s3: You Don't Need That Much Data to Train a Search Agent via RL

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) systems empower large language models (LLMs) to access external knowledge during inference. Recent advances have enabled LLMs to act as search agents via reinforcement learning (RL), improving information acquisition through multi-turn interactions with retrieval engines. However, existing approaches either optimize retrieval using search-only metrics (e.g., NDCG) that ignore downstream utility or fine-tune the entire LLM to jointly reason and retrieve—entangling retrieval with generation and limiting the real search utility and compatibility with frozen or proprietary models. In this work, we propose s3, a lightweight, model-agnostic framework that decouples the searcher from the generator and trains the searcher using a Gain Beyond RAG reward: the improvement in generation accuracy over naïve RAG. s3 requires only 2.4k training samples to outperform baselines trained on over 70× more data, consistently delivering stronger downstream performance across six general QA and five medical QA benchmarks.[1]

1 Introduction

Retrieval-Augmented Generation (RAG) enables large language models (LLMs) to access and reason over external knowledge by retrieving relevant documents and conditioning generation on them (Lewis et al., 2020). As shown in Figure 2, we categorize the evolution of RAG systems into three phases.

Classic RAG. Early approaches relied on static retrieval methods, where queries were fixed and retrieval quality was decoupled from downstream generation performance. Despite their simplicity, these systems often underperformed on queries that need contextual or multi-hop reasoning.

[1]Our code is available at https://anonymous.4open.science/r/s3-94BF.

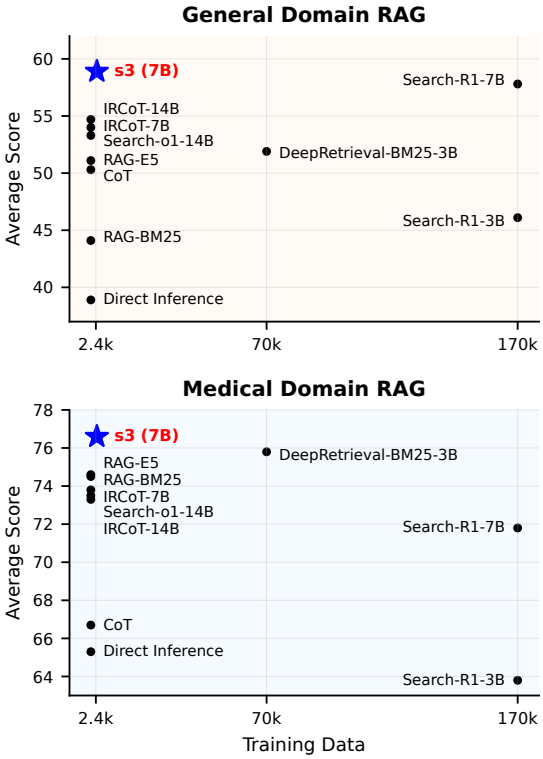

Figure 1: **Training Data vs Averaged Performance across six general and five medical QA Datasets** (tested with Claude-3-Haiku as the generator LLM).

Pre-RL-Zero. To improve retrieval quality, subsequent methods enabled more active participation of the LLM during inference. Active RAG techniques (Yao et al., 2022; Jiang et al., 2023; Trivedi et al., 2023a) interleaved query generation, retrieval, and reasoning in a multi-turn loop. These systems introduced iterative retrieval but typically relied on zero-shot prompting and lacked trainable components. Self-RAG (Asai et al., 2023) distilled such behaviors from larger models into smaller ones via supervised fine-tuning, teaching smaller models to reason and retrieve effectively without external rewards. While these methods improved flexibility and reduced supervision cost, they still did not optimize retrieval using outcome signals.

RL-Zero. The recent emergence of reinforcement

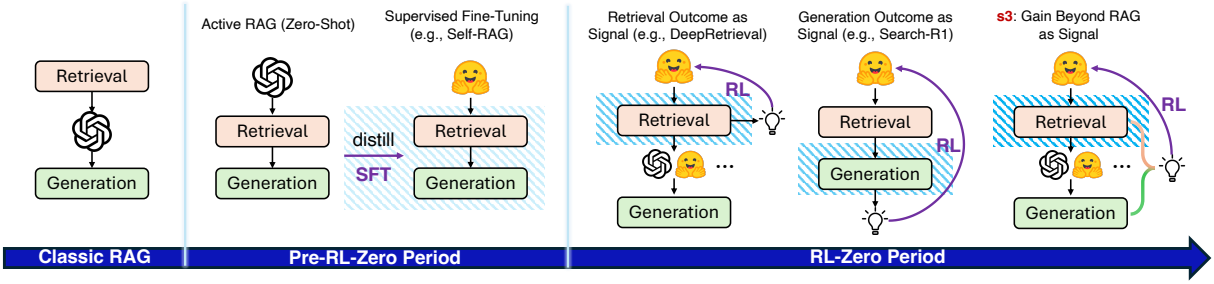

Figure 2: RAG has progressed from fixed or supervised retrieval to RL-based agentic methods. While prior work trains retrieval or generation jointly, s3 focuses solely on the searcher, improving generation without tuning the generator LLM.

