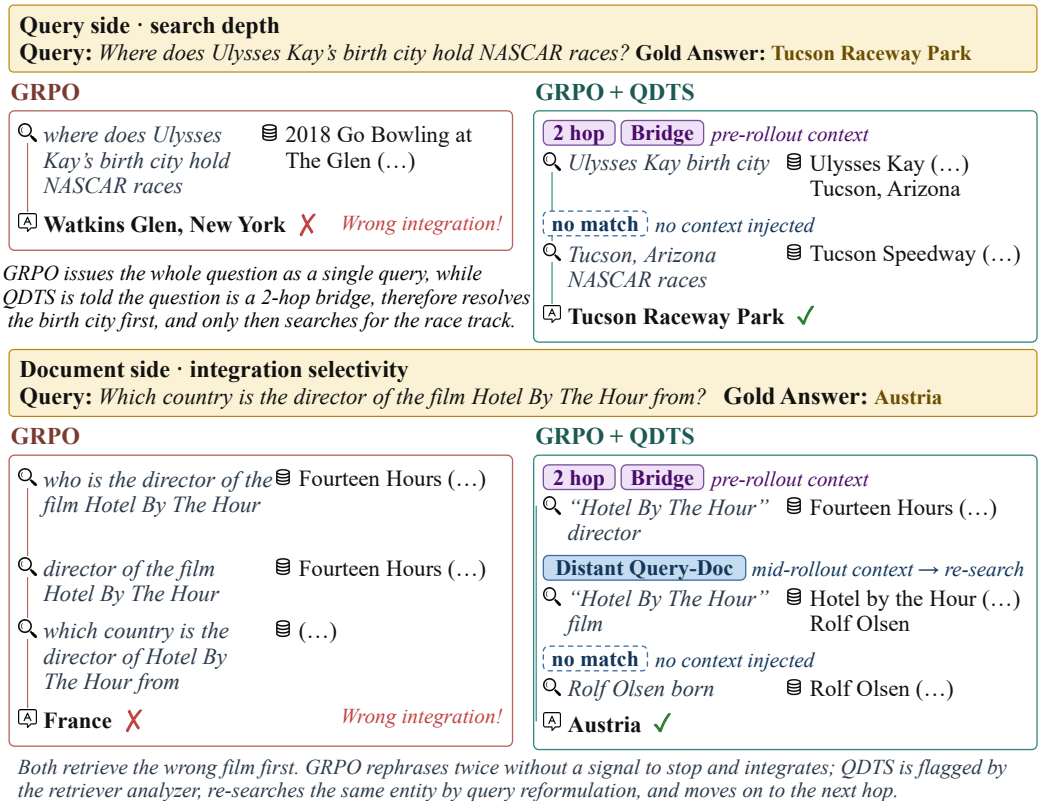


Figure 3: Qualitative examples of conflicts avoided by QDTS. Each case pairs a GRPO trajectory (left) with a GRPO+QDTS trajectory (right) on the same question. *re-search*: the same entity is queried again with a revised query, often after mid-rollout context from the analyzer; a *distant* query-doc triggers a same-entity re-search in the second case.

5 Conclusion

We theoretically showed how two forms of homogenization—query-side and document-side—emerge when training agentic RAG with RLVR. We introduced *query-and-document trajectory steering*, which combines *query-side pre-rollout steering* to steer trajectories according to query-specific search demands and *document-side mid-rollout steering* to adapt search decisions to retrieval outcomes. Experiments across diverse QA benchmarks demonstrate that QDTS mitigates both forms of homogenization and consistently improves exact matching answer accuracy by 9.2% on average.

AI USE STATEMENT

We used generative AI tools as general-purpose writing assistants for grammar and phrasing in the manuscript, and as coding assistants for routine implementation and plotting scripts. Generative AI was not used to generate research ideas, to design the method, to produce or modify any experimental result, or to write the analysis of results. All AI-assisted code was reviewed and tested by the authors, and all AI-assisted text was rewritten and verified by the authors. Retrieval corpora and benchmark datasets are standard public releases and were not AI-generated. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts.

ETHICS STATEMENT

This work trains and evaluates retrieval-augmented agents on public question-answering benchmarks (Natural Questions, HotpotQA, MuSiQue, 2WikiMultihopQA, PopQA, and Bamboogle) over a public Wikipedia corpus. No human subjects, private data, or personally identifiable information beyond what these public releases already contain are involved, and no new data are collected. The method conditions a policy on short textual contexts and does not alter the corpus or the reward, so it introduces no new channel for harmful content beyond those of the underlying language model and retriever; the usual caveats of retrieval-augmented generation, in particular the possibility of confidently integrating an incorrect passage, apply to our baselines and to our method alike, and Table 2 shows that our method reduces rather than increases integration of off-target passages. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

The Agentic RAG MDP formulation and objectives are given in Section 3.1. The parser rules, retrieval statistics and exact context-injection format are specified in Section 3.3 and Appendices D, E, and F, including the full text of every context string in both the pre-rollout and mid-rollout pools. Training hyper-parameters, retrieval configuration, context limits, and the coreset construction procedure are in Section 4 and Appendix H. Anonymized code for reproducing the training and evaluation is available at `https://anonymous.4open.science/r/QDTS-E3D8`.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 2026, pp. 8479–8565, 2026.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 2024, pp. 9112–9141, 2024.

