SEQUIN: LEARNING SELECTIVE QUERY INTERVENTION FOR AGENTIC RAG

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic retrieval-augmented generation (RAG) relies on intermediate search queries to acquire evidence during multi-step reasoning. Query optimization through backbone training requires parameter updates, while standalone rewriting may overlook the evolving reasoning–retrieval context. Moreover, rewriting already effective queries can degrade retrieval, motivating decisions about both how and when to intervene. We propose SEQUIN, a framework that learns how and when to rewrite intermediate queries while keeping the backbone frozen. Its trajectory-aware query rewriter is trained on beneficial rewrites identified through counterfactual rollouts. The intervention policy is initialized from counterfactual comparisons of intervention sequences and further trained through Intervention-Structured Policy Optimization (ISPO). Using final-answer rewards, ISPO combines forced-prefix exploration with episode-level and intervention-prefix advantage estimation. Across four multi-hop question-answering benchmarks, SEQUIN improves macro-average EM on Search-R1 by 2.5 percentage points and CEM on HiPRAG by 1.9 percentage points. On Search-R1, selective intervention reduces rewrite calls by approximately 30% on average across benchmarks relative to always rewriting, while improving macro-average EM. These results demonstrate that selective query intervention can improve frozen Agentic RAG systems without rewriting every intermediate query.

1 INTRODUCTION

Large language models (LLMs) rely on parametric knowledge that may be incomplete or outdated, limiting their ability to answer knowledge-intensive questions accurately (Zhu et al., 2025; Vu et al., 2024). Retrieval-augmented generation (RAG) mitigates these limitations by incorporating information retrieved from external knowledge sources (Lewis et al., 2020). However, conventional RAG typically performs a single retrieval step before generation, making it difficult to address new information needs that arise during multi-step reasoning (Trivedi et al., 2023). Agentic RAG instead interleaves reasoning with retrieval, using intermediate reasoning results to guide subsequent search queries and progressively gather evidence (Li et al., 2025b; Wu et al., 2026a;b).

In Agentic RAG, retrieving relevant evidence depends in part on how well intermediate queries reflect the agent's current information needs (Su et al., 2024; Jiang et al., 2023). Existing work has explored both backbone training and auxiliary rewriting to improve query quality. For example, SmartSearch (Wen et al., 2026) uses process rewards and query refinement during backbone training to improve the agent's query generation. In contrast, Rewrite-Retrieve-Read (Ma et al., 2023) trains a small query rewriter using feedback from a frozen LLM reader, allowing query optimization without updating the answer-generating model.

However, backbone-based query optimization requires parameter updates, making it unsuitable when the deployed model must remain frozen. An auxiliary rewriter can avoid these updates, but treating an intermediate query as a standalone input can overlook information from earlier reasoning and retrieval that is needed to interpret it. Even when this context is available, rewriting is not always beneficial: it can clarify an ambiguous query, but it can also change the meaning of an already effective one. In Figure 1(a), adding "country" and "by area" to "largest South America" clarifies the search target and helps retrieve relevant evidence. In Figure 1(b), however, adding "financial" to "capital of Canada" changes the intended meaning and redirects retrieval toward a different question.

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

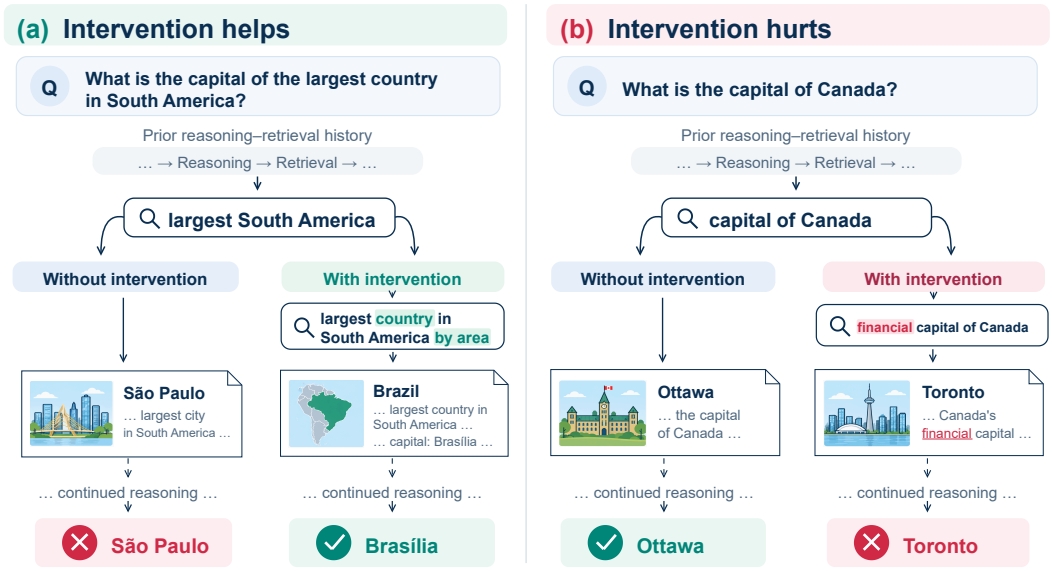

Figure 1: Illustrative examples motivating selective query intervention in Agentic RAG. **(a)** Clarifying an ambiguous intermediate query helps retrieve relevant evidence and supports a correct final answer. **(b)** A rewrite that changes the query's intended meaning redirects retrieval toward a different question and can lead to an incorrect answer. These examples motivate learning both how to rewrite queries and when to retain them.

These considerations motivate learning both *how* to refine intermediate queries using the reasoning–retrieval history and *when* to invoke the rewriter.

To address these questions, we propose SEQUIN (**SE**lective **QU**ery **IN**tervention), a plug-and-play framework that learns how and when to rewrite intermediate search queries while keeping the backbone language model frozen. As shown in Figure 2, SEQUIN combines a *Trajectory-Aware Query Rewriter* with an *Intervention Policy*. Before each retrieval step, the policy uses the original question, the available reasoning–retrieval history and the current query to decide whether to retain the query or invoke the rewriter. When invoked, the rewriter refines the query using the same context. In either case, retrieval proceeds with the selected query, and the resulting evidence is returned to the backbone for continued reasoning.