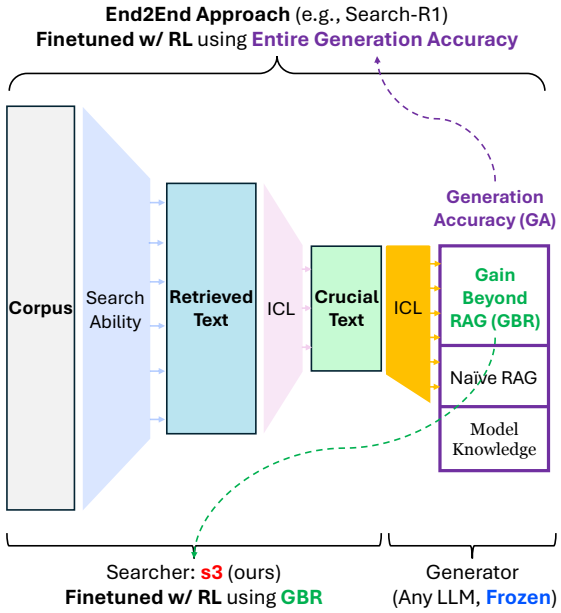

Figure 3: **Decomposition of Agentic RAG.** End-to-end approaches fine-tune the entire model using the entire generation accuracy, making it difficult to isolate the contribution of search. In contrast, s3 freezes the generator and trains only the searcher with Gain Beyond RAG (GBR), a novel reward that quantifies the added value of retrieved context over naïve RAG, enabling modular, efficient optimization.

learning for retrieval marks a new phase—what we refer to as the *RL-Zero* era. DeepSeek-R1-Zero (Guo et al., 2025) showed that even rule-based, outcome-driven rewards (e.g., answer correctness) can train strong reasoning agents. Building on this idea, DeepRetrieval (Jiang et al., 2025) applied RL to train query generators using search-oriented metrics like recall and NDCG. However, these metrics are disconnected from downstream answer quality. Search-R1 (Jin et al., 2025) trained a single model to jointly retrieve and generate via reinforcement learning, using exact match (EM) as the reward. While this approach improves answer accuracy, the tight entanglement between search and generation makes it difficult to isolate genuine retrieval improvements (see Figure 3). Moreover, EM is a brittle reward signal—failing to reward semantically correct answers phrased differently.

This motivates a shift toward a modular framework where search and generation are cleanly separated, and optimization focuses purely on search quality with respect to downstream utility (Dai et al., 2025). We propose s3, a simple yet powerful framework that trains a search-only agent using a *novel* reward signal: *Gain Beyond RAG* (GBR). GBR measures how much better the generator performs when conditioned on retrieved documents from s3, compared to naive top-k retrieval. This setup keeps the generator LLM frozen, sidesteps answer token overfitting, and directly optimizes the retrieval component to serve any black-box LLM.

Remarkably, s3 achieves strong gains with only 2.4k training examples, outperforming DeepRetrieval (focused on retrieval metrics) and Search-R1 (entangled optimization) both in terms of context quality and final answer performance.

Our main contributions are:

- We introduce s3, a modular, RL-based search framework that optimizes for generation quality without touching the generator.
- We define Gain Beyond RAG (GBR), a principled, model-agnostic reward signal that quantifies improvements over standard retrieval.
- We show that s3 outperforms state-of-the-art agentic RAG methods on six general and five medical QA benchmarks, using 70× less training data (see Figure 1).

2 Related Work

2.1 Retrieval-Augmented Generation

Large language models (LLMs) have shown impressive generative capabilities (Touvron et al., 2023; OpenAI, 2023), but their factuality remains bounded (Peng et al., 2023) by their training corpora. Retrieval-Augmented Generation

