CoSearch: Joint Training of Reasoning and Document Ranking via Reinforcement Learning for Agentic Search

Hansi Zeng[1], **Liam Collins**[2], **Bhuvesh Kumar**[2], **Neil Shah**[2], **Hamed Zamani**[1]
[1] Center for Intelligent Information Retrieval, University of Massachusetts Amherst
[2] Snap Inc.

Abstract

Agentic search – the task of training agents that iteratively reason, issue queries, and synthesize retrieved information to answer complex questions – has achieved remarkable progress through reinforcement learning (RL). However, existing approaches such as Search-R1 treat the retrieval system as a fixed tool, optimizing only the reasoning agent while the retrieval component remains unchanged. A preliminary experiment reveals that the gap between an oracle and a fixed retrieval system reaches up to **+26.8%** relative F1 improvement across seven QA benchmarks, suggesting that the retrieval system is a key bottleneck in scaling agentic search performance. Motivated by this finding, we propose CoSearch, a framework that jointly trains a multi-step reasoning agent and a generative document ranking model via Group Relative Policy Optimization (GRPO). To enable effective GRPO training for the ranker—whose inputs vary across reasoning trajectories—we introduce a semantic grouping strategy that clusters sub-queries by token-level similarity, forming valid optimization groups without additional rollouts. We further design a composite reward combining ranking quality signals with trajectory-level outcome feedback, providing the ranker with both immediate and long-term learning signals. Experiments on seven single-hop and multi-hop QA benchmarks demonstrate consistent improvements over strong baselines, with ablation studies validating each design choice. Our results show that joint training of the reasoning agent and retrieval system is both feasible and strongly performant, pointing to a key ingredient for future search agents. Code is available at https://github.com/snap-research/CoSearch

1 Introduction

Large language models (LLMs) have demonstrated strong reasoning capabilities, yet they face fundamental challenges when questions require knowledge beyond their parametric training data. Retrieval-augmented generation (RAG) (Lewis et al., 2020; Zamani et al., 2022) addresses this limitation by allowing models to access external knowledge through retrieval. While early approaches relied on single-turn retrieval followed by answer generation, recent work has shown that multi-step agentic search—where models iteratively reason, generate search queries, and incorporate retrieved information—yields substantially stronger performance on knowledge-intensive tasks (Trivedi et al., 2023; Khattab et al., 2021; Shi et al., 2024b). In these agentic search systems, an agent interacts with a retrieval system iteratively: the agent reasons about what information is needed, issues a search query, observes the retrieved documents, and repeats this process until sufficient evidence is gathered to produce a final answer. Training such agents through reinforcement learning (RL), where the reward is based on final answer correctness, has proven effective at teaching them *when* and *what* to search (Jin et al., 2025; Li et al., 2025b; Zeng et al., 2026). Despite this progress, a common assumption across these systems is that the retrieval model or search engine remains unchanged. This means that the agent treats the retrieval model as a tool and learns to use it and the tool does not adapt.

	7B Main Agent			3B Main Agent		
	Single	Multi	**Avg**	Single	Multi	**Avg**
Standard	0.600	0.506	0.546	0.531	0.371	0.440
Oracle Retrieval	0.709	0.564	0.626	0.636	0.499	0.558
Rel. Gain	**+18.2%**	**+11.5%**	**+14.7%**	**+19.8%**	**+34.5%**	**+26.8%**

Table 1: Oracle retrieval gap (F1 score). Performance averaged over single-hop (PopQA, NQ, TQA) and multi-hop (HotpotQA, 2Wiki, Musique, Bamboogle) benchmarks.

We argue that this assumption introduces a significant bottleneck. Through analysis of RL-trained search agents, we observe that most errors originate not from the reasoning process but from the retrieval component: the agent issues queries that look reasonable, yet the retrieval system fails to return relevant documents, forcing the agent to either retry with reformulated queries or draw incorrect conclusions from noisy information. To quantify this gap, we conduct an oracle retrieval experiment: at each retrieval step, we promote all documents matching the gold (ground-truth) answer to the top positions; for sub-queries where no retrieved document matches the gold answer, the ranking is left unchanged (see Appendix F for details). As shown in Table 1, this simple intervention consistently improves average F1 by **+14.7%** for a 7B-parameter agent and by **+26.8%** for a 3B agent across seven QA benchmarks, demonstrating that the retrieval bottleneck holds broadly regardless of the agent's reasoning capacity. These findings motivate the following question: *can we jointly optimize the retrieval system alongside the multi-turn reasoning agent, so that they learn to complement each other for providing accurate responses?*

In this work, we propose CoSearch, a reinforcement learning (RL) framework that jointly trains a main reasoning agent and a generative ranker for agentic search tasks. The main agent operates in a standard ReAct-style loop (Yao et al., 2023b), generating thoughts and search queries. The generative ranker receives the main agent's sub-queries along with candidate documents from a fixed first-stage dense retriever, and selects and reranks a subset for the main agent to observe; together, the retriever and ranker form the retrieval system. The main agent and the ranker are trained simultaneously using Group Relative Policy Optimization (GRPO) (DeepSeek-AI et al., 2025), with rewards derived from the correctness of the final answer.

Applying GRPO to the ranker introduces two key technical challenges. First, GRPO computes advantages over groups of responses generated from the *same input prompt*. While this is naturally satisfied for the main agent—where multiple trajectories are sampled from the same user query—the ranker receives different sub-queries across trajectories, preventing direct grouping. We address this through a *semantic grouping* strategy that clusters sub-queries by token-level F1 similarity, forming valid GRPO groups without additional rollouts. Second, the ranker's effect on the final answer is indirect: good retrieval may still yield an incorrect answer if reasoning fails, and vice versa. We design a *composite reward* combining a relevance reward based on a ranking quality metric (Hit@k) with a trajectory-level reward reflecting the final answer correctness, providing both immediate feedback on retrieval precision and long-term signal on downstream utility. Both reward components are computed using rule-based metrics, requiring no expensive LLM annotations.

Following prior work (Jin et al., 2025), we evaluate CoSearch on seven QA benchmarks spanning single-hop (PopQA, NQ, TriviaQA) and multi-hop (HotpotQA, 2WikiMultiHopQA, Musique, Bamboogle) settings. Using Qwen2.5-7B-Instruct as the backbone for both the main and ranking agents, our method consistently outperforms strong baselines including Search-R1 (Jin et al., 2025) and fixed ranker variants. We further demonstrate effectiveness with a 3B main agent paired with a 7B ranker. Our main contributions are as follows:

- We identify retrieval quality as a critical bottleneck in RL-trained agentic search systems: an oracle retrieval experiment reveals relative F1 gains of +14.7% for a

7B reasoning agent and +26.8% for a 3B agent, demonstrating a substantial and broadly held performance ceiling imposed by the fixed retrieval system.

- We propose CoSearch, a framework that jointly trains a reasoning agent and a generative document ranker via RL. To enable this, we introduce a semantic grouping strategy for applying GRPO to the ranker despite varying inputs across trajectories, and a composite reward that combines ranking quality with trajectory-level answer correctness.
- Experiments on seven QA benchmarks show consistent improvements over strong baselines, achieving +6.6% relative F1 over Search-R1 with a 7B agent and +10.8% with a 3B agent. Our results show that jointly training the retrieval system and reasoning agent is not only feasible but strongly performant, and is likely a key ingredient for future search agents.

2 Related Work

Agentic search and deep research. Retrieval-augmented generation (Lewis et al., 2020; Guu et al., 2020; Borgeaud et al., 2022; Karpukhin et al., 2020a) enables LLMs to access external knowledge, but single-turn retrieval often fails on complex questions requiring multi-step evidence gathering. This has motivated agentic search systems that interleave reasoning with retrieval (Yao et al., 2023a; Trivedi et al., 2023; Asai et al., 2024), with recent deep research agents (OpenAI, 2025; Google, 2025; Nakano et al., 2021) coupling large reasoning models with web search for complex queries. The advent of RLVR has further enabled training search agents directly from answer correctness (Jin et al., 2025; Li et al., 2025a; Song et al., 2025; Li et al., 2026; Sun et al., 2025b; Guan et al., 2025; Zeng et al., 2026).