Yiqun Chen, Erhan Zhang, Tianyi Hu, Shijie Wang, Zixuan Yang, Meizhi Zhong, Xiaochi Wei, Yan Gao, Yi Wu, Yao Hu, et al. JADE: Bridging the Strategic-Operational Gap in Dynamic Agentic RAG. *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The Entropy Mechanism of Reinforcement Learning for Reasoning Language Models. *arXiv preprint arXiv:2505.22617*, 2025.

Wenlong Deng, Yushu Li, Boying Gong, Yi Ren, Christos Thrampoulidis, and Xiaoxiao Li. On Group Relative Policy Optimization Collapse in Agent Search: The Lazy Likelihood-Displacement. *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

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. *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large Language Models can Self-Correct with Tool-Interactive Critiquing. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 2024, pp. 57734–57811, 2024.

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.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5-Coder Technical Report. *arXiv preprint arXiv:2409.12186*, 2024.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 7036–7050, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. *Proceedings of the Conference on Language Modeling (COLM)*, 2025.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, et al. Tulu 3: Pushing Frontiers in Open Language Model Post-Training. *Proceedings of the Conference on Language Modeling (COLM)*, 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 *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 9802–9822, 2023.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and Narrowing the Compositionality Gap in Language Models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5687–5711, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 36:68539–68551, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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 preprint arXiv:2402.03300*, 2024.

Jianghan Shen, Siqi Luo, Xinyu Cheng, Jing Xiong, Yue Li, Jiyao Liu, Jiashi Lin, Yirong Chen, and Junjun He. CuSearch: Curriculum Rollout Sampling via Search Depth for Agentic RAG. *Proceedings of the Conference on Language Modeling (COLM)*, 2026.

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.

Jingbo Sun, Wenyue Chong, Songjun Tu, Qichao Zhang, Yaocheng Zhang, Jiajun Chai, Xiaohan Wang, Wei Lin, Guojun Yin, and Dongbin Zhao. AutoSearch: Adaptive Search Depth for Efficient Agentic RAG via Reinforcement Learning. In *Findings of the Association for Computational Linguistics (ACL)*, pp. 28059–28079, 2026.

Yunbo Tang, Chengyi Yang, Shiyu Liu, Zhishang Xiang, Zerui Chen, Qinggang Zhang, and Jinsong Su. SAAS: Self-Aware Reinforcement Learning for Over-Search Mitigation in Agentic Search. *arXiv preprint arXiv:2605.29796*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop Questions via Single-Hop Question Composition. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 10:539–554, 2022.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text Embeddings by Weakly-Supervised Contrastive Pre-Training. *arXiv preprint arXiv:2212.03533*, 2022.

Fang Wu, Weihao Xuan, Heli Qi, Aaron Tu, Ximing Lu, Li Erran Li, and Yejin Choi. DeepSearch: Overcome the Bottleneck of Reinforcement Learning with Verifiable Rewards via Tree-Based Search. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 2026, pp. 95559–95584, 2026a.

Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. HiPRAG: Hierarchical Process Rewards for Efficient Agentic Retrieval Augmented Generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 2026, pp. 82317–82347, 2026b.

Yifei Xia, Fangcheng Fu, Wentao Zhang, Jiawei Jiang, and Bin Cui. Efficient Multi-Task LLM Quantization and Serving for Multiple LoRA Adapters. *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 37:63686–63714, 2024.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective Retrieval Augmented Generation. *arXiv preprint arXiv:2401.15884*, 2024.

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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 38:113222–113244, 2026.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient Surgery for Multi-Task Learning. *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 33:5824–5836, 2020.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, et al. Process vs. Outcome Reward: Which is Better for Agentic RAG Reinforcement Learning. *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 38:58701–58729, 2026.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. StepSearch: Igniting LLMs Search Ability via Step-Wise Proximal Policy Optimization. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

A Analysis of Query-Side Behavioral Homogenization

Equation 5 weights each query's continuation advantage at depth h by $w_\theta(x,h) \propto d^{\pi_\theta}(s_h \mid x)$, and the reward that weight can carry is limited in turn by $\varepsilon_\theta(x)$ through Equation 3. The three stages below act on those two factors in order: the first shrinks the reward a deep query can carry, the second shrinks the visitation that would let it be carried, and the third sets the term to zero.