To learn *how* to refine queries in context, we construct counterfactual supervision by evaluating candidate rewrites from the same trajectory prefix. We let the frozen backbone continue with the evidence retrieved by each candidate and retain rewrites that improve final-answer correctness as training targets. To learn *when* to intervene, we freeze the trained rewriter and initialize the intervention policy using labels derived from counterfactual comparisons of intervention sequences. We further develop *Intervention-Structured Policy Optimization* (ISPO), a reinforcement learning method tailored to sequential query intervention. ISPO uses forced exploration of alternative early intervention sequences to obtain feedback that policy sampling may miss. Since earlier interventions can change later decision contexts, it also groups decisions by preceding intervention actions to construct step-level feedback from final-answer rewards.

Our main contributions are as follows:

- We introduce SEQUIN, a plug-and-play framework that learns how and when to rewrite intermediate queries in Agentic RAG using the reasoning–retrieval history, while keeping the backbone frozen.
- We construct counterfactual supervision based on improvements in final-answer correctness to train the rewriter. We also develop ISPO for intervention policy training, combining forced-prefix exploration with intervention-prefix advantage estimation.
- Experiments on four multi-hop question-answering benchmarks demonstrate that SEQUIN improves Search-R1 and HiPRAG, with additional validation on Search-R1-7B. Ablation studies

show that the Query Rewriter provides most of the performance gains. On Search-R1, the intervention policy reduces rewrite calls by approximately 30% on average across benchmarks relative to always rewriting, while improving macro-average EM.

2 Related Work

2.1 Agentic RAG

Existing Agentic RAG methods include training-free workflows and approaches that directly train the backbone. Training-free methods such as Search-o1 (Li et al., 2025a) guide iterative reasoning, search, and evidence integration through prompting or predefined interaction protocols, without updating the backbone model. Training-based approaches directly optimize the backbone's reasoning–retrieval policy (Li et al., 2025b). Search-R1 (Jin et al., 2025) uses outcome-level rewards derived from final task performance, while ReasonRAG (Zhang et al., 2026) and HiPRAG (Wu et al., 2026b) introduce process-level supervision to provide finer-grained feedback for intermediate retrieval decisions. Beyond backbone optimization, recent work also explores training other components of the Agentic RAG pipeline (Jiang et al., 2025). For example, Agentic-R (Liu et al., 2026) trains retrievers for agentic search using query relevance and final-answer correctness, while SPARKLE (Fang et al., 2026) trains external proxy models to coordinate retrieval decisions, query formulation, and evidence selection.

2.2 Query Optimization in RAG

Query optimization has been studied through both search agent training and dedicated rewriting modules. SmartSearch (Wen et al., 2026) uses process rewards to assess intermediate query quality and improves query generation through a three-stage training curriculum, while ExpandSearch (Zhao et al., 2025) uses reinforcement learning to generate multiple query variants for parallel retrieval. Among modular approaches, Rewrite-Retrieve-Read (RRR) (Ma et al., 2023) trains a lightweight rewriter using feedback from a frozen reader, while Q-PRM (Ye et al., 2025) models rewriting as a sequence of atomic refinement actions guided by process-level supervision. Shanian et al. (2026) study the complementarity of query reformulation strategies and evaluate a confidence-based router in simulation, using initial retrieval scores to decide whether to rewrite. SEQUIN learns how and when to rewrite intermediate queries before retrieval in multi-step Agentic RAG while keeping the backbone frozen. Its rewriter and intervention policy condition on the evolving reasoning–retrieval history and are trained using counterfactual supervision derived from final-answer correctness. The intervention policy is further optimized through reinforcement learning.

3 SEQUIN

SEQUIN learns how and when to rewrite intermediate search queries in Agentic RAG while keeping the backbone language model frozen. As illustrated in Figure 2, it consists of a *Trajectory-Aware Query Rewriter* and an *Intervention Policy*. The rewriter refines intermediate queries using the available reasoning–retrieval history, while the policy uses this history and the current query to decide whether to invoke the rewriter before retrieval. Evidence retrieved using the selected query is returned to the backbone to support subsequent reasoning.

We train the two modules in stages. The rewriter learns from beneficial rewrites identified through counterfactual rollouts. With the trained rewriter fixed, we initialize the intervention policy through supervised learning on labels derived from counterfactual comparisons of intervention action sequences. To further train the policy, we develop *Intervention-Structured Policy Optimization* (ISPO), a reinforcement learning method tailored to selective query intervention. ISPO uses intervention action prefixes to structure both rollout exploration and step-level advantage estimation. We first describe rewriter training, then introduce policy training, and finally present the inference procedure.

3.1 Learning to Rewrite from Counterfactual Outcomes

To train the rewriter, we seek supervision that reflects whether the evidence retrieved by a rewritten query helps the agent produce a better final answer. An observed reasoning–retrieval trajectory,

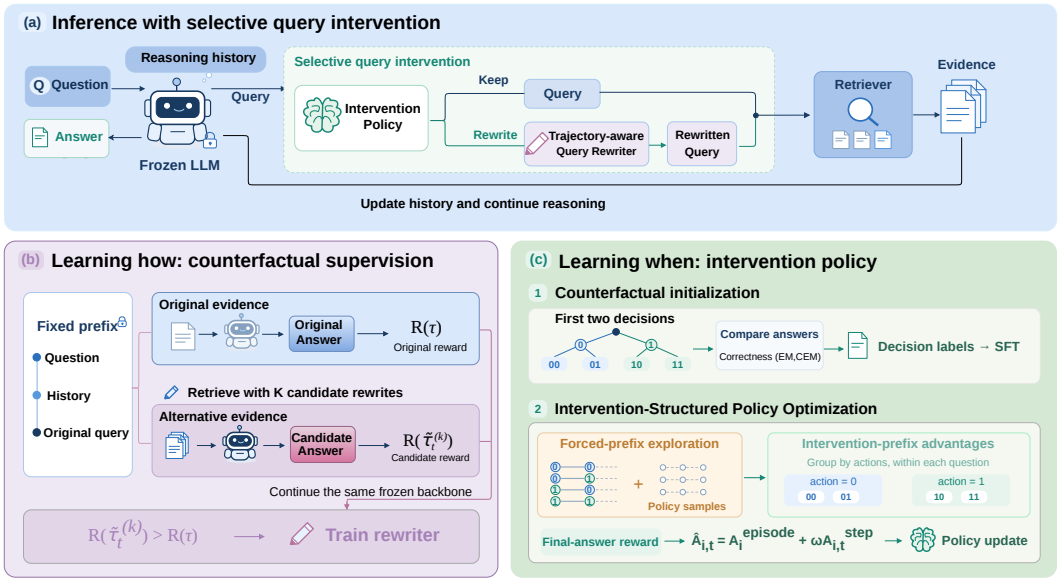

Figure 2: Overview of SEQUIN. **(a)** Selective query intervention before retrieval. **(b)** Query rewriter training via counterfactual supervision. **(c)** Intervention policy training via supervised initialization and ISPO. The backbone remains frozen throughout.

however, records only the outcome of the original query sequence. We therefore construct counterfactual rollouts to evaluate candidate rewrites through the final answers produced with their retrieved evidence.