Training retrievers for LLMs. A separate line of work adapts retrieval for downstream LLM generation. Prior approaches train retrievers via language model supervision (Guu et al., 2020; Shi et al., 2024a; Sachan et al., 2021; Salemi & Zamani, 2024; Zeng et al., 2022; 2023b; 2025; 2024; 2023a), while recent work shows that LLMs can serve as effective listwise rankers (Sun et al., 2023; Pradeep et al., 2023). Most related, Agentic-R (Liu et al., 2026) trains a dense retriever for agentic search via LLM-annotated labels and iterative contrastive learning. In contrast, CoSearch uses a generative ranker trained end-to-end with RL in a simultaneous joint framework, requiring only lightweight rule-based rewards.

RL for tool-augmented LLMs. RLVR (OpenAI, 2024; DeepSeek-AI et al., 2025; Lambert et al., 2025; Kaufmann et al., 2025) has become a key paradigm for optimizing LLMs, with value-free methods such as GRPO (Shao et al., 2024) and RLOO (Ahmadian et al., 2024) offering simpler alternatives to PPO (Schulman et al., 2017). Applying RLVR to tool-augmented settings has attracted growing interest (Li et al., 2025c; Feng et al., 2025a; Dong et al., 2025b; Wang et al., 2025; Feng et al., 2025b; Xue et al., 2025; Dong et al., 2025a), but these works focus on training the *reasoning agent* to use tools more effectively. In contrast, our work applies GRPO to train the *tool itself*, specifically the rankers.

3 Methodology

We present CoSearch, a framework that jointly optimizes a multi-step reasoning (main) agent and a generative document ranker through reinforcement learning (RL) (see Figure 1). We first formalize the framework (§3.1), describe the main agent optimization (§3.2), and then detail our two core technical contributions: a semantic grouping strategy for applying GRPO to the ranker (§3.3) and a composite reward design (§3.4).

3.1 Problem Formulation

We formalize agentic search as a system comprising a reasoning (main) agent and a retrieval system. Given an initial user query q_0, the *main agent* $\pi_{\theta_{\text{main}}}$ interacts with the retrieval system through a sequence of reasoning and retrieval steps to produce a trajectory:

$$y = \big(q_0,\ \tau_1, q_1, o_1,\ \tau_2, q_2, o_2,\ \ldots,\ \tau_m, q_m, o_m,\ \tau_{m+1}, a\big), \tag{1}$$

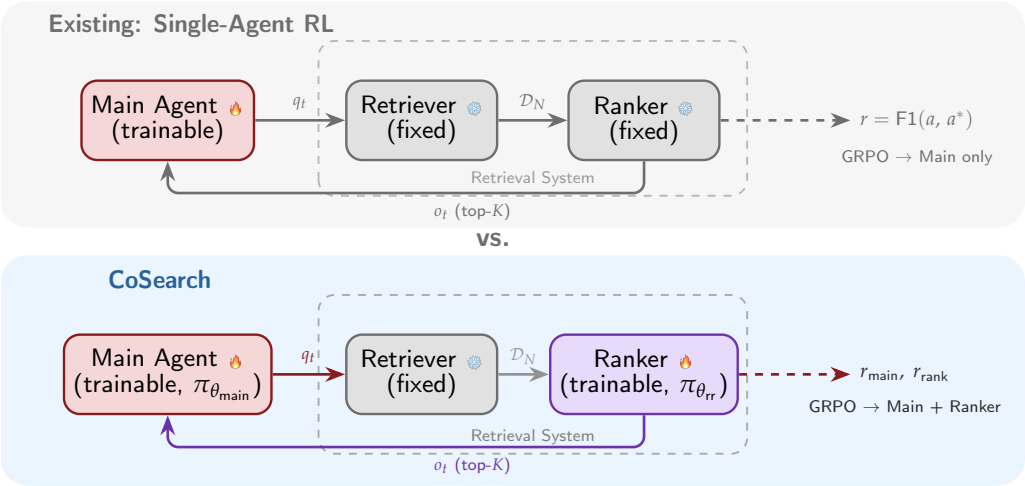

Figure 1: Both panels share the same architecture: Main Agent → Retriever → Ranker. Top: existing approaches treat the entire retrieval system as fixed (❄) and train only the main agent. Bottom: CoSearch decomposes the retrieval system into a fixed dense retriever and a trainable generative ranker (🔥), jointly optimizing both the main agent and the ranker via GRPO.

where τ_t is the reasoning thought at step t, q_t is the sub-query generated by the main agent, o_t is the observation (retrieved documents) returned by the retrieval system, and a is the final answer.

In existing agentic search frameworks (Jin et al., 2025; Gao et al., 2025; Li et al., 2025a; Zheng et al., 2025), the observation o_t is provided by a fixed retrieval. In our framework, we decompose the retrieval system into two stages and make the second stage trainable: a first-stage dense retriever produces a candidate set $\mathcal{D}_N$ of N documents, and a *generative ranker* $\pi_{\theta_{\text{gr}}}$ then selects and ranks a subset $\mathcal{D}_K \subset \mathcal{D}_N$ of K documents to form the observation o_t (see Appendix B for the full algorithm and concrete examples). The dense retriever remains fixed; the ranker is the trainable component. The joint optimization objective is:

$$\max_{\theta_{\text{main}}, \theta_{\text{gr}}} \quad \mathbb{E}_{T \sim \pi_{\theta_{\text{main}}}, \pi_{\theta_{\text{gr}}}} \big[r(T) \big], \tag{2}$$

where $r(T)$ measures the quality of the final answer. Unlike existing approaches that freeze the retrieval system and only optimize θ_{main}, our framework makes o_t a function of the trainable ranker, enabling the ranker to be optimized end-to-end by downstream answer correctness.

3.2 Main Agent Optimization

The main agent is optimized over complete reasoning trajectories using GRPO (DeepSeek-AI et al., 2025). For each query q_0, we sample G trajectories $\{y_i\}_{i=1}^{G}$ from the current policy. The GRPO objective is:

$$\mathcal{J}_{\text{GRPO}}(\theta_{\text{main}}) = \mathbb{E}_{q_0, \{y_i\}_{i=1}^{G}} \left[\frac{1}{G} \sum_{i=1}^{G} \sum_{t=1}^{|y_i|} m_{i,t} \min\big(\rho_{i,t}\, \hat{A}_i,\ \mathrm{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\, \hat{A}_i\big) \right], \tag{3}$$

where $\rho_{i,t} = \pi_\theta(y_{i,t} \mid y_{i,<t}) \,/\, \pi_{\theta_{\text{old}}}(y_{i,t} \mid y_{i,<t})$ is the importance sampling ratio and $\hat{A}_i$ is the group-normalized advantage. Following Jin et al. (2025), we apply a loss mask $m_{i,t}$ that equals 1 for tokens generated by the main agent and 0 for retrieved document tokens.

Reward. The main agent reward evaluates both answer correctness and format compliance:

$$r_{\text{main}}(q_0, y_i) = \begin{cases} s_{\text{ans}}, & \text{if } f = 1, \\ -\alpha, & \text{if } f = 0, \end{cases} \tag{4}$$

where s_{ans} is the token-level F1 score between the predicted and gold answers, $f \in \{0, 1\}$ indicates whether the trajectory follows the required ReAct format (Yao et al., 2023b; Jin et al., 2025), and $\alpha = 0.2$ is the format penalty.

3.3 Ranker Optimization

At each search step t of the i-th rollout, the ranker receives a tuple $(q_0, q_t^{(i)}, \mathcal{D}_N)$, where q_0 is the original user query, $q_t^{(i)}$ is the sub-query issued by the main agent, and $\mathcal{D}_N$ is the top-N documents retrieved by the dense retriever. The ranker first reasons about the relevance of each candidate, then selects and ranks a subset of K documents to form the observation $o_t^{(i)}$. We provide the detailed prompt and input–output examples in Appendix A.2. The ranker is trained with GRPO using the same clipped surrogate objective (Eq. 3).