Stage 1: the reward a deep query can carry decays geometrically in its depth Let $p \in (0,1)$ be the per-hop retrieval success probability, that is, the probability that a well-formed sub-query at hop j returns the evidence required to resolve hop j. A trajectory in $\mathcal{T}^*(x)$ requires every hop to land, so under an optimistic independence assumption across hops,

$$\varepsilon_\theta(x) \;=\; \sum_{\tau \in \mathcal{T}^*(x)} p_\theta(\tau \mid x) \;\lesssim\; p^{\,H^*(x)}, \tag{13}$$

which is Equation 4. Bridge chains are worse than the independence assumption suggests, because the entity resolved at hop j becomes the subject of the sub-query issued at hop $j+1$: a single mis-resolution does not merely fail to add evidence, it misdirects every subsequent retrieval, so failures compound along the chain rather than accumulating independently. Only p itself is a property of the frozen retriever and independent of θ; the bound it implies is what limits, through Equation 3, the update any single deep query can contribute at a shared decision point. In a training mixture where shallow questions predominate, the terms of Equation 5 that carry $\delta(x,h) > 0$ are therefore the ones with the least reward behind them, and the aggregate turns negative beyond a band $\bar{H}$ while remaining correct for each query taken alone.

Stage 2: the visitation that would carry it shrinks, and does so self-reinforcingly That an action has higher aggregate advantage does not by itself explain why the policy becomes *deterministic* about it. The mechanism is supplied by the entropy dynamics of softmax policies. Cui et al. (2025) show that under a small learning rate the one-step change in policy entropy is governed by a covariance,

$$\mathcal{H}(\pi_\theta^{k+1}) - \mathcal{H}(\pi_\theta^{k}) \;\approx\; -\operatorname{Cov}_{a\sim\pi_\theta}\big(\log \pi_\theta(a),\, \Delta z(a)\big), \qquad \Delta z \propto A^{\pi_\theta}, \tag{14}$$

so entropy falls fastest for actions with both high probability and high advantage, and rises only for rare actions with high advantage. Early termination after a shallow search is precisely the former: it is already the modal continuation under the base model, and by Stage 1 it carries positive aggregate advantage on a shallow-dominated batch. Acting on $\bar{\delta}_\theta(h)$ therefore lowers $\pi_\theta(a_{\text{search}} \mid s_h)$ sharply for $h \geq \bar{H}$, which lowers $d^{\pi_\theta}(s_h \mid x)$ at those depths and shifts w_θ further toward the shallow queries, so each update makes $\bar{\delta}_\theta(h)$ more negative than before. Cui et al. (2025) further report the empirical relation $R = -a\exp(\mathcal{H}) + b$ between reward and entropy, meaning that reward improvement is *purchased with* behavioral diversity: the optimizer is incentivized to spend exactly the depth variability an adaptive search policy requires. The deep-search mode is not discarded because it is known to be wrong; it is discarded because it is rare and, at the margin, unprofitable.

Stage 3: the term leaves the aggregate entirely Stages 1 and 2 leave the policy concentrated on a narrow band around $\bar{H}$. From this point the estimator removes the signal that would reverse the concentration. Group-relative optimization samples, for each question x, a group of n rollouts $\{\tau_i\}_{i=1}^n \sim \pi_\theta(\cdot \mid x)$ and forms

$$\hat{A}_i = \frac{R(\tau_i) - \mu_x}{\sigma_x}, \qquad \mu_x = \frac{1}{n}\sum_j R(\tau_j), \qquad \sigma_x = \operatorname{std}_j\big(R(\tau_j)\big). \tag{15}$$

For a question with $H^*(x) > \bar{H}$, every rollout now truncates before the evidence chain is complete and receives the same terminal reward, so $\sigma_x \to 0$, $\hat{A}_i \equiv 0$, and

$$\nabla_\theta J(\theta \mid x) = \mathbb{E}\Big[\textstyle\sum_t \hat{A}_i\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Big] = 0. \tag{16}$$

The queries carrying the strongest evidence for deeper search are no longer outweighed in Equation 5; their terms are absent from it. The support vanishes with the gradient: because d^{π_θ} is policy-induced, the agent ceases to collect experience at the very states where deeper continuation would be evaluated, and gradient starvation together with support shrinkage makes the shallow mode an absorbing state of the optimization.

Relation to concurrent work The degenerate group of Equation 16 is the one DAPO identifies and discards by dynamic sampling (Yu et al., 2026). A concurrent work attacks it from the sampling side rather than the conditioning side: Shen et al. (2026) confine advantage estimation to a depth-selected subset of an enlarged rollout pool, excluding groups in which no trajectory is retained. Both treat the composition of the advantage group as the object to be repaired, whereas QDTS changes the distribution the group is drawn from. A separate collapse mode of GRPO in agent search, lazy likelihood displacement (Deng et al., 2026), arises when incorrect trajectories that share a prefix with correct ones push the latter's likelihood down; it is a token-level counterpart of the state-level interference in Equation 5, and its regularizer is complementary to conditioning.