Given a question q, the frozen backbone B generates a multi-step reasoning–retrieval trajectory:

$$\tau = \big((s_1, q_1, c_1), \ldots, (s_T, q_T, c_T), y\big), \tag{1}$$

where T is the number of retrieval steps, s_t is the available reasoning–retrieval context before the t-th retrieval, q_t is the generated search query, c_t is the corresponding retrieved evidence, and y is the final answer. The context s_t includes the preceding reasoning, earlier queries, and previously retrieved evidence. Both modules take the input $x_t = (q, s_t, q_t)$.

For each intermediate query q_t, we use the DeepSeek-V4-flash (Xu et al., 2026) API to generate K candidate rewrites conditioned on x_t. We denote the candidate set by $\mathcal{Q}_t = \{\tilde{q}_t^{(k)}\}_{k=1}^{K}$. Each candidate retrieves alternative evidence:

$$\tilde{c}_t^{(k)} = \mathrm{Retrieve}(\tilde{q}_t^{(k)}). \tag{2}$$

To construct the corresponding counterfactual trajectory $\tilde{\tau}_t^{(k)}$, we retain the original trajectory prefix up to and including q_t, replace c_t with $\tilde{c}_t^{(k)}$, and let the frozen backbone continue reasoning and retrieval until it produces a final answer. Evidence retrieved using the rewritten query is provided to the backbone to guide subsequent reasoning and answer generation.

We evaluate final-answer correctness using the evaluation metric adopted by the corresponding backbone. Let a denote the gold answer, and define

$$R(\tau) = \mathrm{Eval}(y, a) \in \{0, 1\}, \tag{3}$$

where Eval is EM for Search-R1 and CEM for HiPRAG. A candidate rewrite is beneficial if $R(\tilde{\tau}_t^{(k)}) > R(\tau)$. Among the beneficial candidates, we select one rewrite as the supervision target $\tilde{q}_t^{\star}$. Retrieval steps for which no candidate improves the final answer are discarded. The resulting pairs $(x_t, \tilde{q}_t^{\star})$ are then used to fine-tune the rewriter f_ϕ.

3.2 Learning When to Intervene

Before each retrieval step, the intervention policy π_θ selects an action $a_t \in \{0, 1\}$ based on x_t. Action $a_t = 1$ invokes the rewriter, whereas $a_t = 0$ keeps the original query unchanged. Both

actions are followed by retrieval. The policy learns when applying the trained rewriter is beneficial for the final task performance. We keep the rewriter fixed throughout policy training, which consists of counterfactual supervised initialization followed by ISPO.

Counterfactual Policy Initialization. Rewriting a query can change the retrieved evidence and, in turn, affect subsequent reasoning and intervention decisions. We therefore construct supervision by comparing the final answers obtained under alternative intervention sequences.

Specifically, we generate counterfactual rollouts covering all four action sequences at the first two retrieval decisions:

$$\mathcal{A}_2 = \{(0,0),(0,1),(1,0),(1,1)\}. \tag{4}$$

Prescribed actions are executed when their decision points are reached, while subsequent retrieval queries are left unchanged. We evaluate the resulting answers using an answer-correctness metric, with F1 used as a tie-breaker. A sequence is preferred over another if it achieves a higher correctness score, or a higher F1 score when their correctness scores are equal.

When one sequence is preferred over all alternatives, we retain its action labels at the corresponding decision points. Otherwise, we group the sequences by their first action and select the best sequence within each group by ranking them according to the correctness score, then F1, and finally fewer rewrites. We then compare the best sequences from $\{(0,0),(0,1)\}$ and $\{(1,0),(1,1)\}$ and retain only the preferred first-action label. When only one retrieval decision is reached, we directly compare keeping and rewriting the query using the same criterion. Comparisons without a clear preference produce no labels.

Each retained action label is paired with the input x_t recorded at the corresponding decision point in its rollout. In particular, a second-step label is paired with the context produced after that rollout's first intervention decision. We use these input–action pairs for supervised fine-tuning of the intervention policy.

Intervention-Structured Policy Optimization. Further policy training must address two challenges in selective query intervention. First, an early preference for one action can leave alternative intervention sequences insufficiently explored. Second, earlier interventions may change the reasoning context of later decisions, which should be reflected in how step-level feedback is constructed. To address these challenges, we design ISPO around a shared intervention-prefix structure for exploration and advantage estimation.

Building on GiGPO's episode and step-level advantage framework (Feng et al., 2025), we develop two complementary designs: *forced-prefix exploration* with *intervention-prefix advantage estimation*. The former collects outcomes from alternative early intervention sequences, while the latter groups decisions by their preceding intervention actions to construct step-level feedback. Together, these designs use the intervention sequence structure to organize both rollout collection and feedback for policy updates.

Forced-prefix exploration. At each iteration, we take a snapshot π_{old} of the current policy and keep it fixed throughout rollout collection and the subsequent update. For each question, we combine forced rollouts covering all four prefixes in $\mathcal{A}_2$ with policy-sampled rollouts. Prescribed actions are executed only when their decision points are reached; all other actions are sampled from π_{old}. This provides feedback on prescribed alternatives alongside the policy's current behavior.

Intervention-prefix advantage estimation. For the G rollouts generated for each question, we compute final-answer rewards and episode-level advantages:

$$R_i = R(\tau_i) = \text{Eval}(y_i, y^\star), \qquad A_i^{\text{episode}} = R_i - \frac{1}{G}\sum_{j=1}^{G} R_j, \tag{5}$$

where y_i and $y^\star$ are the predicted and reference answers, respectively. Eval is EM for Search-R1 and CEM for HiPRAG.