However, directly applying GRPO to the ranker introduces a fundamental challenge. GRPO requires multiple responses sampled from *the same input prompt* to compute group-normalized advantages. For the main agent, this grouping is natural: all G trajectories originate from the same query q_0. For the ranker, however, the input includes the sub-query $q_t^{(i)}$, which varies across rollouts—different reasoning paths produce different sub-queries. Consequently, we cannot naively treat all ranker calls under the same q_0 as a single GRPO group.

A naive alternative would be to sample dedicated ranker rollouts for GRPO: for each of the $G \cdot m$ sub-queries produced by the main agent, generate H independent ranker responses and continue each resulting trajectory to completion to obtain a reward. However, each ranker response alters the observation o_t and consequently all downstream reasoning, producing new sub-queries that themselves require H rollouts—cascading across m steps into $G \cdot H^m$ total trajectories, an exponential blowup that is clearly intractable.

Semantic grouping and filtering. We address this challenge by exploiting a key observation: although sub-queries are not identical across rollouts, many are *semantically equivalent*—different reasoning paths tend to ask similar sub-questions, differing only in surface-level phrasing. For a fixed q_0, the G rollouts produce a pool of sub-queries $\mathcal{Q}(q_0) = \{q_t^{(i)} \mid i \in [1, G],\ t \in [1, T_i]\}$. We partition $\mathcal{Q}(q_0)$ into semantic groups $\mathcal{G}(q_0) = \{g_1, g_2, \ldots, g_M\}$ using the following greedy procedure. We iterate through the sub-queries and assign each query q to an existing group g_m if its token-level F1 similarity with the group's representative query q_m^{rep} exceeds a threshold δ:

$$q \in g_m \iff \text{F1}_{\text{token}}(q,\ q_m^{\text{rep}}) \geq \delta, \tag{5}$$

where δ is a similarity threshold. Token-level F1 treats each query as a bag of tokens and computes the harmonic mean of precision and recall. If no existing group matches, a new group is created with q as its representative. Groups with fewer than $k_{\min}$ members are discarded to maintain stable advantage estimation.

Each remaining group g_m contains ranker calls with semantically equivalent inputs under the same q_0, forming a valid GRPO group. Advantages are computed within each group using standard group normalization. This strategy incurs *zero additional sampling cost*: we reuse the trajectories already generated for the main agent, rather than performing separate rollouts for the ranker. We provide a detailed algorithm and an illustration in Appendix C.

3.4 Ranker Reward Design

The ranker aims to help the main agent solve a multi-step reasoning task by providing useful documents at each search step. However, reward design is non-trivial. A trajectory-level reward based on the correctness of the final answer cannot accurately assign credit to individual search steps, because each step's contribution is mixed with subsequent retrieval decisions and with the main agent's reasoning, making the signal noisy and unstable. A relevance-based reward also has an important limitation in our setting. Since pseudo-relevance labels are constructed by matching documents to the gold answer, they

fail to supervise intermediate retrieval steps whose retrieved documents are useful for decomposition or reasoning but do not directly mention the final answer. To address these two issues, we use a composite reward that combines trajectory-level supervision with relevance-based supervision.

Relevance reward. The relevance reward r_{rel} measures ranking quality using pseudo-relevance labels: a document is labeled pseudo-relevant if it contains the gold answer. We compute r_{rel} as the average of Hit@k over a set of cutoff values $\mathcal{K}$:

$$r_{\text{rel}} = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \text{Hit@}k,$$

where Hit@k indicates whether any pseudo-relevant document appears in the top-k positions of the ranker's output.

Main agent reward. The main agent reward r_{main} follows the same definition as Eq. 4, i.e., it is given by the token-level F1 score between the predicted and gold answers. Notice that Eq. 4 assigns a penalty $-\alpha$ when the main agent violates the required format ($f = 0$). In this case, we do not propagate this penalty to the ranker. Instead, we filter out such trajectories and exclude them from reward computation and subsequent optimization.

Composite reward. Let $I_{\text{ans}} \in \{0, 1\}$ indicate whether the candidate set $\mathcal{D}_N$ contains a pseudo-relevant document, and let γ be a threshold controlling the reward composition. The final ranker reward is:

$$r_{\text{rank}} = \begin{cases} -\alpha, & f = 0, \\ r_{\text{rel}}, & f = 1,\ I_{\text{ans}} = 1,\ r_{\text{rel}} \leq \gamma, \\ r_{\text{rel}} + r_{\text{main}}, & f = 1,\ I_{\text{ans}} = 1,\ r_{\text{rel}} > \gamma, \\ r_{\text{main}}, & f = 1,\ I_{\text{ans}} = 0,\ r_{\text{rel}} = 0. \end{cases} \tag{6}$$

The format indicator f and penalty α are the same as those in Eq. 4. The design follows three cases. When $I_{\text{ans}} = 1$, a pseudo-relevant document exists in the candidate set. If r_{rel} is low ($r_{\text{rel}} \leq \gamma$), it indicates that the ranker fails to rank relevant documents effectively. In this case, we rely solely on r_{rel} for supervision, since incorporating r_{main} would introduce noise from the main agent's reasoning process. When $r_{\text{rel}} > \gamma$, the ranking is sufficiently strong, and we additionally incorporate r_{main} to capture whether the retrieved documents contribute to the final answer. When $I_{\text{ans}} = 0$, no pseudo-relevant document exists in the candidate set, and we rely on r_{main} as the only available signal.

In sum, the composite reward r_{rank} prioritizes ranking precision when a relevant document is available but poorly ranked, adds trajectory-level credit when the ranking is already strong, and falls back to answer correctness alone when no relevant document exists in the candidate set.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on seven QA benchmarks spanning single-hop and multi-hop settings: PopQA (Mallen et al., 2022), Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (TQA) (Joshi et al., 2017), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (2Wiki) (Ho et al., 2020), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). We use the test splits from Search-R1 (Jin et al., 2025) for consistent comparison. Performance is measured by token-level F1 score.

Baselines. We compare against Direct Inference, CoT (Wei et al., 2023), RAG (Lewis et al., 2020), Search-o1 (Li et al., 2025a), Search-R1 (Jin et al., 2025), ZeroSearch (Sun et al., 2025a), and two internal variants: (1) **Retrieval Only**, which removes the ranker and directly uses the top-K documents from the dense retriever; and (2) **Fixed Ranker**, which uses a ranker fine-tuned on a relevance-based IR dataset but kept frozen during RL training.

Method	General QA			Multi-Hop QA				Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	Musique	Bamboogle	
Qwen2.5-7B-Instruct								
Direct Inference	0.279	0.504	0.215	0.292	0.304	0.118	0.372	0.298
CoT	0.198	0.456	0.150	0.244	0.264	0.095	0.310	0.245
RAG	0.420	0.689	0.387	0.371	0.244	0.148	0.296	0.365
Search-o1	0.345	0.526	0.248	0.316	0.286	0.126	0.422	0.324
Search-R1	0.556	0.681	**0.539**	0.576	0.547	0.272	0.558	0.533
ZeroSearch	0.562	0.693	0.521	0.575	0.559	0.291	0.572	0.539
Retrieval Only	0.564	0.728	0.508	0.572	0.556	0.320	0.574	0.546
Fixed Ranker	0.578	0.735	0.514	0.578	0.560	0.315	0.588	0.553
CoSearch	**0.582**	**0.747**	0.527	**0.595**	**0.587**	**0.334**	**0.601**	**0.568**
Oracle	0.680	0.764	0.684	0.626	0.584	0.393	0.652	0.626
Qwen2.5-3B-Instruct								
Direct Inference	0.198	0.343	0.168	0.202	0.249	0.055	0.068	0.180
CoT	0.191	0.363	0.159	0.197	0.229	0.070	0.100	0.187
RAG	0.317	0.478	0.291	0.371	0.193	0.110	0.058	0.223
Search-o1	0.329	0.501	0.351	0.225	0.128	0.096	0.216	0.236
Search-R1	0.458	0.637	0.460	0.450	0.397	0.179	0.414	0.428
ZeroSearch	0.451	0.642	0.471	0.455	0.387	0.191	0.431	0.433
Retrieval Only	0.465	0.651	0.478	0.454	0.406	0.189	0.436	0.440
Fixed Ranker	0.515	0.697	0.483	0.474	0.411	0.195	0.448	0.460
CoSearch	**0.538**	**0.714**	**0.489**	**0.478**	**0.428**	**0.193**	**0.457**	**0.471**
Oracle	0.548	0.749	0.612	0.589	0.510	0.308	0.589	0.558

Table 2: Main results (token-level F1) on seven QA benchmarks. Results are grouped by main agent backbone. The CoSearch ranker uses Qwen2.5-7B-Instruct in both settings. Best in **bold** within each scale block. Oracle (gray) promotes the answer-containing documents to rank 1, serving as an upper bound on ranking.