B Analysis of Document-Side Behavioral Homogenization

Equation 7 bounds the corrective mass by the product of a detection rate that the policy controls and a recovery rate that the retriever fixes. This appendix gives the conditions under which the update drives the first factor down.

The state cannot supply g_t; only the passage text can The retriever returns k passages for every sub-query, so a miss produces the same transition signature as a hit. There is no null return and no score that the state records. Coverage must therefore be inferred from the passages themselves, and that inference is weak exactly where it matters, since a near-miss passage shares the surface features of a hit, namely the same entity and the same relation type, while lacking the fact under test. In addition, the accumulated context often exceeds the effective capacity of the model, so that a fact present in the context is not reliably attended to. States with $g_t = 0$ and $g_t = 1$ are therefore often indistinguishable to the policy, and a single integrate-or-search rule is applied to both.

Integration wins on average, and the update makes it win more Under a single rule, integration yields the higher return on average, because almost every correct answer is produced by integrating retrieved passages, whereas an additional search lengthens the trajectory and, by Equation 4, lowers the probability of a correct answer. The update therefore reinforces integration in both coverage classes. The trajectories that detect a miss and recover from it, which are already rare, become rarer with each update, so the detection rate in Equation 7 falls, the corrective mass $\varepsilon_\theta^{\mathrm{corr}}(x)$ vanishes in the sense of Equation 3, and S approaches zero. The loop closes in the visitation as well. Because d^{π_θ} is policy-induced, states that are reached after a successful recovery become rarer, so the agent stops collecting the experience that would show recovery paying off. The document side thus mirrors the query side, with the detection rate in place of the search depth and the coverage indicator in place of the gold depth.

Process rewards relocate the judgment rather than removing it A process reward can score intermediate retrievals against $\mathcal{D}^*$, and the baselines we compare do exactly that, so g_t is available to the reward during training. Two limitations remain. First, the reward can only re-weight trajectories already produced by the policy, so Equation 7 continues to bound the number of corrective trajectories available for learning; a denser signal cannot compensate for a vanishing corrective set. Second, no annotation is available at inference time, so whatever the reward teaches must ultimately be internalized by the policy through its own reading of the retrieved passages—a capability that the first paragraph identifies as weak. Correction-based methods such as CRAG (Yan et al., 2024) and Self-RAG (Asai et al., 2024) address this limitation by supplying the judgment externally at each step, at the cost of additional inference compute. Document-side mid-rollout steering instead provides a cheaper statistic computed directly from the retriever's own embeddings, as described in §3.3.2.

C Overall Process of QDTS Visualized

Figure 4 traces the two stages of §3.3 on two questions, a 1-hop lookup ("What is Don Quixote's horse's name?") and a 3-hop bridge ("What company owns the label of the Top Gun theme guitarist?"). The top-left panel shows the gold search trajectories, which are the behaviors that a demand-adaptive policy should produce: one search for the first question and three chained searches for the second. The top-right panel shows what reward-only training produces instead. The realized depth is close to two for both questions, so the lookup question over-searches and the bridge question stops short, and