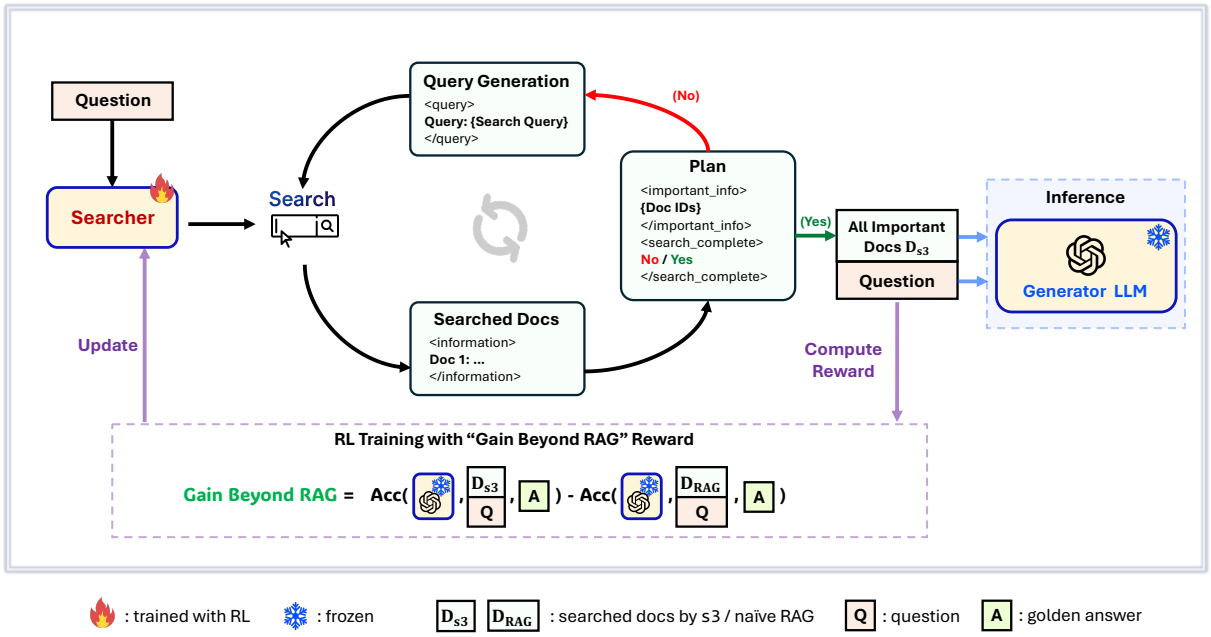

Figure 4: **Overview of the s3 framework.** The searcher iteratively generates queries, retrieves documents, and selects useful documents until completion. The final context D_{s3} is then passed to a frozen generator LLM. The searcher is trained using Gain Beyond RAG (GBR), which quantifies improvement over naïve top-k retrieval from the original question.

(RAG) (Lewis et al., 2020; Gao et al., 2023) augments LLMs by prepending retrieved documents to their input, enabling access to up-to-date or domain-specific information. The effectiveness of this setup, however, depends heavily on the retrieval quality. Early efforts improve retrieval through supervised query rewriting (Nogueira and Cho, 2019; Lin et al., 2023a), where LLMs are fine-tuned to generate better search queries from manually labeled or distilled training data. These methods require significant annotation effort and often optimize for imitation rather than end-task performance. Recent works have introduced Active RAG methods (Yao et al., 2022; Trivedi et al., 2023a; Asai et al., 2023; Lyu et al., 2024), which prompt LLMs to iteratively retrieve and reason in a zero-shot or few-shot manner. While flexible, these methods typically rely on handcrafted prompting patterns and lack direct optimization by interacting with environment.

2.2 RL for Agentic Retrieval and Searcher-Centric Optimization

The emergence of reinforcement learning (RL) for large language models has given rise to agentic retrieval, where models interact with search engines and improve by receiving outcome-based feedback—such as whether the final answer is correct. We refer to this shift as the RL-Zero period, sparked by the insight that even simple rewards like answer correctness can elicit strong reasoning and search behavior (Guo et al., 2025). Within this paradigm, retrieval-centric methods like DeepRetrieval (Jiang et al., 2025) optimize query generation for search metrics (e.g., recall, NDCG), which often fail to reflect answer utility—i.e., whether the retrieved context helps generate a correct answer. Conversely, end-to-end approaches like Search-R1 (Jin et al., 2025) train LLMs to retrieve and generate jointly using exact match rewards, but require full model access and entangle search with answer token alignment.

In contrast, s3 takes a searcher-centric approach that avoids generator fine-tuning. It directly optimizes retrieval quality using a generation-aware reward, enabling lightweight and modular training that is compatible with black-box LLMs.

3 s3: Optimized Search-Select-Serve Flow with Reinforcement Learning

We introduce s3, a lightweight, model-agnostic framework that equips a tunable search agent with structured, multi-turn access to external knowledge. As illustrated in Figure 4, the searcher LLM interacts with a search engine iteratively: it generates queries, retrieves documents, selects a subset of useful evidence, and decides whether to continue searching. A frozen generator LLM then consumes the accumulated evidence to produce a final answer. To ensure a fair reward baseline, s3 begins by retrieving top-k ($k = 3$ in our experiments) documents from the original question, just like naïve

RAG. The searcher is trained using the Gain Beyond RAG (GBR) reward, which measures the improvement in generation accuracy when using its retrieved context versus this baseline. This modular design enables targeted optimization of retrieval quality, decoupled from answer generation.

3.1 Multi-Turn Search-Select Loop

Given a question Q, the system consists of (1) a searcher LLM (policy) $\pi_{\texttt{s3}}$, (2) a search engine $\mathcal{R}$, (3) a frozen generator LLM $\mathcal{G}$.

s3 first retrieves top-k documents using $q_0 = Q$, yielding $\mathcal{D}_0 = \mathcal{R}(Q)$. A subset $\mathcal{D}_0^{\text{sel}} \subseteq \mathcal{D}_0$ is selected to form the initial context. It then performs a sequence of search rounds $t = 1, 2, \ldots, T$, structured as follows:

s3 Loop

1. **Query Generation:** The searcher emits a query q_t in <query>...</query>.
2. **Search:** Documents $\mathcal{D}_t = \mathcal{R}(q_t)$ are retrieved in <information>...</information>
3. **Select:** Useful documents are selected between <important_info>...</important_info>, corresponding to subset $\mathcal{D}_t^{\text{sel}} \subseteq \mathcal{D}_t$.
4. **Stop decision:** The model declares <search_complete>[1/0]</search_complete>.