Implementation details. For the main agent, we evaluate both Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct (Yang et al., 2024) to assess performance across different model scales. The ranker in CoSearch is based on Qwen2.5-7B-Instruct. We use the 2018 Wikipedia dump (Karpukhin et al., 2020b) as the knowledge source and E5 base model (Wang et al., 2024) as the first-stage dense retriever. During rollout, each query generates $G = 8$ trajectories with up to 6 search calls per trajectory. The first-stage retriever returns $N = 50$ candidate passages, from which the ranker selects the top $K = 5$. Training uses a learning rate of 1×10^{-6}, rollout batch size of 512, and effective batch size of 128 for training, and sampling temperature of 1.0. For semantic grouping, we set the similarity threshold to $\delta = 0.8$ and minimum group size $k_{\min} = 3$. The composite reward threshold is $\gamma = 0.5$ and the Hit@k cutoff set is $\mathcal{K} = \{1, 3, 5\}$. Further details on training data and Fixed Ranker training are provided in Appendix D.

4.2 Main Results

Table 4.2 presents the full results across all seven benchmarks. We observe that:

Joint training consistently outperforms all baselines. CoSearch achieves an average F1 of 0.568 with the 7B agent, outperforming Search-R1 by a relative 6.6%, Retrieval Only by 4.0% and Fixed Ranker by 2.7%. The improvement holds across all seven benchmarks, indicating that jointly training the ranker provides a robust and general benefit. Among non-RL baselines, RAG improves over direct inference by 22.5% (0.365 vs. 0.298), and Search-R1 further surpasses RAG by 46.0% (0.533 vs. 0.365), confirming the value of iterative reasoning, yet CoSearch pushes performance further by optimizing the retrieval system itself.

Weaker agents benefit more from retrieval improvements. With a 3B main agent paired with a 7B ranker, CoSearch achieves a relative 7.0% gain over Retrieval Only and 2.4% over

Configuration	Single-Hop	Multi-Hop	Avg
CoSearch (full)	**0.619**	**0.529**	**0.568**
(a) Reward signal			
w/o r_{main}	0.612	0.508	0.560
w/o r_{rel}	0.591	0.513	0.552
Replace Hit@k with nDCG@k	0.605	0.519	0.556
Replace Composite with LLM-as-judge	0.608	0.520	0.558
(b) Semantic grouping			
w/o semantic grouping	0.595	0.503	0.549
(c) Architecture			
Shared backbone	0.613	0.518	0.559
Cross-Encoder Ranker	0.601	0.501	0.551
3B ranker	*diverged (training crash)*		

Table 3: Ablation study. The main agent uses Qwen2.5-7B-Instruct. Each row changes one component of the full model. Averages are computed over single-hop and multi-hop groups separately.

Fixed Ranker. Compared to the oracle upper bound, CoSearch closes approximately 27% of the retrieval gap for the 7B agent and 26% for the 3B agent, confirming that end-to-end RL optimization meaningfully narrows the performance ceiling.

4.3 Ablation Study

Table 4.3 reports ablation results. On reward design, removing the trajectory-level reward r_{main} reduces average F1 by a relative 1.4%, while removing the ranking-quality reward r_{rel} causes a larger drop of 2.8%, confirming that both components provide complementary supervision: r_{rel} supplies a direct signal on retrieval precision, while r_{main} captures whether retrieved documents actually contribute to final answer correctness. Replacing Hit@k with nDCG@k reduces performance by a relative 2.1%, and substituting the composite reward with an LLM-as-judge yields a 1.8% drop—both confirming that our composite design is effective and computationally lightweight.

Removing the semantic grouping and filtering strategy—treating all ranker calls sharing the same initial query q_0 as one group regardless of sub-query similarity—results in the largest single-component degradation of 3.3% relative (0.568 $\rightarrow$ 0.549). This confirms that mixing semantically heterogeneous sub-queries into a single GRPO group produces noisy advantage estimates that impede ranker learning, and that the filtering step described in §3.3 is critical for stable training.

Among architectural choices, a shared backbone degrades F1 by 1.6% due to conflicting ranking and reasoning gradients. A cross-encoder ranker drops 2.9%: its pointwise scoring is ill-suited for direct RL optimization. Scaling the ranker to 3B causes training to diverge entirely—the smaller model cannot reliably produce valid permutation outputs over 50 candidates (see Appendix A.2).

4.4 Analysis

Figure 2 compares all four systems across three dimensions over 100 training steps. Panels (a) and (b) reveal a clear positive correlation between ranking quality (Hit@5) and downstream F1, with CoSearch's ranker starting below the Fixed Ranker but steadily surpassing it during training. Panel (c) shows that CoSearch and Oracle require fewer search turns, suggesting higher-quality documents reduce the need for additional retrieval rounds.

Figure 3 tracks the ranker's training dynamics. In panels (a) and (c), pre-filter rewards eventually converge toward post-filter, suggesting the distributional gap is progressively mitigated. Panel (b) shows pre-filter relevance reward rising steadily (from $\sim$0.35 to $\sim$0.50), indicating genuine improvement in ranking ability. Panel (d) shows group size increasing

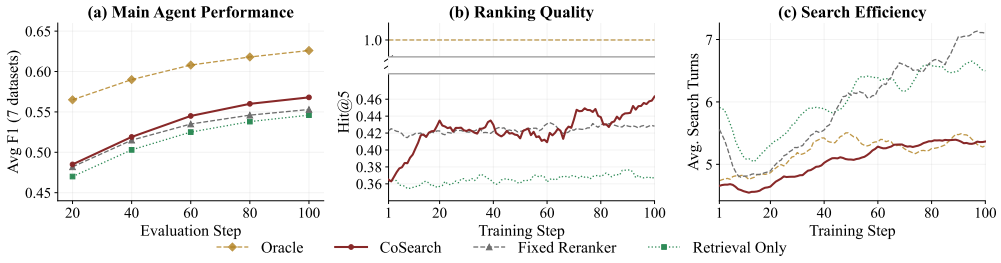

Figure 2: Training and evaluation dynamics across 100 steps. (a) Validation F1 (avg. over 7 datasets) every 20 steps. (b) Ranking quality (Hit@5) at each training step; y-axis broken to show Oracle = 1.0 above. (c) Average search turns per trajectory.

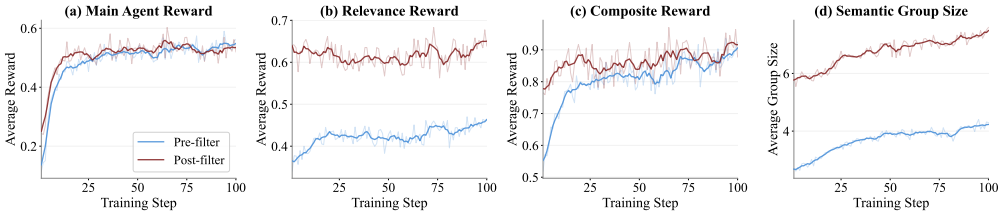

Figure 3: Ranker training dynamics across 100 steps. **Pre-filter**: statistics over all semantic groups; **post-filter**: statistics over groups that pass the minimum-size filter described in §3.3. (a) Main agent reward. (b) Relevance reward (Hit@5). (c) Composite reward. (d) Average semantic group size.

monotonically, with post-filter roughly twice pre-filter, reflecting increasingly consistent sub-queries as the policy converges.