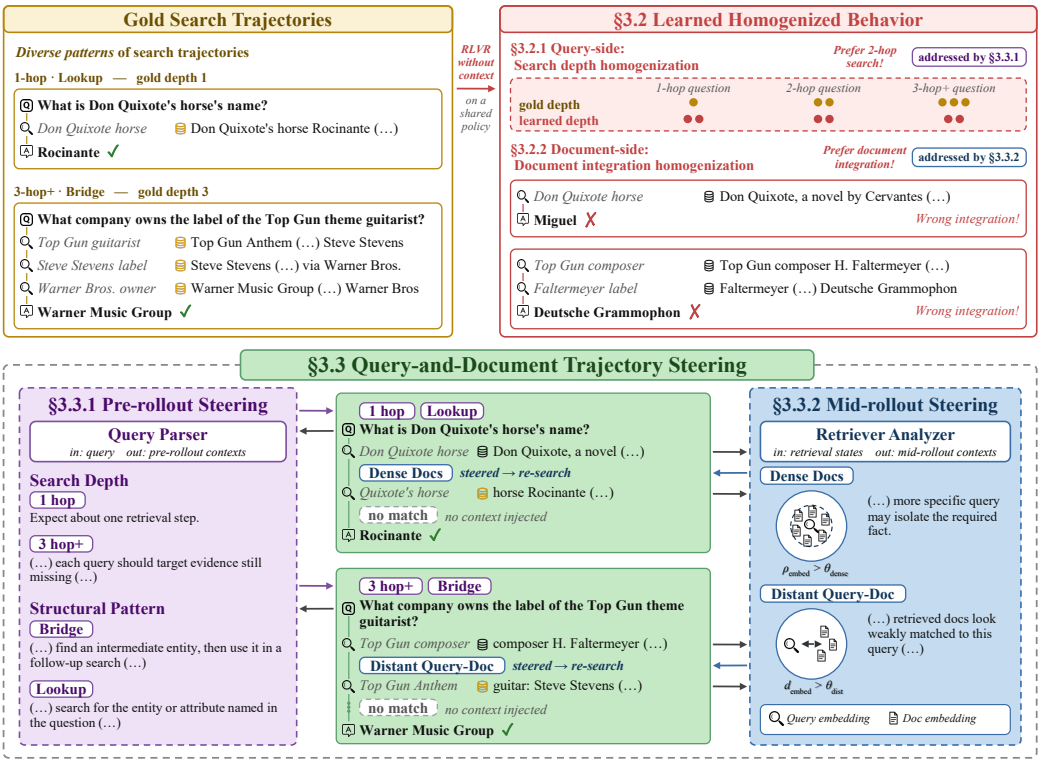


Figure 4: Overall process of QDTS in more detail.

in both cases the agent integrates the first retrieved passage even though it is off target, which yields the wrong answers *Miguel* and *Deutsche Grammophon*.

The lower panel shows the same two questions under QDTS. Before the rollout, the query parser assigns each question a class from its surface form, 1-hop / Lookup for the first and 3-hop+ / Bridge for the second, and the corresponding pre-rollout context C_{pre} is prepended to the prompt; the context pool on the left lists the guidance strings that each tag contributes. During the rollout, the retriever analyzer on the right inspects every retrieval. For the first question, the initial search returns a passage about the novel rather than the horse, and the inter-document density ρ_{embed} exceeds its threshold, so the analyzer appends a mid-rollout context C_{mid} that describes the retrieval as dense and the agent issues a more specific query, which returns *Rocinante*. For the second question, the first search returns the film's composer rather than its guitarist, the query–document distance d_{embed} exceeds its threshold, and the appended context leads the agent to re-search for the anthem itself before it resolves the remaining hops to *Warner Music Group*. Retrievals that trigger no state receive no context, as the "no match" rows indicate. In both cases, the depth of the trajectory is set by the class context and the decision to re-search is set by the retrieval state, which are the two behaviors that §3.2 showed a retrospective reward cannot restore on its own.

D Query Parser

The parser ψ is a deterministic, rule-based tagger that assigns each query a structural pattern and a search depth from surface cues alone. It involves no learned component and no language-model call, and it runs in approximately 0.11 ms per query on a CPU for hop assignment and 0.15 ms for hop and pattern together. Measured against the gold class labels, the parser assigns the correct class to 78.2% of HotpotQA, 86.9% of NQ, 86.4% of MuSiQue, and 94.5% of 2Wiki questions, which is 86.5% on average. We describe the two taggers in turn.

Structural Pattern The pattern tagger tests the query against the classes in Table 6 in order of priority and returns the first class whose cues match. The CompareOrder, CompareSame, and BridgeWhere classes are recognized directly from lexical cues; when none of them matches, the

pattern is determined by the hop count, with three or more hops mapped to CHAIN, two hops to BRIDGE, and one hop to LOOKUP.

Table 6: Search-pattern classes and the surface cues that trigger them, in priority order.

Priority	Class	Cues
1	COMPAREORDER	A two-option construction (*or*, *between … and*, *out of … and*) together with an ordering cue (*earlier*, *older*, *later*, *younger*, or *first* outside a series ordinal).
2	COMPARESAME	*same* with *both* or *and*; *have in common*; *same nationality*; *same country*.
3	BRIDGEWHERE	*where* with *located*, *situated*, *based*, *headquartered*, or *founded*; or a place noun followed by *where*.
4	CHAIN	No lexical match; hop count ≥ 3.
5	BRIDGE	No lexical match; hop count $= 2$.
6	LOOKUP	No lexical match; hop count $= 1$.

Search Depth The hop tagger first computes a lower bound in $\{1, 2, 3\}$ from the number of relative clauses (*who*, *which*, *that*, *where*, *whose*), role nouns, non-attributive *of*-links, possessives, WH-words, and the query length. It then applies an ordered set of rules that can raise the estimate. A query is assigned three hops if it contains at least three *of*-links combined with a relative clause or role noun, or two relative clauses or WH-words. A query is assigned two hops if it contains a possessive attribute such as *X's birthplace*, a relational phrase such as *directed by* or *born in*, the phrase *have in common*, or at least 22 tokens. Queries that match none of these rules are assigned one hop. The resulting count is bucketed into $\{1, 2, 3^+\}$ for use as the depth attribute of $C_{\mathrm{pre}}(x)$.