The loop continues until search_complete is True (1) or the turn limit is reached. The final context is $\mathcal{D}_{\texttt{s3}} = \bigcup_{t=0}^{T} \mathcal{D}_t^{\text{sel}}$, which is passed (served) to the generator to produce the final output:

$$\hat{A} = \mathcal{G}(Q, \mathcal{D}_{\texttt{s3}})$$

Initialization (Begin with Search). Initializing with $q_0 = Q$ ensures the loop begins with the same context as naïve RAG, making the Gain Beyond RAG reward reflect true search improvements.

3.2 Training via Gain Beyond RAG (GBR)

To train $\pi_{\texttt{s3}}$, we frame search as a reinforcement learning problem. The reward signal, *Gain Beyond RAG (GBR)*, quantifies the improvement in generation accuracy over a fixed top-k baseline:

$$\text{GBR}(Q) = \text{Acc}(\mathcal{G}(Q, \mathcal{D}_{\texttt{s3}}), A) - \text{Acc}(\mathcal{G}(Q, \mathcal{D}_{\text{RAG}}), A) \quad (1)$$

where A is the gold-standard answer, and $\mathcal{D}_{\text{RAG}} = \mathcal{R}(Q)$ is the top-k retrieval from the original question. $\text{Acc}(\cdot)$ is a task-specific metric, which we instantiate as *Generation Accuracy* (see §4.1) for RAG performance.

This reward ensures the searcher is incentivized to retrieve documents that meaningfully enhance the generator's output quality, independent of surface-form answer similarity. To improve training efficiency, we precompute the baseline accuracy term $\text{Acc}(\mathcal{G}(Q, \mathcal{D}_{\text{RAG}}), A)$ and restrict training to examples where it equals 0. This effectively filters out questions already solvable by naïve RAG, allowing s3 to focus on harder queries where improved retrieval is essential for generation success.

3.3 Search Policy Optimization

We optimize the search policy $\pi_{\texttt{s3}}$ via reinforcement learning using the Gain Beyond RAG (GBR) reward. Each rollout consists of a complete search trajectory: emitted queries, document selections, and a stop decision. Once the final context $\mathcal{D}_{\texttt{s3}}$ is constructed, the generator $\mathcal{G}$ produces an answer, and the GBR reward is computed. The generator remains frozen; gradients are backpropagated only through the search policy. Our method is agnostic to the specific advantage estimation algorithm. In this work, we use *Proximal Policy Optimization* (PPO) (Schulman et al., 2017) due to its strong empirical stability (Jiang et al., 2025; Jin et al., 2025). The PPO objective is:

$$\mathcal{L}_{\texttt{PPO}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\Big[\sum_{t=1}^{T} \min\Big(r_t(\theta)\,\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\Big)\Big] \quad (2)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$ is the probability ratio between the current and reference policies, $\hat{A}_t$ is the estimated advantage, and ϵ is clipping threshold.

4 Experiments

4.1 Experimental Setups

Evaluation Metric. We measure performance using *Generation Accuracy*, which combines a fast span-matching test (Ma et al., 2021; Lin et al., 2021) with a lightweight LLM-based correctness check (Figure 13). Given a model prediction p and a set of gold answers $\mathcal{A}$, we compute:

$$\text{GenAcc} = \text{span\_check} \vee \text{judge\_check} \quad (3)$$

which can be either 1 or 0, determined by the following evaluation flow:

Methods	Searcher	#Train	Single-Hop NQ†	TriviaQA	PopQA	Multi-Hop HotpotQA†	2wiki	Musique	Avg.
#Test Data			3,610	11,313	14,267	7,405	12,576	2,417	
End-to-End Fine-Tuning									
$SFT_{Qwen2.5\text{-}3B\text{-}Inst}$	-	170k	23.7 (17.5)	41.6 (34.3)	18.1 (14.0)	18.0 (13.7)	22.1 (20.8)	5.1 (2.9)	21.4 (17.2)
$R1_{Qwen2.5\text{-}7B\text{-}Inst}$	-	170k	35.6 (28.8)	60.2 (53.4)	22.4 (20.5)	29.4 (24.0)	30.0 (29.1)	10.7 (7.8)	31.4 (27.3)
Search-R1-3B	(self) 3B	170k	47.0 (27.9)	65.6 (46.2)	46.4 (34.9)	33.5 (22.1)	28.5 (24.4)	6.0 (2.8)	37.8 (26.4)
Search-R1-7B	(self) 7B	170k	56.9 (48.2)	73.8 (64.0)	50.6 (46.8)	54.6 (43.5)	51.6 (38.4)	28.5 (20.6)	52.7 (43.6)
Generator (Qwen2.5-7b-Instruct) Frozen									
Direct Inference	-	0	37.3 (4.4)	55.1 (32.9)	19.9 (8.3)	28.1 (7.6)	36.9 (9.1)	10.6 (1.2)	31.3 (10.6)
CoT	-	0	37.7 (10.3)	60.6 (35.4)	22.2 (11.3)	31.1 (13.4)	31.6 (18.9)	10.6 (4.2)	32.3 (15.6)
RAG_{BM25}	-	0	43.6 (3.8)	69.8 (29.7)	34.6 (12.4)	45.3 (15.1)	38.5 (10.3)	11.5 (1.5)	40.6 (12.1)
RAG_{E5}	-	0	62.1 (5.8)	74.5 (33.8)	54.5 (20.3)	46.6 (13.6)	40.1 (7.8)	13.0 (2.0)	48.5 (13.9)
IRCoT	(self) 7B	0	63.2 (6.2)	75.6 (34.3)	54.5 (19.3)	50.9 (15.4)	48.7 (9.6)	16.4 (2.5)	51.6 (14.5)
IRCoT	14B	0	63.9 (6.3)	75.5 (34.9)	55.5 (20.3)	52.5 (16.0)	47.4 (9.3)	17.2 (2.7)	52.0 (14.9)
Search-R1-3B (Ret)	3B	170k	56.6 (6.6)	68.6 (32.5)	49.4 (18.8)	41.5 (13.6)	33.2 (7.8)	12.1 (1.9)	43.6 (13.5)
Search-R1-7B (Ret)	7B	170k	61.3 (8.1)	73.7 (35.9)	51.9 (20.7)	58.6 (20.0)	50.8 (12.2)	**27.6 (7.1)**	54.0 (17.3)
s3	7B	2.4k	**66.1 (7.2)**	**78.5 (36.8)**	**57.4 (21.9)**	**59.0 (21.8)**	**51.6 (12.4)**	23.9 (6.1)	**56.1 (17.7)**
Generator (Qwen2.5-14b-Instruct) Frozen									
Direct Inference	-	0	38.8 (8.2)	62.7 (39.0)	24.5 (10.8)	30.2 (9.5)	38.6 (7.2)	12.5 (1.8)	34.5 (12.8)
CoT	-	0	40.5 (10.2)	66.2 (41.6)	24.6 (13.6)	32.9 (12.3)	33.2 (13.8)	12.6 (5.2)	35.0 (16.1)
RAG_{BM25}	-	0	54.8 (16.4)	76.7 (44.8)	41.5 (22.7)	50.4 (18.3)	49.9 (6.4)	17.7 (3.1)	48.5 (18.6)
RAG_{E5}	-	0	62.4 (18.7)	77.4 (50.7)	55.1 (34.0)	47.4 (20.9)	44.9 (10.1)	16.1 (3.3)	50.6 (23.0)
IRCoT	7B	0	63.0 (18.8)	77.7 (50.1)	56.3 (33.5)	50.7 (22.7)	53.2 (12.4)	17.5 (4.1)	53.1 (23.6)
IRCoT	(self) 14B	0	63.9 (19.2)	78.2 (51.7)	56.1 (33.8)	51.6 (23.7)	54.0 (12.0)	19.1 (5.2)	53.8 (24.3)
Search-R1-3B (Ret)	3B	170k	59.2 (16.5)	75.6 (47.4)	52.3 (30.3)	45.5 (18.3)	44.0 (8.3)	16.0 (2.9)	48.8 (20.6)
Search-R1-7B (Ret)	7B	170k	63.8 (18.0)	76.3 (49.5)	54.6 (33.3)	56.7 (25.3)	56.7 (11.0)	**30.2 (9.1)**	56.4 (24.4)
s3	7B	2.4k	**67.2 (18.3)**	**79.5 (48.9)**	**57.8 (35.7)**	**57.1 (23.3)**	**57.1 (11.6)**	26.7 (7.8)	**57.6 (24.3)**
Generator (Claude-3-Haiku) Frozen									
Direct Inference	-	0	48.1 (25.7)	76.5 (64.8)	35.7 (30.9)	35.5 (24.2)	28.9 (24.0)	8.8 (4.3)	38.9 (29.0)
CoT	-	0	61.5 (2.9)	81.0 (30.0)	43.2 (9.1)	48.8 (8.8)	46.2 (6.8)	21.2 (2.3)	50.3 (10.0)
RAG_{BM25}	-	0	50.5 (3.8)	75.5 (28.4)	35.9 (8.0)	50.2 (11.4)	40.7 (8.1)	11.8 (0.8)	44.1 (10.1)
$DeepRetrieval_{BM25}$	3B	70k	64.4 (3.7)	80.2 (23.2)	45.5 (8.2)	54.5 (10.2)	47.1 (8.0)	22.2 (1.7)	52.3 (8.1)
RAG_{E5}	-	0	66.5 (4.3)	80.7 (28.9)	55.7 (8.9)	50.7 (11.5)	39.2 (7.8)	14.0 (1.2)	51.1 (10.4)
IRCoT	7B	0	68.0 (4.2)	81.7 (29.3)	55.5 (8.9)	54.8 (11.7)	46.5 (8.1)	17.4 (1.6)	54.0 (10.6)
IRCoT	14B	0	68.3 (4.2)	81.6 (29.5)	56.1 (8.6)	55.5 (11.9)	47.7 (8.4)	18.9 (1.7)	54.7 (10.7)
Search-o1	14B	0	67.3 (4.7)	81.2 (29.8)	50.2 (9.3)	58.1 (12.6)	48.8 (8.4)	14.2 (1.2)	53.3 (11.0)
Search-R1-3B (Ret)	3B	170k	60.7 (3.3)	74.5 (24.8)	50.1 (6.9)	45.7 (10.0)	33.1 (7.0)	12.7 (1.3)	46.1 (8.9)
Search-R1-7B (Ret)	7B	170k	68.1 (4.1)	80.9 (25.9)	55.7 (7.0)	62.0 (11.2)	51.0 (7.2)	**29.3 (3.2)**	57.8 (9.8)
s3	7B	2.4k	**70.5 (3.2)**	**84.0 (24.6)**	**57.7 (5.9)**	**62.4 (11.1)**	**52.4 (8.3)**	26.2 (7.9)	**58.9 (10.2)**