K	Retrieval Only	Fixed Ranker	CoSearch	Oracle
1	0.498 (−8.8%)	0.513 (−7.2%)	0.538 (−5.3%)	0.612 (−2.2%)
3	0.531 (−2.7%)	0.539 (−2.5%)	0.561 (−1.2%)	0.631 (+0.8%)
5	0.546	0.553	**0.568**	0.626
10	0.555 (+1.6%)	0.559 (+1.1%)	0.571 (+0.5%)	0.639 (+2.1%)

Table 4: Effect of the number of ranked documents K on Avg F1. The ranker selects the top K from N=50 first-stage candidates. K=5 (bold) is the default used in Table 4.2. Percentages in parentheses indicate relative change from K=5 for each method.

Table 4.4 examines how K affects performance. When K is reduced from 5 to 1, CoSearch degrades by only 5.3%, comparable to Oracle (−2.2%) and far less than Retrieval Only (−8.8%) and Fixed Ranker (−7.2%). Notably, CoSearch at K=3 (0.561) already surpasses Fixed Ranker at K=10 (0.559), and increasing K from 5 to 10 yields diminishing returns across all methods (+0.5% to +1.6%).

Conclusion

In this work, we identified retrieval quality as a significant bottleneck for agentic search systems, with oracle retrieval gaps of up to +26.8% relative F1. Motivated by this finding, we presented CoSearch, a framework that jointly trains a reasoning agent and a generative document ranker via RL. Our semantic grouping strategy enables efficient GRPO training for the ranker without additional rollouts, and our composite reward provides both immediate ranking-quality and long-term trajectory-level signals. Experiments on seven QA benchmarks demonstrate consistent improvements, achieving +6.6% relative F1 over Search-R1 with a 7B agent and +10.0% with a 3B agent. Our results show that joint training of reasoning and retrieval is not only feasible but strongly performant, pointing to a key ingredient for future search agents.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #2402873, and in part by the Office of Naval Research under contract #N000142412612. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024. URL https://doi.org/10.18653/v1/2024.acl-long.662.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Sebastian Borgeaud et al. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic entropy-balanced policy optimization, 2025a. URL https://arxiv.org/abs/2510.14545.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. *CoRR*, abs/2507.19849, 2025b. doi: 10.48550/ARXIV.2507.19849. URL https://doi.org/10.48550/arXiv.2507.19849.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025a. URL https://arxiv.org/abs/2504.11536.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. In *Advances in Neural Information Processing Systems*, 2025b. doi: 10.48550/arXiv.2505.10978. URL https://arxiv.org/abs/2505.10978.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*, 2025.

Google. Gemini deep research — your personal research assistant. https://gemini.google/overview/deep-research/, 2025. Accessed: 2025-12-18.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. Deeprag: Thinking to retrieve step by step for large language models, 2025. URL https://arxiv.org/abs/2502.01142.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

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. In *The Second Conference on Language Modeling (COLM)*, 2025.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

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 Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering. *ArXiv*, abs/2004.04906, 2020b. URL https://api.semanticscholar.org/CorpusID:215737187.

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback, 2025. URL https://openreview.net/forum?id=f70kIurx4b.

Omar Khattab, Christopher Potts, and Matei Zaharia. Baleen: robust multi-hop reasoning at scale via condensed retrieval. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Tom Kwiatkowski et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Nathan Lambert et al. Tülu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, NeurIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Mingchen Li, Hansi Zeng, Zhuoyang Qian, Jiatan Huang, Hamed Zamani, and Hongkun Yu. Rice-po: Turning retrieval interactions into credit signals for reasoning agents. *ArXiv*, abs/2605.26352, 2026. URL https://api.semanticscholar.org/CorpusID:288671738.

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. *CoRR*, abs/2501.05366, 2025a. doi: 10.48550/ARXIV.2501.05366. URL https://doi.org/10.48550/arXiv.2501.05366.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. In *Advances in Neural Information Processing Systems*, 2025b. URL https://arxiv.org/abs/2504.21776.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl, 2025c. URL https://arxiv.org/abs/2503.23383.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Yuchen Li, Daiting Shi, Dawei Yin, and Zhicheng Dou. Agentic-r: Learning to retrieve for agentic search. *ArXiv*, abs/2601.11888, 2026. URL https://api.semanticscholar.org/CorpusID:284910800.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint*, 2022.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Ouyang Long, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback. *ArXiv*, abs/2112.09332, 2021. URL https://api.semanticscholar.org/CorpusID:245329531.

OpenAI. Learning to reason with llms, 2024. URL https://openai.com/index/learning-to-reason-with-llms.

OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, 2025. Accessed: 2025-12-18.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy J. Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *ArXiv*, abs/2312.02724, 2023. URL https://api.semanticscholar.org/CorpusID:265659387.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Devendra Singh Sachan, Siva Reddy, William L. Hamilton, Chris Dyer, and Dani Yogatama. End-to-end training of multi-document reader and retriever for open-domain question answering. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=5KWmB6JePx.

Alireza Salemi and Hamed Zamani. Towards a search engine for machines: Unified ranking for multiple retrieval-augmented large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 741–751, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657733. URL https://doi.org/10.1145/3626772.3657733.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, 2024a.

Zhengliang Shi, Shuo Zhang, Weiwei Sun, Shen Gao, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. Generate-then-ground in retrieval-augmented generation for multi-hop question answering. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7339–7353, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.397. URL https://aclanthology.org/2024.acl-long.397/.

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, 2025.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching, 2025a.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, and Ji-Rong Wen. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025b. URL https://aclanthology.org/2025.findings-emnlp.739/.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agent. *ArXiv*, abs/2304.09542, 2023.

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. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557.

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, 2024. URL https://arxiv.org/abs/2212.03533.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica Lam, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Fei-Fei Li, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

An Yang et al. 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 Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural*

Language Processing, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023b. URL https://arxiv.org/abs/2210.03629.

Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. Retrieval-enhanced machine learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pp. 2875–2886, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531722. URL https://doi.org/10.1145/3477495.3531722.

Hansi Zeng, Hamed Zamani, and Vishwa Vinay. Curriculum learning for dense retrieval distillation. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022. URL https://api.semanticscholar.org/CorpusID:248426770.

Hansi Zeng, Surya Kallumadi, Zaid Alibadi, Rodrigo Nogueira, and Hamed Zamani. A personalized dense retrieval framework for unified information access. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023a. URL https://api.semanticscholar.org/CorpusID:258332095.

Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. Scalable and effective generative information retrieval. *Proceedings of the ACM Web Conference 2024*, 2023b. URL https://api.semanticscholar.org/CorpusID:265213270.

Hansi Zeng, Chen Luo, and Hamed Zamani. Planning ahead in generative retrieval: Guiding autoregressive generation through simultaneous decoding. *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. URL https://api.semanticscholar.org/CorpusID:269302879.

Hansi Zeng, Julian Killingback, and Hamed Zamani. Scaling sparse and dense retrieval in decoder-only llms. *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025. URL https://api.semanticscholar.org/CorpusID:276558404.

Hansi Zeng, Zoey Li, Yifan Gao, Chenwei Zhang, Xiaoman Pan, Tao Yang, Fengran Mo, Jiacheng Lin, Xian Li, and Jingbo Shang. Synplanresearch-r1: Encouraging tool exploration for deep research with synthetic plans, 2026. URL https://arxiv.org/abs/2603.07853.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments, 2025. URL https://arxiv.org/abs/2504.03160.