E CONTEXT INJECTION

This appendix lists the contexts that the two stages inject. The pre-rollout context $C_{\mathrm{pre}}(x) = c_{\psi(x)}$ is rendered once into the user prompt before the question, as a one-line class tag followed by the guidance strings of the active pattern and hop tags (Table 7). The mid-rollout context $C_{\mathrm{mid},t}$ is appended after the observation of each search and is not carried into later turns (Table 8). Both tables list the hand-written seed string of each tag or state next to the string that context optimization (Equation 9 for C_{pre} and Equation 12 for C_{mid}) produced from it. For C_{pre}, the optimization kept every seed and added an addendum to the multi-hop classes. For C_{mid}, the optimization kept the "Distant Query-Doc" and "Dense Docs" strings and removed the "Close Query-Doc" and hedged strings as well as the class reminders, which we discuss below. Token counts use the Qwen2.5-7B tokenizer. On the training set, $|C_{\mathrm{pre}}|$ is 126 tokens on average, and $|C_{\mathrm{mid},t}|$ is about 20 tokens when a state is reported and zero otherwise.

Strings Removed by Optimization The seed set for C_{mid} initially contained a string for every retrieval state and a class reminder, a second instruction for the pattern of $\psi(x)$ that was appended after every search. The optimization of Equation 12 retained only the "Distant Query-Doc" and "Dense Docs" strings. The removal is consistent with the role of the mid-rollout stage. The "Close Query-Doc" and hedged strings describe retrievals for which the policy already takes the appropriate action, and emitting them adds tokens without raising the corrective mass. The class reminder prescribes the next action rather than describing the retrieval, and its effect on the policy was too strong. Under the reminder, the agent followed the prescribed action even when the retrieved passages already resolved the sub-question, which increased over-search on the Bridge and Chain classes. The pre-rollout context already carries the same pattern information at s_0, and repeating it at every turn added a constraint without adding information.

F FITTING THE RETRIEVER ANALYZER

Rather than fixing the thresholds Ω_{stat} at absolute values, we set them as quantiles of each statistic's empirical distribution, estimated on a held-out set of retrievals. This choice is necessary because cosine similarities live on a scale that depends on both the embedding model and the corpus, so a

Table 7: **Pre-rollout context C_{pre} by class tag.** The prompt carries the line "Question type: [Structural pattern] × [Search depth]" followed by the guidance strings of both tags. The seed strings are hand-written; the addenda are produced by context optimization and are applied only to the multi-hop patterns (Bridge, BridgeWhere, Chain, CompareSame, CompareOrder) with 2-hop or 3-hop+ tags.

Tag	Seed string	After optimization (added)
Structural pattern		
LOOKUP	Direct fact lookup; one well-formed search is often enough.	—
BRIDGE	Find an intermediate entity, then search for the final attribute.	Do not paraphrase the full question; name the missing linked entity from the prior documents and search for it.
BRIDGEWHERE	Resolve the related entity, then search for its place.	Query the concrete place or region already in evidence; avoid restating the full question.
CHAIN	Three or more hops; search sequentially until the question is answerable.	Each hop unlocks the next missing fact; do not rephrase the entire question at every step.
COMPARESAME	Retrieve the property for each entity, then decide whether they match.	After one side, search for the same property of the other entity, then compare.
COMPAREORDER	Retrieve comparable values, then decide which is larger or earlier.	After one side's value, query the other side for the same kind of value, then decide the order.
Search depth		
1-hop	Expect about one retrieval step.	—
2-hop	Expect about two retrieval steps with an intermediate entity.	Hop 1 covers one aspect; hop 2 should retrieve documents that hop 1 did not.
3-hop+	Expect three or more sequential retrieval steps.	Each next query targets evidence that is still missing after the prior hops.

Table 8: **Mid-rollout context $C_{\mathrm{mid},t}$ by retrieval state and class reminder.** Each retrieval-state string is emitted by the analyzer when the corresponding statistic crosses its quantile threshold. The seed strings are hand-written; the right column shows what remained after optimization, where "removed" means that the optimization emits nothing for that state, and "nothing added" means the seed string is kept as is.

Tag	Trigger	Seed string	After optimization (added)
Retrieval state (from ϕ)			
Distant Query-Doc	d_{embed} high	The top retrieved documents look weakly matched to this query; a revised search may find better coverage.	"nothing added"
Hedged	d_{embed} borderline	The top retrieved documents look only moderately matched to this query; useful facts may be partial.	"removed"
Close Query-Doc	d_{embed} low	The top retrieved documents look well matched to this query; the answer or the next link may already be in them.	"removed"
Dense Docs	ρ_{embed} high	The retrieved documents look densely packed with many details; a more specific query may isolate the needed fact.	"nothing added"
Class reminder (from $\psi(x)$)			
Bridge / Chain	every search	From the documents above, name the missing linked entity or place that is not in the question, and search for it. Do not rephrase the full question.	"removed"
Compare	every search	If you have the property for one entity, search for the same property of the other entity named in the question.	"removed"
Lookup	every search	If the answer is already in the documents, answer now; do not search again.	"removed"

threshold that works for one retriever would not transfer to another. The seed set for $\{c_r\}$ included a context for statistics that fall just over a threshold, which described the retrieval as borderline instead of declaring it a miss. The optimization of Equation 12 removed it together with the context for well-matched retrievals, and the retained contexts describe the retrieval state in non-forcing terms that leave the decision to the policy. The seed and retained strings for each state are listed in Appendix E.