For step-level feedback, we group decisions within each question by retrieval-step index and preceding intervention actions, $(t, a_{i,<t})$, where $a_{i,<t} = (a_{i,1}, \ldots, a_{i,t-1})$. Let $\mathcal{H}_{i,t}$ denote the group containing decision (i,t). Grouped decisions share prior intervention choices but may have different

queries and reasoning contexts. For a rollout with L_i intervention decisions, we compute

$$U_{i,t} = \gamma^{L_i - t + 1} R_i, \qquad A^{\text{step}}_{i,t} = U_{i,t} - \frac{1}{|\mathcal{H}_{i,t}|} \sum_{(j,u) \in \mathcal{H}_{i,t}} U_{j,u}, \tag{6}$$

where γ is the discount factor. We set $A^{\text{step}}_{i,t} = 0$ when its group does not contain both actions or has constant returns. The combined advantage is

$$\hat{A}_{i,t} = A^{\text{episode}}_i + \omega A^{\text{step}}_{i,t}, \tag{7}$$

where ω weights step-level feedback.

Policy update. We optimize both forced and policy-sampled decisions using a weighted clipped surrogate. For each executed action, we define

$$r_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t} \mid x_{i,t})}{\pi_{\text{old}}(a_{i,t} \mid x_{i,t})}, \tag{8}$$

where π_θ is the policy being optimized and $x_{i,t}$ is its input at decision (i,t). For forced decisions, the denominator still evaluates the prescribed action under π_{old}. The objective is

$$\mathcal{L}_{\text{RL}}(\theta) = -\mathbb{E}_{(i,t)\sim\mathcal{D}} \left[\min \left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\big(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_{i,t} \right) \right], \tag{9}$$

where ϵ controls the clipping range, and $\mathcal{D}$ is the empirical distribution assigning relative weight λ_{f} to prescribed actions within forced prefixes and unit weight to all policy-sampled actions.

3.3 Inference with Selective Query Intervention

At inference time, SEQUIN applies the learned intervention policy before each retrieval call. After the backbone generates an intermediate query q_t, the policy encodes $x_t = (q, s_t, q_t)$. A linear classification head maps the policy model's final-layer hidden representation at the decision position to two action logits. Softmax converts these logits into probabilities of keeping and rewriting the query. The action is determined by

$$p_t = \pi_\theta(a_t = 1 \mid x_t), \qquad a_t = \mathbb{I}[p_t \geq \eta], \tag{10}$$

where η is the intervention threshold. Further architectural details are provided in Appendix C.

The selected action determines the query sent to the retriever:

$$q^{\text{ret}}_t = \begin{cases} q_t, & a_t = 0, \\ f_\phi(q, s_t, q_t), & a_t = 1, \end{cases} \qquad c_t = \text{Retrieve}(q^{\text{ret}}_t). \tag{11}$$

Both actions execute retrieval, while only $a_t = 1$ requires a call to the rewriter.

The retrieved evidence c_t is returned through the backbone's existing retrieval interface and added to its context, which retains the original query q_t and preceding reasoning. The backbone then continues generation, and the policy evaluates each subsequent query using the updated reasoning–retrieval history. This process repeats until the final answer is produced, with the backbone parameters frozen throughout.

4 Experiments

4.1 Experimental Settings

Datasets and Evaluation Metrics. We evaluate SEQUIN using examples from the official development splits of HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022), together with Bamboogle (Press et al., 2023). We report Exact Match (EM) and Containment Exact Match (CEM) for each benchmark, along with their macro-averages across the four benchmarks. For HiPRAG-based methods, we use CEM as the primary evaluation metric because their answers include explanatory text.

Baselines. We compare SEQUIN with baselines covering generation without retrieval, conventional RAG, and Agentic RAG. **Direct Generation** answers questions using only the language

Table 1: Main results on four multi-hop question-answering benchmarks, measured by EM and CEM. Average denotes the unweighted mean across the four benchmarks. Best results in each column are shown in bold.

	HotpotQA		2WikiMulti		Bamboogle		MuSiQue		Average	
Method	EM	CEM	EM	CEM	EM	CEM	EM	CEM	EM	CEM
Direct Gen	0.144	0.161	0.232	0.242	0.032	0.040	0.013	0.021	0.105	0.116
CoT	0.148	0.180	0.198	0.227	0.208	0.248	0.026	0.043	0.145	0.175
RAG	0.218	0.273	0.191	0.238	0.064	0.072	0.024	0.031	0.124	0.154
IRCoT	0.245	0.325	0.180	0.238	0.144	0.152	0.032	0.054	0.150	0.192
Search-R1	0.306	0.336	0.284	0.316	0.208	0.224	0.063	0.078	0.215	0.239
+ SEQUIN	**0.327**	0.353	**0.292**	0.316	**0.256**	0.264	**0.086**	0.092	**0.240**	0.256
HiPRAG†	0.005	0.508	0.000	0.562	0.000	0.264	0.000	0.140	0.001	0.369
+ SEQUIN†	0.005	**0.512**	0.000	**0.583**	0.000	**0.312**	0.000	**0.141**	0.001	**0.387**

† HiPRAG-based methods generate answers with explanatory text, resulting in near-zero EM despite containing the correct answer in many cases. We use CEM as the primary evaluation metric for these methods.

model's parametric knowledge without external retrieval. **Chain-of-Thought (CoT)** (Wei et al., 2022) prompts the model to reason step by step before producing the final answer. **RAG** (Lewis et al., 2020) retrieves relevant documents once and conditions answer generation on the retrieved evidence. **IRCoT** (Trivedi et al., 2023) interleaves chain-of-thought reasoning with iterative retrieval, acquiring additional evidence as reasoning progresses. **Search-R1** and **HiPRAG** use reinforcement learning to optimize multi-step reasoning and retrieval. In addition to serving as baselines, these two agents provide the frozen backbones for evaluating SEQUIN.

Implementation Details. We use Search-R1-3B and HiPRAG as backbones in the main experiments and conduct additional evaluation on Search-R1-7B. The backbone parameters remain frozen throughout SEQUIN training. We initialize the Query Rewriter and Intervention Policy from Qwen2.5-3B-Instruct and Qwen2.5-0.5B-Instruct (Yang et al., 2024), respectively. Training data for both modules are constructed from the training splits of HotpotQA and 2WikiMultiHopQA. For retrieval, we use BM25 (Robertson & Zaragoza, 2009) over a 2018 Wikipedia dump (Karpukhin et al., 2020). We retrieve the top three documents per query and allow at most four retrieval steps per question. The training procedure is summarized in Algorithm 1 in the appendix. Additional implementation details are provided in Appendix C.