Appendix

A Agent Prompts

A.1 Main Agent Prompt

The main reasoning agent operates in a REACT-style loop with a single `search` tool. At each turn, the agent outputs a `<reason>` block followed by either a `<tool_call>` block (to issue a search query) or an `<answer>` block (to finalize its response). The full prompts used across all experiments are presented below.

```
You are a tool-augmented research agent for wiki-based factoid question answering.

Your task is to answer questions drawn from Wikipedia-style datasets.
The final answer is evaluated using exact match (EM) or token-level F1, so it must be short and precise.

You have ONE tool available:
- search(query: string) -> returns a list of Wikipedia passages

============================================================
CRITICAL OUTPUT FORMAT (MUST FOLLOW EXACTLY)
============================================================

For EVERY assistant turn, you MUST output EXACTLY TWO TAG BLOCKS in this order:

1) <reason> ... </reason>
2) EITHER:
   (A) <tool_call> ... </tool_call>
   OR
   (B) <answer> ... </answer>

No other text is allowed outside these tags.
Do NOT output <tool_response>. The environment will provide tool results separately.

Allowed patterns:
- <reason> ... </reason>
  <tool_call> ... </tool_call>

- <reason> ... </reason>
  <answer> ... </answer>

If you violate the format, your output is invalid.

============================================================
TOOL CALL JSON SCHEMA (STRICT)
============================================================

When calling the tool, the <tool_call> block MUST contain ONLY a valid JSON object:

<tool_call>
{
  "name": "search",
  "arguments": {
    "query": "<string>"
  }
}
</tool_call>

Rules:
- "name" MUST be exactly "search"
- "arguments" MUST be an object
- "query" MUST be a single string
- Do NOT add extra keys
- Do NOT wrap JSON in Markdown
- Do NOT include comments, trailing commas, or natural language

============================================================
GENERAL TOOL USAGE
============================================================

Use the search tool whenever additional evidence would help you determine the correct answer.
If you believe you already have sufficient information to answer correctly, answer directly.

You may use multiple search calls across turns.

============================================================
```

```
SEARCH GUIDELINES
============================================================

- Write search queries that are clear and specific to what you want to confirm or find.
- After receiving evidence, reassess whether you can answer; if not, search again with a refined query.

============================================================
REASONING CONTENT REQUIREMENTS
============================================================

- Do NOT include tool JSON inside <reason>.
- Do NOT include <tool_call> or <answer> tags inside <reason>.

============================================================
ANSWER REQUIREMENTS (STRICT: SHORT ANSWER)
============================================================

Inside <answer>, you MUST:
- Output ONLY the final answer string
- Do NOT include explanations, reasoning, or extra text
- Do NOT include citations, sources, or formatting
- Use a concise canonical form (Wikipedia-style when possible)

Examples of valid answers:
- Paris
- 1997
- George Washington
- The Lord of the Rings

If the expected answer type is a person/place/organization/title/date, output only that span.
If multiple surface forms are possible, output the most standard form.

============================================================
INTEGRITY
============================================================

- Do not fabricate facts.
- If you are uncertain, use search to verify.
- If evidence is conflicting, search again with a query that resolves the conflict.

============================================================
BEGIN
============================================================

Question: {question}
```

A.2 Ranker Prompt

The ranker receives the original user query q_0, the sub-query q_t issued by the main agent at step t, and the top-N candidate documents $\mathcal{D}_N$ returned by the dense retriever. It outputs a `<reason>` block containing its relevance analysis, followed by a `<rerank>` block listing the selected K document indices in descending order of relevance (e.g., `[3] > [1] > [5] > [2]`). The full prompt template is shown below.

```
You are a document ranking agent assisting a search-augmented reasoning system.

You will be given:
- An original user question
- A sub-query issued by the reasoning agent at the current search step
- A list of N candidate documents retrieved by a dense retriever,
  each labeled [1], [2], ..., [N]

Your task is to select the K most relevant documents and rank them in descending
order of relevance to help the reasoning agent answer the original question.

============================================================
CRITICAL OUTPUT FORMAT (MUST FOLLOW EXACTLY)
============================================================

You MUST output EXACTLY TWO TAG BLOCKS in this order:

1) <reason> ... </reason>
2) <rerank> ... </rerank>

No other text is allowed outside these tags.
```

```
============================================================
REASON BLOCK
============================================================

Inside <reason>, analyze the relevance of each candidate document.
Consider whether the document provides factual evidence that can help answer the
original question, taking the sub-query into account as additional context.

============================================================
RERANK BLOCK
============================================================

Inside <rerank>, list exactly K document indices in descending order of relevance:

<rerank>[id1] > [id2] > ... > [idK]</rerank>

Rules:
- Use only the provided document indices (e.g., [1], [3], [7])
- Select exactly K indices; do NOT select more or fewer
- Do NOT duplicate indices
- Do NOT include any explanation inside <rerank>

============================================================
BEGIN
============================================================

Original Question: {original_question}
Sub-Query: {sub_query}

Candidate Documents:
{documents}
```

A.3 Ranker Input/Output Examples

We present two concrete examples of ranker inputs and outputs at training step 100. Each example shows the original question, the sub-query issued by the main agent at a single retrieval step, the first five of the N=50 candidate passages (truncated), and the ranker's full output comprising a relevance analysis (<reason>) followed by a ranked list of K=5 document indices (<rerank>).

Example 1: answer-containing document present ($I_{\text{ans}} = 1$), high composite reward Golden answer: *Harpswell*. answer_in_docs = true, $r_{\text{rel}} = 1.00$, $r_{\text{main}} = 1.00$, $r_{\text{total}} = 2.00$.

Input (first 5 of 50 candidates shown):

```
Original Question: What town has an island with a community building built in 1911-12?
Sub-Query: island with community building built 1911-12

Passages (50 total):
[1] "Bailey Island Library Hall"
Bailey Island Library Hall (locally just Library Hall) is a historic community building
at 2167 Harpswell Island Road...
[2] "Bumpkin Island"
Bumpkin Island, also known as Round Island, Bomkin Island, Bumkin Island, or Ward's Island,
is an island in the Hingham Bay area of the Bosto...
[3] "Chocorua Island Chapel"
Although the camp had fallen into disrepair by that date, the chapel continued to provide
worship services for visitors and area residents. Ev...
[4] "Gooden Grant House"
octagonal tower with turreted roof projects at the southwest corner, and a
partially-enclosed single-story porch wraps around the west and south sides...
[5] "Cherry Grove Community House and Theatre"
almost 400 other such sites have been identified as candidates. Cherry Grove Community
House and Theatre The Cherry Grove Communit...
... [45 more candidates not shown]
```

Output:

```
<reason>
The Initial Query asks for a town with an island that has a community building built in
1911-12. Passage [1] directly mentions a community building built in 1911-12 and locates
it on Harpswell Island Road, making it the top choice. Passage [48] also fits the criteria
but contains less detailed information. Passages [2], [32] discuss islands but do not
```

```
mention a community building from that period. Passage [39] describes a small island with
no matching structure.
</reason>
<rerank>[1] > [48] > [2] > [32] > [39]</rerank>
```

Example 2: answer-containing document absent ($I_{\text{ans}} = 0$), ranker relies on r_{main} Golden answer: *yes*. `answer_in_docs` = false, $r_{\text{rel}} = 0.00$, $r_{\text{main}} = 1.00$, $r_{\text{total}} = 1.00$.