G GROUP-RELATIVE ESTIMATOR UNDER CONTEXT CONDITIONING

We train the policy with group-relative optimization, sampling n rollouts per query under the conditioned policy,

$$\hat{\nabla}_\theta J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T_i} \hat{A}_i \, \nabla_\theta \log \pi_\theta\big(a_{i,t} \mid \tilde{s}_{i,t}\big), \qquad \hat{A}_i = \frac{R(\tau_i) - \mu_x}{\sigma_x}. \tag{17}$$

Two properties of this estimator deserve attention. First, $C_{\mathrm{pre}}(x)$ is deterministic given x and is therefore identical across the group. It shifts the distribution from which the group is sampled without affecting the group-relative baseline. Second, $C_{\mathrm{mid},i,t}$ depends on the retrievals that are realized and therefore varies within the group. It introduces behavioral variance precisely in those states where retrieval has failed, which are the states where σ_x would otherwise collapse to zero and Equation 16 would apply.

H IMPLEMENTATION DETAILS

The agent uses a frozen E5 retriever (Wang et al., 2022) over the Wikipedia-18 corpus (Karpukhin et al., 2020), retrieving the top 3 passages at each search step. Search and answer formatting follows the Search-R1 (Jin et al., 2025) and StepSearch (Zheng et al., 2025) multi-turn templates, with state masking on retrieved passages so that retrieved tokens contribute no gradient. The coreset is built per source by a rarity-first covering procedure over the intent templates $t(q) =$ [structural pattern] $\times$ [search depth], selecting rare classes first until each source contributes its share of the total, so that the 18 classes are populated rather than following the natural frequency of the mixture. Hop-count and pattern labels are retagged from the gold supporting documents before this selection, which affects only the composition of the training set, not the class assignment used to build contexts and the reward. EM and token F1 are computed after string normalization, against multiple references where a dataset provides them. Experiments have been conducted on 4 NVIDIA H100 GPUs.

Training hyper-parameters We train with GRPO (Shao et al., 2024), sampling 8 rollouts per question with a training batch of 16 questions, that is, 128 trajectories per step. Each rollout batch is consumed in one PPO epoch with a mini-batch of 8 questions and a micro-batch of 4. The actor uses a constant learning rate of 5×10^{-7}, a clip ratio of 0.2, an entropy coefficient of 0.001, and gradient clipping at 1.0. The KL penalty against the reference policy uses the low-variance estimator with coefficient 0.0014. Each rollout permits up to 5 search–answer turns, and retrieved passages are masked from the training loss. Context limits are 4,096 tokens for the prompt, 800 for the response and 1,024 for the observation at each turn.

I ABLATION ON THE BACKBONE, THE RETRIEVER, AND THE TRAINING CORESET

Ablation on the Model Types We repeat the comparison on three backbones from different families. Table 9 shows that QDTS improves over GRPO on all of them, by a similar margin in each case, and that the absolute best result is obtained with the strongest backbone. The mechanism is thus not tied to a particular model, and its benefit is complementary to the reasoning capability of the underlying backbone.

Table 9: Ablation on different type of models.

		HotpotQA		2Wiki		MuSiQue		Bamboogle		Avg.	
Model	Setting	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Qwen2.5-7B	GRPO	39.0	47.5	45.8	53.5	21.7	27.8	37.6	51.2	36.0	45.0
Qwen2.5-7B	QDTS	49.8	54.3	52.5	57.9	26.8	33.1	42.4	58.5	42.9	51.0
Qwen3-8B	GRPO	42.6	48.2	50.5	56.8	24.4	31.6	44.8	58.8	40.6	48.9
Qwen3-8B	QDTS	**50.5**	**61.3**	**53.9**	**58.6**	**27.6**	**35.3**	**47.2**	**59.6**	**44.8**	**53.7**
Llama-3.1-8B	GRPO	40.6	46.2	42.9	48.1	22.6	28.8	38.4	46.2	36.1	42.3
Llama-3.1-8B	QDTS	46.8	53.3	46.0	52.5	25.0	31.5	43.2	56.5	40.2	48.5

Ablation on the Retriever Types We include a stronger dense retriever BGE-M3 in addition to the default dense retriever E5. Table 10 shows that the stronger retriever raises both the baseline and QDTS, and that the improvement from QDTS is essentially unchanged in size. While a stronger retriever improves what evidence is retrieved, steering improves how the agent acts on that evidence.