4.2 Main Results

Table 1 shows that Search-R1-3B with SEQUIN outperforms Direct Generation, CoT, RAG, and IRCoT in both EM and CEM across all four benchmarks. More importantly, SEQUIN further improves both frozen Agentic RAG backbones: macro-average EM on Search-R1-3B increases from 0.215 to 0.240, while macro-average CEM on HiPRAG increases from 0.369 to 0.387. These results show that query intervention can complement the agents' existing reasoning–retrieval capabilities without further backbone updates. Improvements are also observed on Bamboogle and MuSiQue, neither of which is used to train SEQUIN's rewriter or intervention policy. This indicates that the benefits extend beyond the benchmarks used for module training. Additional results on Search-R1-7B are reported in Appendix D, showing that the improvements also extend to a larger backbone.

5 Further Analysis

We analyze the contributions of Rewriter training, Intervention Policy training, and selective intervention on Search-R1-3B. We then examine sensitivity to the forced-action loss weight and the intervention threshold.

5.1 Effect of Query Rewriter Training

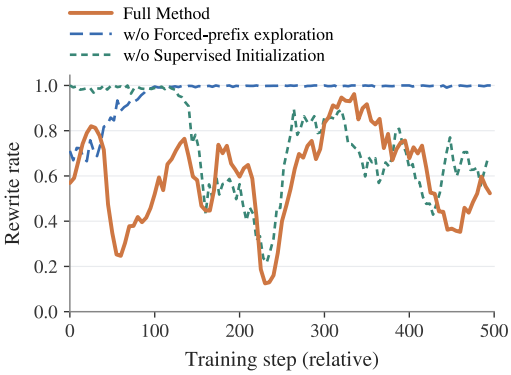

Figure 3: Effect of supervised initialization and forced-prefix exploration on rewriting rates during reinforcement learning. Rates are measured over policy-sampled decisions.

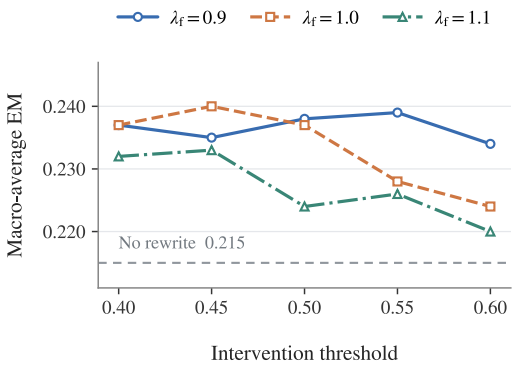

Figure 4: Macro-average EM across intervention thresholds η for different forced-action loss weights λ_f. The horizontal dashed line denotes the no-rewrite baseline.

To assess the benefit of counterfactual rewriter training, we compare always-rewrite variants using untuned and trained 3B rewriters, with the Search-R1 agent as the No Rewrite baseline. The two rewriters share the same initial checkpoint, prompting scheme, and input format. As shown in Table 2, the untuned rewriting yields a macro-average EM of 0.201, below the original agent's 0.215. Training raises this score to 0.232, an absolute gain of approximately 0.031. These results indicate that task-specific training is important for obtaining beneficial query rewrites.

Method	HotpotQA	2Wiki	Bamboogle	MuSiQue	Avg.
No Rewrite	0.306	0.284	0.208	0.063	0.215
Always Rewrite (Untuned)	0.305	0.248	0.184	0.067	0.201
Always Rewrite (Trained)	0.321	0.287	0.240	0.078	0.232
Selective (SFT Only)	0.315	0.289	0.224	0.065	0.223
Selective (ISPO Only)	0.315	**0.296**	**0.256**	0.077	0.236
SEQUIN (SFT + ISPO)	**0.327**	0.292	**0.256**	**0.086**	**0.240**

Table 2: Ablations of rewriter and policy training on Search-R1-3B (EM). Selective variants share the same trained rewriter.

5.2 Effect of Intervention Policy Training

We examine optimization with ISPO, supervised initialization and forced-prefix exploration, keeping the backbone and trained rewriter fixed throughout these comparisons.

Optimization with ISPO. To assess the additional benefit of ISPO, we compare an SFT-only intervention policy with one further optimized through ISPO, keeping the backbone and trained Query Rewriter fixed. Table 2 shows that ISPO increases macro-average EM from 0.223 to 0.240, a gain of 1.7 percentage points. The SFT-only policy falls below Always Rewrite with the trained rewriter (0.232), whereas the policy further trained with ISPO surpasses it. These results support the contribution of reinforcement learning beyond supervised initialization.

Supervised Initialization. We remove supervised initialization and train the intervention policy directly through ISPO, while retaining forced-prefix exploration and the same frozen backbone and trained Query Rewriter. As shown in Table 2, Selective (ISPO Only) achieves a macro-average EM of 0.236, compared with 0.240 for the full procedure. Figure 3 shows that ISPO learns selective intervention even without supervised initialization. Supervised initialization further improves final-answer accuracy.

Forced-Prefix Exploration. We remove forced-prefix exploration and sample all intervention actions from the policy snapshot, while retaining supervised initialization and keeping the backbone and trained Query Rewriter fixed. Figure 3 shows that the policy-sampled rewriting rate stabilizes near 100%, whereas the full method retains selective intervention. This suggests that forced-prefix exploration helps avoid an always-rewrite policy in this setting.

5.3 Effect of Selective Intervention

Selective versus Always Rewriting. We compare SEQUIN with Always Rewrite using the same trained Query Rewriter on Search-R1-3B. Table 3 shows that SEQUIN reduces rewrite calls by

Table 3: Query rewriting strategies on Search-R1-3B. All strategies that rewrite use the same trained Query Rewriter. Rewrite calls are normalized to Always Rewrite separately for each benchmark. The highest EM per benchmark is bold.

Method	HotpotQA		2WikiMultiHopQA		Bamboogle		MuSiQue	
	EM	Calls (%)	EM	Calls (%)	EM	Calls (%)	EM	Calls (%)
No Rewrite	0.306	0.0	0.284	0.0	0.208	0.0	0.063	0.0
Always Rewrite	0.321	100.0	0.287	100.0	0.240	100.0	0.078	100.0
Random Intervention	0.310	77.3	0.285	70.2	0.216	72.7	0.078	73.0
SEQUIN	**0.327**	76.8	**0.292**	69.4	**0.256**	72.8	**0.086**	67.5

23.2%–32.5% across benchmarks, with an unweighted mean reduction of 28.4%, while increasing macro-average EM from 0.232 to 0.240. EM improves across all four benchmarks. This reduction in rewrite usage is obtained with a final-answer correctness reward and no explicit penalty on rewrite calls, showing that selective intervention can reduce rewriting while improving answer accuracy in this setting.