Table 1: Performance comparison on **general-domain QA datasets**. Datasets marked with † are the source of training data used by Search-R1 and s3. We show generation accuracy (§4.1) as the main results, and exact match scores in brackets. We use E5-base-v2 as the retriever and Wikipedia-2018 as the corpus. "Searcher" shows the number of parameters of the searcher model. "#Train" shows the amount of training data used to train the searcher. $DeepRetrieval_{BM25}$ is trained on NQ, Search-R1 and s3 are trained on NQ+HotpotQA with different training size (170k vs 2.4k). Results are averaged by three runs.

Evaluation Flow of Generation Accuracy

Input: Prediction p, Gold Answers $\mathcal{A}$

Step 1: Normalize p and $\mathcal{A}$ (lowercase, remove punctuation and articles).
Step 2: `span_check` → If any $a \in \mathcal{A}$ is a token span in p, return `GenAcc` = 1.
Step 3: `judge_check` → Prompt LLM: *"Does p contain any of $\mathcal{A}$?"*
Step 4: Return `GenAcc` = 1 if LLM says yes; else 0.

Why Exact Match Falls Short - An Example

Golden answer: `"Barack Obama"`
LLM response: `"The 44th President of the United States was Barack Obama."`
Exact match: 0 (response ≠ golden)
Generation Accuracy: 1 (span_check succeeds)

We choose this metric because it better captures semantic correctness and aligns more closely with human judgment than traditional exact match (see Appendix B for supporting evidence).

Datasets. Following prior study (Jin et al., 2025), we construct the training set by combining samples from Natural Questions (NQ) and HotpotQA. Since `span_check` may incorrectly accept answers for questions with semantic negation (e.g., treating "not true" as matching "true"), we remove all yes/no and true/false questions from the training set to ensure reliable reward signals. To focus training on harder examples, we filter out samples where the generator LLM (Qwen2.5-14B-Instruct) already produces a correct answer using naïve RAG retrieval. This reduces the dataset size from 169,615 to 70,286. As later shown in Figure 5, s3 rapidly converges within ∼15 training steps with random