Input (first 5 of 50 candidates shown):

```
Original Question: Are Naked Obsession and New York (1916 Film) from the same country?
Sub-Query: New York (1916 Film)

Passages (50 total):
[1] "New York (1916 film)"
New York (1916 film) New York is a lost 1916 American silent comedy drama film directed by
George Fitzmaurice and starring Florence Reed...
[2] "New York (1916 film)"
thus Oliver King becomes a benedict, and Reel 3 contains two views of a nude model.
New York is a lost 1916 American silent comedy drama film...
[3] "Lights of New York (1916 film)"
Lights of New York is a 1916 American silent drama film directed by Van Dyke Brooke.
Produced by the Vitagrap...
[4] "Lights of New York (1916 film)"
with him anyway. Lights of New York (1916 film) Lights of New York is a 1916 American
silent drama film directed by Van Dyke Brooke...
[5] "The Pride of New York"
German officer and nurse in room where bed is shown including taking nurse to room and
excluding other young woman...
... [45 more candidates not shown]
```

Output:

```
<reason>
The Initial Query asks whether Naked Obsession and New York (1916 Film) are from the same
country. The Sub-Query targets New York (1916 Film). Passages [1] and [2] directly
describe this film and confirm its American origin, making them the top choices. Passages
[3] and [4] describe a different 1916 film (Lights of New York) that is also American but
less directly relevant. Passage [5] discusses yet another unrelated film and ranks lowest.
</reason>
<rerank>[1] > [2] > [3] > [4] > [5]</rerank>
```

B Two-Stage Retrieval System

At each retrieval step t, the two-stage retrieval system takes the original query q_0, the sub-query q_t issued by the main agent, and the corpus $\mathcal{C}$, and returns an observation o_t consisting of K ranked documents. The first stage is a fixed dense retriever that efficiently recalls N candidate documents; the second stage is the generative ranker $\pi_{\theta_{\text{gr}}}$, which reasons about candidate relevance and outputs an explicit ranked list. Algorithm 1 gives the full procedure. For the prompt format and concrete input–output examples of the generative ranker, see Appendix A.2.

Algorithm 1 Two-Stage Retrieval at Step t

Require: original query q_0, sub-query q_t, corpus $\mathcal{C}$, retrieval size N, ranking size K
Ensure: observation $o_t = \mathcal{D}_K$ (top-K ranked documents)

$\mathcal{D}_N \leftarrow \textsc{DenseRetrieve}(q_t, \mathcal{C}, N)$ ▷ fixed retriever
$I_{\text{ans}} \leftarrow \mathbf{1}[\text{gold answer} \in \mathcal{D}_N]$ ▷ used for composite reward, §3.4
$prompt \leftarrow \textsc{BuildPrompt}(q_0, q_t, \mathcal{D}_N, K)$
$output \leftarrow \pi_{\theta_{\text{gr}}}(prompt)$ ▷ generative ranker produces ranked list
$\mathcal{D}_K \leftarrow \textsc{ParseRanking}(output, \mathcal{D}_N, K)$ ▷ extract top-K documents
return $\mathcal{D}_K$

C Semantic Grouping Algorithm

Applying GRPO to the ranker requires grouping ranker calls with the same input prompt. Since sub-queries vary across rollouts, we perform a two-level grouping: first split by whether the candidate set contains the gold answer (I_{ans}), then cluster within each split by token-level F1 similarity. Figure 4 illustrates the procedure for a simplified example with G=5 rollouts and up to 5 search steps per trajectory; in our experiments we use G=8 rollouts with up to 6 search steps. Algorithm 2 gives the full procedure.

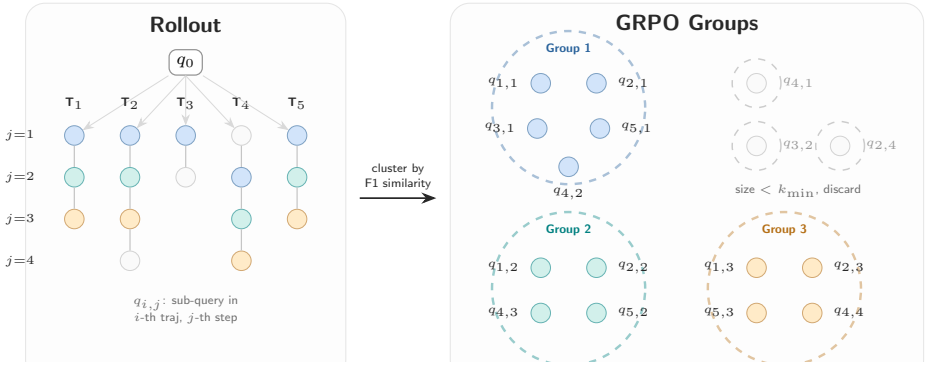

Figure 4: An illustration of semantic grouping for GRPO training, shown for G=5 rollout trajectories with up to 5 search steps each. Left: each dot is a sub-query $q_{i,j}$ (trajectory i, step j) colored by group membership. Right: all sub-queries are clustered by token-level F1 similarity into GRPO groups; gray singletons (size $< k_{\min}$) are discarded before optimization.

D Additional Implementation Details

Training data. The RL training set consists of 51,200 questions drawn from four datasets: Natural Questions (20,480 questions, 40%), HotpotQA (14,220, 28%), Musique (9,000, 18%), and 2WikiMultiHopQA (7,500, 15%). All questions are sampled from each dataset's official training split. For evaluation, we use the official test split of each benchmark, or the evaluation split when no test split is available.

Fixed Ranker training. The Fixed Ranker baseline is trained as follows. We randomly sample 50K queries from MS MARCO and 50K from Natural Questions, then use `e5-base` to retrieve the top-50 documents for each query. Queries for which no relevant document appears in the top-50 are discarded, yielding 92,160 training queries. The ranker is trained with RL using the same Hit@$\{1, 3, 5\}$ composite reward and the same generative listwise output format as the jointly trained ranker in CoSearch.

E Semantic Grouping Examples

Tables 5 and 6 present 20 sampled initial queries from the step-100 rollout along with their sub-queries after semantic grouping and filtering. For each initial query, the G=8 rollout trajectories produce a pool of sub-queries; these are clustered by token-level F1 similarity with threshold δ=0.8. Column **C** indicates the cluster index assigned to each sub-query. The table illustrates that multi-hop questions naturally produce multiple distinct semantic clusters (e.g., one cluster for each sub-question in a two-hop chain), while simpler questions tend to produce a single large cluster of near-paraphrases.

Initial Query	C	Sub-query
who has played the most state of origins	0	who has played the most state of origins
	1	current record holder for most State of Origin appearances
	2	most State of Origin appearances
	2	most appearances in State of Origin
what is the host nation of the 2004 summer olympic g...	0	host nation of 2004 Summer Olympic Games
	1	host nation of 2004 Summer Olympics
	1	host nation of the 2004 Summer Olympics
What is the birth city of Blessed John the Fool-For-...	0	capital of former Soviet Union and country with AGT show
	0	capital of the former Soviet Union and country with AGT
	1	Blessed John the Fool-For-Christ birth city
	1	Blessed John the Fool-For-Christ birth city Azerbaijan
	1	Blessed John the Fool-For-Christ birth city Baku Azerbaijan
What race is the majority of the population of the c...	0	The Unbeatables I production country
	1	The Unbeatables I
	1	The Unbeatables I produced
Who plays the writer who mentioned The Angel Pub in ...	0	Charles Dickens The Angel Pub
	0	Charles Dickens The Angel Pub writings
	0	Charles Dickens mentioned The Angel Pub
	1	The Man Who Invented Christmas
Are Naked Obsession and New York (1916 Film)...	0	Naked Obsession
	1	New York (1916 Film)
Standard Chartered Bank, who sponsors the annual Hon...	0	Standard Chartered Bank headquarters
	1	Standard Chartered Bank sponsors Hong Kong Marathon
on what day of the year is the sun near the star reg...	0	when is the sun near Regulus (Alpha Leo) on the calendar
	0	when is the sun near Regulus (Alpha Leo) on the exact date
	0	when is the sun near Regulus (Alpha Leo) on the year
	1	day of the year when the sun is near Regulus (Alpha Leo)
	1	day of the year when the sun is near Regulus (alpha leo)
	1	exact day of the year when the Sun is near Regulus (Alpha Leo)
who did the congress send to london as a minister in...	0	who did the congress send to london as a minister in 1784
	1	who was sent as a minister to london by the congress in 1784
	1	who was sent as a minister to london in 1784
	1	who was sent as minister to london by congress in 1784
In what season of the 2017 year is the Netflix Germa...	0	Dark (TV series) 2017 release season
	0	Dark (TV series) release schedule 2017
	1	Dark (Netflix German series) season release
	1	Dark Netflix German series 2017 release season
	1	Dark Netflix German series release season

Table 5: Sub-queries after semantic grouping and filtering (Part 1 of 2, training step 100). For each initial query, sub-queries from G=8 rollouts are clustered by token-level F1 similarity (δ=0.8). Column **C** denotes the cluster index.