Learned versus Random Intervention. To assess the value of learned intervention decisions, we compare SEQUIN with Random Intervention using the same trained Query Rewriter. The random baseline independently invokes the rewriter at each retrieval step with a fixed probability. As shown in Table 3, SEQUIN outperforms Random Intervention in EM on every benchmark, achieving a macro-average EM of 0.240 versus 0.222. Their mean relative rewrite counts across the four benchmarks are 71.6% and 73.3%, respectively. With a slightly lower average rewrite budget, SEQUIN achieves a 1.8-percentage-point EM gain, supporting the value of learning which queries to rewrite.

5.4 Hyperparameter Sensitivity

Forced-Action Loss Weight. The weight λ_{f} controls the relative contribution of prescribed actions to the policy loss. We train intervention policies with $\lambda_{\mathrm{f}} \in \{0.9, 1.0, 1.1\}$ and compare their macro-average EM at the same inference thresholds. As shown in Figure 4, $\lambda_{\mathrm{f}} = 1.0$ achieves the highest EM at $\eta = 0.45$, while $\lambda_{\mathrm{f}} = 0.9$ performs best at $\eta \in \{0.50, 0.55, 0.60\}$. Across all tested thresholds, $\lambda_{\mathrm{f}} = 1.1$ yields lower EM than the other two weights. Thus, no single training weight performs best at every tested threshold, and increasing the forced-action loss weight does not necessarily improve accuracy.

Inference Threshold. Keeping each trained policy fixed, we vary the intervention threshold over $\eta \in \{0.40, 0.45, 0.50, 0.55, 0.60\}$. Figure 4 shows macro-average EM as a function of η, with one curve for each training weight. For $\lambda_{\mathrm{f}} = 0.9$, EM ranges from approximately 0.234 to 0.239, exhibiting the smallest variation among the three weights. All tested configurations exceed the No Rewrite baseline of 0.215, indicating that the accuracy gains are maintained across the evaluated weights and thresholds.

6 Conclusion

We presented SEQUIN, a framework that learns how and when to rewrite intermediate queries in Agentic RAG while keeping the backbone frozen. SEQUIN combines a trajectory-aware query rewriter trained on beneficial counterfactual rewrites with an intervention policy trained through counterfactual supervised initialization and ISPO. ISPO combines forced-prefix exploration with episode-level and intervention-prefix advantage estimation. Experiments on four multi-hop question-answering benchmarks show improvements in macro-average EM on Search-R1 and CEM on HiPRAG. Further analyses on Search-R1 demonstrate the benefits of task-specific rewriter training and reinforcement learning for the intervention policy. Compared with always rewriting, SEQUIN reduces rewrite calls by approximately 30% on average across benchmarks while improving macro-average EM. It also outperforms random intervention at a similar average rewrite budget. These findings support learning query interventions from their downstream effects as a modular approach to improving frozen Agentic RAG systems.

AI USE STATEMENT

We used generative AI tools to assist with drafting and polishing the manuscript, including improving clarity, grammar, and readability, as well as with code implementation and debugging. All AI-assisted text and code were reviewed and revised by the authors. We take full responsibility for the final manuscript, code, and reported results, including the accuracy, originality, and validity of the work.

ETHICS STATEMENT

This work presents a selective query intervention framework for Agentic RAG. All experiments are conducted on publicly available multi-hop question-answering benchmarks, without deployment in real-world decision-making or safety-critical applications. We do not anticipate direct ethical risks from the reported experiments, while acknowledging potential biases and inaccuracies in the underlying language models and retrieved evidence.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe the datasets, evaluation metrics, backbone models, and retrieval settings in Section 4.1. Appendix C provides implementation details and training hyperparameters for the Query Rewriter and intervention policy. Algorithm 1 summarizes the training procedure, and Figures 6 and 7 provide the prompt templates. All experiments use publicly available benchmarks.

REFERENCES

Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. Sparkle: A structured and plug-and-play agentic retrieval policy for adaptive rag models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 38699–38719, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=QXEhBMNrCW.

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.

Pengcheng Jiang, Jiacheng Lin, Zhiyi Shi, Zifeng Wang, Luxi He, Yichen Wu, Ming Zhong, Peiyang Song, Qizheng Zhang, Heng Wang, et al. Adaptation of agentic ai: A survey of post-training, memory, and skills. *arXiv preprint arXiv:2512.16301*, 2025.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

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

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025a.

Yangning Li, Weizhi Zhang, Yuyao Yang, Wei-Chieh Huang, Yaozu Wu, Junyu Luo, Yuanchen Bei, Henry Peng Zou, Xiao Luo, Yusheng Zhao, Chunkit Chan, Yankai Chen, Zhongfen Deng, Yinghui Li, Hai-Tao Zheng, Dongyuan Li, Renhe Jiang, Ming Zhang, Yangqiu Song, and Philip S. Yu. A survey of RAG-reasoning systems in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 12120–12145, 2025b. URL https://aclanthology.org/2025.findings-emnlp.648/.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Yuchen Li, Daiting Shi, Dawei Yin, and Zhicheng Dou. Agentic-r: Learning to retrieve for agentic search. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 15987–16005, 2026.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 5303–5315, 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 *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Sara Shanian, Xiaoqin Yi, Pavlo Ruban, and Kurt MacDonald. Better together: Complementary query rewriting under a strong rag baseline. *arXiv preprint arXiv:2609.05637*, 2026.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12991–13013, 2024.

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

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

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, et al. Freshllms: Refreshing large language models with search engine augmentation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13697–13720, 2024.

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

Tongyu Wen, Guanting Dong, and Zhicheng Dou. Smartsearch: Process reward-guided query refinement for search agents. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1971–1982, 2026.

Haiyan Wu, Chenchen Wang, Chaoqun Sun, Chengxiong Lu, Yan-Hong Chen, Zhiqiang Zhang, and Xiaoqing Feng. Reflective rag: Self-evaluation driven strategy optimization in agentic retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13252–13267, 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 *International Conference on Learning Representations*, volume 2026, pp. 82317–82347, 2026b.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 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 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Xiaopeng Ye, Chen Xu, Chaoliang Zhang, Zhaocheng Du, Jun Xu, Gang Wang, and Zhenhua Dong. Q-prm: Adaptive query rewriting for retrieval-augmented generation via step-level process supervision. In *EMNLP (Findings)*, pp. 15113–15128, 2025.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, et al. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 58701–58729, 2026.