Methods	Searcher	#Train	Medical RAG-QA Datasets (MIRAGE)					Avg.
			MedQA-US	MedMCQA	PubMedQA	BioASQ-Y/N	MMLU-Med	
#Test Data			1,273	4,183	500	618	1,089	
w/o retrieval	-	0	61.7$_{(45.8)}$	55.8$_{(29.3)}$	55.6$_{(0.0)}$	76.9$_{(0.0)}$	76.4$_{(35.8)}$	65.3$_{(22.2)}$
Corpus: Wikipedia 2018 (Karpukhin et al., 2020)								
RAG$_{BM25}$	-	0	61.6$_{(48.2)}$	57.5$_{(45.2)}$	52.8$_{(4.6)}$	73.6$_{(6.3)}$	77.6$_{(61.9)}$	64.6$_{(33.2)}$
DeepRetrieval$_{BM25}$	3B	70k	62.5$_{(45.4)}$	61.3$_{(44.8)}$	56.2$_{(8.2)}$	**77.3$_{(9.2)}$**	**79.2$_{(57.9)}$**	67.3$_{(33.1)}$
RAG$_{E5}$	-	0	61.5$_{(46.7)}$	58.0$_{(44.7)}$	54.6$_{(3.8)}$	73.3$_{(5.3)}$	77.9$_{(62.2)}$	65.1$_{(32.5)}$
IRCoT	7B	0	62.8$_{(45.1)}$	60.5$_{(45.4)}$	54.2$_{(8.6)}$	73.0$_{(13.8)}$	78.7$_{(58.2)}$	65.8$_{(34.2)}$
IRCoT	14B	0	61.7$_{(48.9)}$	60.3$_{(46.7)}$	53.0$_{(7.6)}$	75.2$_{(11.8)}$	77.2$_{(61.9)}$	65.5$_{(35.4)}$
Search-o1	14B	0	64.5$_{(55.4)}$	59.6$_{(47.7)}$	52.2$_{(1.8)}$	74.9$_{(0.2)}$	77.7$_{(63.9)}$	65.8$_{(33.8)}$
Search-R1-3B (Ret)	3B	170k	58.8$_{(47.2)}$	53.7$_{(41.4)}$	53.8$_{(4.4)}$	63.6$_{(4.4)}$	68.4$_{(55.4)}$	59.7$_{(30.6)}$
Search-R1-7B (Ret)	7B	170k	62.6$_{(45.7)}$	59.2$_{(42.8)}$	55.4$_{(5.2)}$	71.2$_{(6.5)}$	69.3$_{(53.3)}$	63.5$_{(30.7)}$
s3	7B	2.4k	**65.7$_{(47.1)}$**	**61.5$_{(44.3)}$**	**56.6$_{(5.2)}$**	**77.3$_{(7.1)}$**	76.0$_{(56.3)}$	**68.3$_{(32.0)}$**
Corpus: Wikipedia+PubMed+Textbook (Xiong et al., 2024)								
RAG$_{BM25}$	-	0	65.4$_{(43.1)}$	59.9$_{(44.4)}$	79.4$_{(10.8)}$	88.4$_{(6.5)}$	79.6$_{(57.1)}$	74.5$_{(32.4)}$
DeepRetrieval$_{BM25}$	3B	70k	65.0$_{(35.1)}$	65.1$_{(44.2)}$	78.6$_{(16.2)}$	89.5$_{(7.4)}$	79.3$_{(49.1)}$	75.8$_{(30.4)}$
RAG$_{E5}$	-	0	64.1$_{(43.4)}$	60.1$_{(45.0)}$	79.4$_{(10.8)}$	89.8$_{(5.0)}$	78.8$_{(58.8)}$	74.6$_{(32.6)}$
IRCoT	7B	0	63.9$_{(38.6)}$	62.7$_{(45.3)}$	75.4$_{(13.0)}$	87.2$_{(5.8)}$	**79.7$_{(54.9)}$**	73.8$_{(31.5)}$
IRCoT	14B	0	62.7$_{(43.8)}$	62.3$_{(46.6)}$	74.0$_{(10.8)}$	87.9$_{(5.3)}$	79.6$_{(59.0)}$	73.3$_{(33.1)}$
Search-o1	14B	0	65.0$_{(50.1)}$	61.1$_{(47.6)}$	74.2$_{(12.0)}$	89.3$_{(5.3)}$	78.1$_{(59.5)}$	73.5$_{(34.1)}$
Search-R1-3B (Ret)	3B	170k	57.5$_{(45.5)}$	54.8$_{(40.7)}$	71.4$_{(7.8)}$	73.3$_{(3.6)}$	62.0$_{(47.6)}$	63.8$_{(29.0)}$
Search-R1-7B (Ret)	7B	170k	62.1$_{(43.2)}$	61.9$_{(44.2)}$	78.6$_{(8.0)}$	86.3$_{(5.3)}$	69.9$_{(48.9)}$	71.8$_{(29.9)}$
s3	7B	2.4k	**65.7$_{(45.7)}$**	**65.3$_{(45.4)}$**	**81.5$_{(13.6)}$**	**92.1$_{(6.5)}$**	78.3$_{(56.2)}$	**76.6$_{(33.5)}$**

Table 2: **Performance on medical-domain QA datasets** (Xiong et al., 2024), using *Claude-3-Haiku* as the generator. We report *judge_check* as the primary metric (see §4.1), with exact match in brackets. Retrieval is performed with E5-base-v2 under two corpus settings: **Wikipedia-2018** and **Wikipedia+PubMed+Textbook**. s3 achieves the highest overall accuracy among all retrieval-augmented methods in both settings. None of the methods is trained on medical data: DeepRetrieval$_{BM25}$ is trained on 70k NQ, Search-R1 on 170k NQ+HotpotQA, and s3 on 2.4k NQ+HotpotQA. Results are averaged by three runs.

seed 42 (for data shuffling). For evaluation, we use the checkpoints at step 20. Given a batch size of 120, this corresponds to approximately 2.4k training examples, highlighting the data efficiency of our method. We evaluate on six general-domain QA benchmarks: NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2022), HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), and Musique (Trivedi et al., 2022), as well as MIRAGE (Xiong et al., 2024), a suite of five medical-domain QA datasets.

Baselines. We compare s3 against a diverse set of RAG systems: **(1) End-to-End Fine-tuning.** Fully fine-tuned models that jointly retrieve and generate using outcome-based RL or supervision: Search-R1 (3B/7B), SFT (3B), and R1 (7B) where 3B/7B for SFT and R1 are based on Qwen2.5-3B/7B-Instruct. **(2) Static Retrieval+Frozen Generator.** Methods that retrieve documents using a fixed or scripted strategy, then pass them to a frozen generator: RAG-BM25, RAG-E5: retrieval via BM25 or E5-base (Wang et al., 2022). DeepRetrieval-BM25 (3B): RL-trained searcher optimizing recall, paired with BM25. **(3) Active Retrieval+Frozen Generator.** A diagnostic setting where we extract the documents retrieved during a model's reasoning trajectory and feed them to a frozen generator: Search-R1-3B/7B (Ret), IRCoT (7B/14B), and Search-o1 (14B) all fall under this category, differing only in whether retrieval is learned (Search-R1) or prompted (IRCoT (Trivedi et al., 2023b), Search-o1 (Li et al., 2025)). **(4) s3.** Our s3 trains only the searcher using GBR and forwards selected documents to a frozen generator, with no fine-tuning. We place more details in Appendix A.1.