Initial Query	C	Sub-query
What part did The King of Hollywood play in China Seas?	0	The King of Hollywood China Seas
	0	The King of Hollywood China Seas part
	1	Clark Gable role in China Seas
	1	Clark Gable's role in China Seas
What is the Izzo (H.O.V.A.) performer's record label?	0	Izzo (H.O.V.A.) performer record label
	1	Jay-Z current record label
	1	Jay-Z record label
Where was the place of death of the director of film...	0	director of film When A Man Sees Red 1934
	0	director of film When A Man Sees Red 1934 Pursued
	0	director of film When A Man Sees Red 1934 version
	1	Frank Ellis death place
	1	Frank Ellis place of death
	1	place of death Frank Ellis
Who was born later, Dani Pacheco or Agnė Čepelytė?	0	Dani Pacheco birth year
	1	Agnė Čepelytė birth year
Who is the operator of Embassy of Northern Cyprus in...	0	Embassy of Northern Cyprus in Istanbul operator
	0	Embassy of Northern Cyprus in Kemerhisar operator
	0	Embassy of Northern Cyprus in Samsun operator
What town has an island with a community building bu...	0	island with community building built 1911-12
	0	town with an island and a community building built in 1911-12
	0	town with island and community building built in 1911-12
The M66 is a motorway in Lancashire and Greater Manc...	0	M66 motorway in Lancashire and Greater Manchester
	0	M66 motorway in Lancashire and Greater Manchester, England
who plays the queen of hearts in alice and wonderland	0	who plays the Queen of Hearts in Alice and Wonderland
	0	who plays the Queen of Hearts in Alice in Wonderland
	0	who plays the Queen of Hearts in recent Alice in Wonderland films
What nationality is the director of film Porky'S Rev...	1	James Komack nationality
What presenter of Market Kitchen has won a Guild of ...	0	Market Kitchen presenter Guild of Food Writers award
	0	Market Kitchen presenter who won Guild of Food Writers award
	0	Market Kitchen presenter won Guild of Food Writers award

Table 6: Sub-queries after semantic grouping and filtering (Part 2 of 2, training step 100). For each initial query, sub-queries from G=8 rollouts are clustered by token-level F1 similarity (δ=0.8). Column **C** denotes the cluster index.

Algorithm 2 Semantic Grouping for Ranker GRPO Training

Require: ranker calls $\mathcal{O}$ (from G rollouts of q_0), similarity threshold δ, minimum group size $k_{\min}$
Ensure: UID-labeled ranker calls ready for GRPO

// Step 1: Split by answer availability
Partition $\mathcal{O}$ into $\mathcal{O}_{\text{easy}}$ ($I_{\text{ans}} = 1$) and $\mathcal{O}_{\text{hard}}$ ($I_{\text{ans}} = 0$)
for each split $\mathcal{B} \in \{\mathcal{O}_{\text{easy}}, \mathcal{O}_{\text{hard}}\}$ **do**
  // Step 2: Greedy cluster by token-level F1
  clusters $\leftarrow \{\}$ ▷ cluster_id $\rightarrow$ representative sub-query
  for each call $o \in \mathcal{B}$ **do**
    $q \leftarrow \text{NORMALIZE}(o.\text{sub_query})$
    matched $\leftarrow$ False
    for each $(c, q^{\text{rep}}) \in$ *clusters* **do**
      if $\text{F1}_{\text{token}}(q, q^{\text{rep}}) \geq \delta$ **then**
        assign $o \rightarrow$ cluster c; *matched* $\leftarrow$ True; **break**
      end if
    end for
    if not *matched* **then**
      create new cluster with q as representative; assign $o \rightarrow$ new cluster
    end if
  end for
  // Step 3: Discard small groups
  Remove all clusters with $|\text{cluster}| < k_{\min}$
  // Step 4: Assign UIDs
  for each surviving cluster c **do**
    uid $\leftarrow$ `{main_uid}_{easy|hard}_cluster_{c}`
    assign uid to all $o \in c$
  end for
end for

F Oracle Retrieval Construction

The oracle retrieval experiment (Table 1) measures the performance gap caused by imperfect retrieval. Unlike post-hoc analysis, the oracle trajectory is constructed *online* during rollout: the agent's reasoning at each step is conditioned on the (potentially modified) observation, so improved retrieval at step t influences all subsequent reasoning and sub-queries.

Concretely, given an initial query q_0 with gold answer a^*, the agent generates a trajectory step by step following Eq. 1. At each step t, the agent produces a reasoning thought τ_t and sub-query q_t, and the dense retriever returns a candidate set $\mathcal{D}_N$. Let $\mathcal{D}^+ = \{d \in \mathcal{D}_N \mid a^* \subseteq d\}$ denote the documents in $\mathcal{D}_N$ that contain the gold answer. The observation presented to the agent is:

$$o_t = \begin{cases} \mathcal{D}^+ \,\|\, \text{top-}(K - |\mathcal{D}^+|) \text{ from } \mathcal{D}_N \setminus \mathcal{D}^+, & \text{if } \mathcal{D}^+ \neq \emptyset, \\ \text{top-}K \text{ from } \mathcal{D}_N, & \text{if } \mathcal{D}^+ = \emptyset, \end{cases} \tag{7}$$

where $\|$ denotes concatenation. When documents matching the gold answer exist in the candidate set, we promote them to the top positions and fill the remaining slots with the highest-ranked non-matching documents. For intermediate sub-queries where no retrieved document matches the gold answer, we use the default retriever ranking unchanged. The agent then observes o_t, continues reasoning, and repeats until it produces a final answer a. Algorithm 3 provides the full procedure.

G Search Turn Distribution

Table G shows the full search turn distribution. CoSearch and Oracle concentrate 91% and 87% of questions at 1–2 turns respectively, while Fixed Ranker pushes 55% of questions to 3 turns, requiring significantly more retrieval rounds on average (2.62 vs. 1.67). This

Algorithm 3 Oracle Retrieval Rollout

Require: Initial query q_0; gold answer a^*; main agent $\pi_{\theta_{\text{main}}}$; candidate set size N; output size K
Ensure: Oracle trajectory y^{oracle}

$t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  $\tau_t, q_t \leftarrow \pi_{\theta_{\text{main}}}(q_0, \tau_1, q_1, o_1, \ldots, \tau_{t-1}, q_{t-1}, o_{t-1})$ ▷ agent reasons and generates sub-query
  $\mathcal{D}_N \leftarrow \text{DenseRetrieve}(q_t, N)$
  $\mathcal{D}^+ \leftarrow \{d \in \mathcal{D}_N \mid a^* \subseteq d\}$ ▷ gold-answer-matching documents
  if $\mathcal{D}^+ \neq \emptyset$ **then**
    $o_t \leftarrow \mathcal{D}^+ \,\|\, \text{top-}(K - |\mathcal{D}^+|) \text{ from } \mathcal{D}_N \setminus \mathcal{D}^+$ ▷ promote to top
  else
    $o_t \leftarrow \text{top-}K \text{ from } \mathcal{D}_N$ ▷ keep default ranking
  end if
until agent outputs final answer a
return $y^{\text{oracle}} = (q_0, \tau_1, q_1, o_1, \ldots, \tau_m, q_m, o_m, \tau_{m+1}, a)$

Model	0	1	2	3	4	5	6	Avg
Oracle	0.0	49.3	38.2	9.8	2.3	0.3	0.1	1.66
CoSearch	**0.0**	**42.3**	**48.9**	**8.1**	**0.7**	**0.0**	**0.0**	**1.67**
Fixed Reranker	0.0	4.2	35.7	54.5	5.6	0.0	0.0	2.62
Retrieval Only	0.0	33.2	40.4	16.3	7.9	1.8	0.3	2.05

Table 7: Search turn distribution (% of questions) at validation step 100. **Avg** is the mean number of search turns per question.

confirms that higher-quality ranking reduces the number of retrieval steps needed to resolve a question.