Shu Zhao, Tan Yu, and Anbang Xu. Beyond the limitation of a single query: Train your llm for query expansion with reinforcement learning. *arXiv preprint arXiv:2510.10009*, 2025.

Chenghao Zhu, Nuo Chen, Yufei Gao, Yunyi Zhang, Prayag Tiwari, and Benyou Wang. Is your llm outdated? a deep look at temporal generalization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7433–7457, 2025.

APPENDIX

A LIMITATIONS AND FUTURE WORK

Limitations. SEQUIN operates only on queries issued by a frozen backbone, which limits its ability to address reasoning errors and evidence missing from the retrieval corpus. Its outcome-based supervision provides imperfect credit assignment for individual interventions, while repeated counterfactual rollouts increase training cost. Moreover, fewer rewrite calls do not necessarily translate into lower end-to-end latency because policy inference introduces additional computation.

Future Work. Future work could extend selective intervention to evidence verification and reasoning revision, develop more reliable credit assignment with evidence-grounded feedback, and reduce training cost through adaptive rollout allocation. Evaluating transfer across backbone and retrieval configurations, together with end-to-end efficiency, is another useful direction.

B TRAINING PROCEDURE OF SEQUIN

Algorithm 1 summarizes the two-stage training procedure of SEQUIN, with the backbone frozen throughout. First, we train the Query Rewriter using counterfactual rollouts that replace retrieved

Algorithm 1 Training Procedure of SEQUIN

Input: Datasets $\mathcal{D}_R$, $\mathcal{D}_{\mathrm{SFT}}$, $\mathcal{D}_{\mathrm{RL}}$; frozen backbone B; candidate count K; rollout group size G; discount factor γ; advantage weight ω; forced-action loss weight λ_{f}; clipping parameter ϵ
Output: Query Rewriter f_ϕ and Intervention Policy π_θ

Part I: Learning to rewrite

$\mathcal{S}_R \leftarrow \emptyset$
for all $q \in \mathcal{D}_R$ **do**
  $\tau \leftarrow \mathrm{OriginalRollout}(B, q)$
  for each retrieval step t in τ **do**
    $x_t \leftarrow (q, s_t, q_t)$; $\mathcal{Q}_t \leftarrow \mathrm{Candidates}(x_t, K)$
    for all $q' \in \mathcal{Q}_t$ **do**
      $\widetilde{\tau}_{q'} \leftarrow \mathrm{Replay}(B, \tau, t, \mathrm{Retrieve}(q'))$
    $\mathcal{Q}_t^+ \leftarrow \{q' \in \mathcal{Q}_t : R(\widetilde{\tau}_{q'}) > R(\tau)\}$
    if $\mathcal{Q}_t^+ \neq \emptyset$ **then**
      $\tilde{q}_t^\star \leftarrow \mathrm{SelectOne}(\mathcal{Q}_t^+)$
      $\mathcal{S}_R \leftarrow \mathcal{S}_R \cup \{(x_t, \tilde{q}_t^\star)\}$
$f_\phi \leftarrow \mathrm{SFT}(\mathcal{S}_R)$; freeze f_ϕ

Part II: Learning when to intervene

$\mathcal{A}_2 \leftarrow \{0,1\}^2$; $\mathcal{S}_\pi \leftarrow \emptyset$
Counterfactual policy initialization
Define $\mu_0(x) = 0$ for all inputs x
for all $q \in \mathcal{D}_{\mathrm{SFT}}$ **do**
  $\mathcal{G}_q \leftarrow \mathrm{ForcedRollouts}(q, B, f_\phi, \mathcal{A}_2, \mu_0)$
  $\mathcal{S}_\pi \leftarrow \mathcal{S}_\pi \cup \mathrm{PreferencePairs}(\mathcal{G}_q)$
$\pi_\theta \leftarrow \mathrm{SFT}(\mathcal{S}_\pi)$
Intervention-Structured Policy Optimization
for each RL iteration **do**
  $\pi_{\mathrm{old}} \leftarrow \mathrm{Snapshot}(\pi_\theta)$; $\mathcal{C} \leftarrow \emptyset$
  $\mathcal{B} \leftarrow \mathrm{SampleBatch}(\mathcal{D}_{\mathrm{RL}})$
  for all $q \in \mathcal{B}$ **do**
    $\mathcal{G}_q^{\mathrm{forced}} \leftarrow \mathrm{ForcedRollouts}(q, B, f_\phi, \mathcal{A}_2, \pi_{\mathrm{old}})$
    $\mathcal{G}_q^{\mathrm{policy}} \leftarrow \mathrm{PolicyRollouts}(q, B, f_\phi, \pi_{\mathrm{old}}, G-4)$
    $\mathcal{G}_q \leftarrow \mathrm{Concat}(\mathcal{G}_q^{\mathrm{forced}}, \mathcal{G}_q^{\mathrm{policy}})$
    Compute $R_i = \mathrm{Eval}(y_i, y^\star)$ for all $\tau_i \in \mathcal{G}_q$
    Compute A_i^{episode} using equation 5
    Group decisions within $\mathcal{G}_q$ by $(t, a_{i,<t})$
    Compute $A_{i,t}^{\mathrm{step}}$ using equation 6; set it to zero for groups lacking either action or return variation
    for each decision (i,t) in $\mathcal{G}_q$ **do**
      $\hat{A}_{i,t} \leftarrow A_i^{\mathrm{episode}} + \omega A_{i,t}^{\mathrm{step}}$
      $\ell_{i,t}^{\mathrm{old}} \leftarrow \log \pi_{\mathrm{old}}(a_{i,t} \mid x_{i,t})$
      $w_{i,t} \leftarrow \lambda_{\mathrm{f}}$ if $a_{i,t}$ was forced, otherwise 1
      Append $(x_{i,t}, a_{i,t}, \ell_{i,t}^{\mathrm{old}}, \hat{A}_{i,t}, w_{i,t})$ to $\mathcal{C}$
  Update θ on $\mathcal{C}$ using the weighted objective in equation 9
return f_ϕ, π_θ

evidence while preserving the original query and preceding history. The highest-reward candidate is retained for supervised training only if it improves the final-answer reward over the original rollout. Second, with the trained rewriter fixed, we initialize the intervention policy using supervision derived from comparisons among the four action prefixes in $\{0,1\}^2$. We then optimize the policy through ISPO, combining forced-prefix and fully policy-sampled rollouts. Policy updates use episode-level advantages and step-level advantages grouped by retrieval-step index and preceding intervention actions, under the weighted objective defined in the main text.