Table 10: Ablation on different type of retrievers.

		HotpotQA		2Wiki		MuSiQue		Bamboogle		Avg.	
Retriever	Setting	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
E5	GRPO	39.0	47.5	45.8	53.5	21.7	27.8	37.6	51.2	36.0	45.0
E5	QDTS	49.8	54.3	**52.5**	57.9	26.8	33.1	42.4	**58.5**	42.9	51.0
BGE-M3	GRPO	41.1	49.3	45.6	54.6	23.2	28.6	39.2	50.8	37.3	45.8
BGE-M3	QDTS	**51.7**	**55.2**	52.1	**58.8**	**28.5**	**35.8**	**44.8**	56.0	**44.3**	**51.5**

Effectiveness of Coreset Selection by QDTS We compare diversity-based coreset sampling against random entity sampling, with and without QDTS. Table 11 shows that without QDTS the two sampling strategies perform almost identically, while with QDTS diversity sampling becomes clearly better. Diverse training data is only useful to an agent that can learn different behaviors from it; without conditioning, the conflicting demands in a diverse batch cancel, and the extra diversity cannot be fully exploited.

Table 11: Coreset selection vs. random entity sampling.

QDTS	Train Coreset	HotpotQA		2Wiki		MuSiQue		Bamboogle		Avg.		Δ	
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
×	Random Sampling	37.7	47.9	46.6	49.8	21.2	29.1	39.2	51.5	36.2	44.6	–	–
×	Diversity Sampling	39.0	47.5	45.8	53.5	21.7	27.8	37.6	51.2	36.0	45.0	-0.2	+0.4
✓	Random Sampling	41.5	50.2	52.2	**59.1**	20.8	32.1	40.8	54.9	38.8	49.1	–	–
✓	Diversity Sampling	**49.8**	**54.3**	**52.5**	57.9	**26.8**	**33.1**	**42.4**	**58.5**	**42.9**	**51.0**	+4.1	+1.9

J Additional Case Studies

Figure 5 continues the case study of Figure 3 with two further examples. A 3-hop chain in which the analyzer flags the second hop and the agent recovers with a reformulated query (left), and a case in which the same mid-rollout context is injected but the agent judges the retrieved passage sufficient and proceeds to the next hop rather than re-searching (right), which shows that the context steers the decision without forcing it.

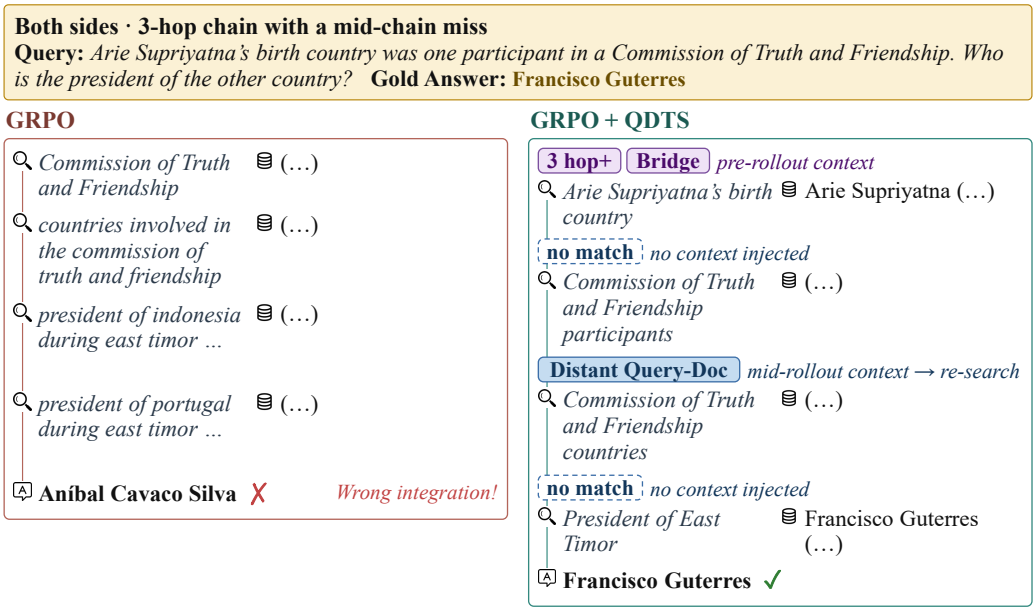


Figure 5: Additional qualitative examples of conflicts avoided by QDTS, in the format of Figure 3. In the first case the mid-rollout context triggers a same-entity re-search; in the second it is injected under the same cue, but the agent keeps the retrieved passage and moves to the next hop.