C Additional Implementation Details

All training experiments are conducted with CUDA 12.4 using bfloat16 (BF16) precision. For retrieval, we follow the setup of Search-R1 (Jin et al., 2025).

C.1 Query Rewriter

We initialize the Query Rewriter from Qwen2.5-3B-Instruct. To construct its training data, we sample equal numbers of questions from the training splits of HotpotQA and 2WikiMultihopQA. We use DeepSeek-V4-Flash to generate candidate rewrites for intermediate search queries and evaluate them through counterfactual rollouts. Filtering candidates according to improvements in final-answer correctness yields approximately 20,000 supervised training examples.

We fine-tune the rewriter using LoRA with a learning rate of 3×10^{-5}, a per-device batch size of 4, and 16 gradient accumulation steps, giving an effective batch size of 64 on a single GPU. The maximum sequence length is 2,816 tokens. Rewriter training uses a single RTX 4090 GPU with 48 GB of memory.

C.2 Intervention Policy

Policy Architecture. We initialize the intervention policy from Qwen2.5-0.5B-Instruct and add a trainable linear classification head. At each retrieval decision, the policy processes the trajectory prefix through the current search query and extracts the final-layer hidden state at the corresponding decision boundary. The linear head maps this representation to two logits, and a softmax produces the probabilities of keeping the original query or invoking the rewriter. In the SFT implementation, the decision position is supplied with each example or located at the end of the latest `</search>` tag. Inputs are limited to 2,816 tokens using left truncation to preserve the most recent context.

Supervised Initialization. We train the intervention policy on approximately 7,000 labeled decision examples obtained from counterfactual rollouts on questions from the training splits of HotpotQA and 2WikiMultihopQA. The SFT objective is unweighted cross-entropy over the two actions. We update the LoRA parameters and the classification head while keeping the pretrained base parameters frozen. LoRA is applied to the query and value projections with rank 16, scaling factor $\alpha = 32$, and dropout 0.1. We use AdamW with a learning rate of 3×10^{-4} and weight decay 0.01, a batch size of 16, and 8 gradient accumulation steps. We train for 10 epochs using a linear learning-rate schedule with a warmup ratio of 0.05.

ISPO. We further optimize the intervention policy using ISPO (built on GiGPO (Feng et al., 2025)) on 12,000 questions sampled from the training splits of HotpotQA and 2WikiMultihopQA. For Search-R1-3B, each question has a rollout group of size 8. Four trajectories cover the forced action prefixes $\{(0,0),(0,1),(1,0),(1,1)\}$ at the first two retrieval decisions, with prescribed actions executed whenever their decision points are reached. The remaining four trajectories are fully policy-sampled. We train for one epoch with a learning rate of 3×10^{-5} and 8 gradient accumulation steps on two NVIDIA A800 GPUs. The reasoning backbone and the trained Query Rewriter remain frozen during policy optimization.

D Performance Across Backbone Sizes

To examine the effectiveness of SEQUIN across backbone sizes, we extend the Search-R1-3B evaluation presented in the main text to Search-R1-7B. We compare the original Search-R1-7B agent with its SEQUIN-augmented counterpart on four multi-hop question answering benchmarks, keeping the backbone frozen throughout SEQUIN training. Table 4 reports the per-benchmark results and their average. Together with the 3B results in the main text, this comparison assesses whether the benefits of selective query intervention extend to a larger backbone.

E Case Study

Figure 5 illustrates how SEQUIN uses retrieved evidence to refine an intermediate query. Search-R1 identifies Roger Young as the director but fails to retrieve his birthplace, returning "Unknown" after four retrievals. In SEQUIN, the earlier evidence explicitly links Roger Young to the role of director. The Query Rewriter incorporates this role into "Roger Young director birthplace", which retrieves the relevant biography and enables the correct answer, "Champaign, Illinois", after two retrievals.

Table 4: Results in the 7B backbone setting on four multi-hop question-answering benchmarks, measured by EM and F1. Average denotes the unweighted mean across the four benchmarks. Best results in each column are shown in bold.

	HotpotQA		2WikiMulti		Bamboogle		MuSiQue		Average	
Method	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Direct Gen	0.165	0.2459	0.210	0.2408	0.088	0.1695	0.023	0.0968	0.122	0.1883
CoT	0.145	0.2442	0.178	0.2330	0.280	0.3830	0.036	0.1005	0.160	0.2402
RAG	0.259	0.3641	0.214	0.2557	0.104	0.2096	0.023	0.0671	0.150	0.2241
IRCoT	0.301	0.4202	0.200	0.2699	0.224	0.3053	0.053	0.1440	0.195	0.2849
Search-R1	0.391	0.5060	0.337	0.3946	0.328	0.4285	0.124	0.2147	0.295	0.3860
+ SEQUIN	**0.403**	**0.5147**	**0.346**	**0.4034**	**0.384**	**0.4775**	**0.130**	**0.2185**	**0.316**	**0.4035**

F PROMPTS

Figures 6 and 7 present the prompt templates for the Query Rewriter and Intervention Policy, respectively. Both use the original question, the reasoning–retrieval history, and the current query to support query refinement and intervention decisions.



Figure 5: A case study comparing Search-R1 and SEQUIN. SEQUIN incorporates the occupation qualifier "director", supported by earlier retrieved evidence, into the birthplace query. All retrieval calls are shown; reasoning and retrieved documents are abbreviated as indicated.

Prompt for Query Rewriter

You are a search query rewriter for multi-step retrieval reasoning. Your task is to rewrite the current search query to improve retrieval quality. Do not answer the question directly. Do not generate explanation. Output only one rewritten search query.

Question:
{question}

Context before this step:
{trace}

Original query:
{original_query}

Figure 6: Prompt template for the Query Rewriter.

Prompt for Intervention Policy

Judge whether the latest search query in the following trajectory prefix should be rewritten before retrieval.
Rewrite only if the query is vague, incomplete, ambiguous, redundant, poorly constrained, or unlikely to retrieve the needed evidence.

Question:
{question}

Trajectory prefix:
{full_trace}

Figure 7: Prompt template for the Intervention Policy.