Models for Training and Evaluation. Throughout all the training processes, we use Qwen2.5-7B-Instruct (Yang et al., 2024) as the base searcher LLM to train, and use Qwen2.5-14B-Instruct[1] as the frozen generator for both answer generation and judge_check for reward computation. For evaluation, we use Claude-3-Haiku as the LLM for judge_check to ensure high evaluation quality. We test three frozen generators: Qwen2.5-7b/14b-Instruct and Claude-3-Haiku. Both training and evaluation are conducted on five NVIDIA A100 80GB PCIe GPUs. RAGEN (Wang et al., 2025) and VERL (Sheng et al., 2024) are used as the base architecture for multi-turn RL training. We place more details in Appendix A.2.

5 Results

We evaluate s3 across six general-domain and five medical-domain QA benchmarks, with frozen generators ranging from Qwen2.5-7B/14B to Claude-

[1] In this paper, we use "GPTQ-Int4" version of "Qwen2.5-14B-Instruct" for its high efficiency. We deploy frozen LLMs using vLLM (Kwon et al., 2023) for fast inference.

#Retrieval → #Select	#Turns	#MaxContexts	Single-Hop			Multi-Hop			Avg.
			NQ†	TriviaQA	PopQA	HotpotQA†	2wiki	Musique	
8 → 3	3	9	70.5(3.2)	84.0(24.6)	57.7(5.9)	62.4(11.1)	52.4(8.3)	26.2(7.9)	58.9(10.2)
5 → 3	3	9	69.6(3.5)	83.4(24.3)	57.4(5.8)	62.0(11.9)	53.8(7.8)	24.5(2.3)	58.5(9.3)
5 → 3	4	12	70.0(3.5)	83.8(24.8)	57.7(5.8)	62.5(12.3)	54.7(8.0)	25.7(3.2)	59.1(9.6)
3 → 3	4	12	68.9(3.7)	82.0(24.9)	56.4(6.1)	62.0(11.9)	51.7(7.7)	24.7(2.8)	57.7(9.5)
3 → 3	3	9	69.4(3.5)	82.3(24.4)	57.0(5.7)	61.8(11.7)	51.5(8.2)	25.1(2.3)	57.9(9.3)

Table 3: Study of the numbers of retrieved documents (#Retrieval) and turns (#Turns). Maximum selection is set to 3 across all settings. We use the frozen Claude-3-Haiku as the generator LLM for this study.

3-Haiku. We report generation accuracy as the primary metric and provide detailed comparisons across baselines, reward functions, and training efficiency.

General Domain RAG Performance. Table 1 summarizes results across general QA datasets. s3 achieves the highest average accuracy of 58.9%, outperforming all static, zero-shot, and end-to-end tuned baselines. This is particularly notable given its extreme data efficiency—trained on just 2.4k examples, compared to 70k for DeepRetrieval and 170k for Search-R1.

Takeaway #1: Searcher-Only is better than End-to-End Optimization for RAG

s3 consistently outperforms Search-R1 on search quality, revealing that most of the performance gain in RAG stems from improving the search capability instead of aligning generation outputs.

Compared to IRCoT-14B, which conducts zero-shot retrieval with 2× the parameter count, s3 gains +4.6 points on average. Relative to Search-R1-7B (Ret), which uses the same backbone, s3 improves by +1.5 points while avoiding any generator tuning. These gains are consistent across both single-hop datasets (e.g., 70.0% on NQ) and multi-hop datasets (e.g., 62.4% on HotpotQA), showing that learned search behavior transfers across reasoning complexity.

Medical Domain QA Performance. Table 2 reports performance on the MIRAGE suite (Xiong et al., 2024) under both corpus settings. s3 achieves the highest average accuracy (76.6%) when using the combined Wikipedia+PubMed+Textbook corpus, surpassing all retrieval-augmented baselines.

Interestingly, while Search-R1 shows competitive scores on Wikipedia-only corpora, its performance deteriorates on richer corpora, indicating overfitting to shallow heuristics or memorized formats. In contrast, s3 and DeepRetrieval remain robust, with s3 achieving 81.5% on PubMedQA and outperforming IRCoT across four of five tasks.

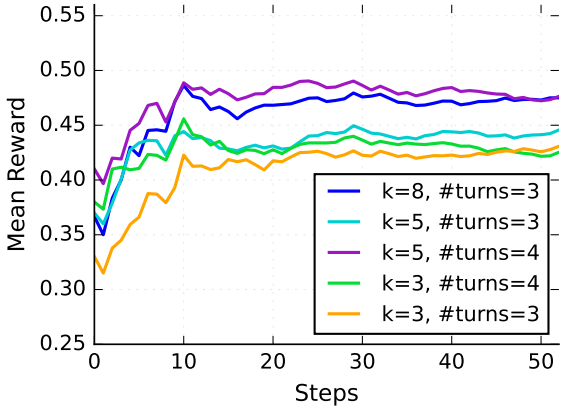

Figure 5: Reward Curves for top $k = \{3, 5, 8\}$ and #turns $= \{3, 4\}$. The maximum selection is kept as 3.

Takeaway #2: Searcher-Only Training enables Domain Transfer

s3's zero-shot success on medical QA, despite training only on general QA, suggests that reinforcement-learned search skills generalize more reliably than generation-tuned approaches.

Retrieval Behavior and Search Dynamics We analyze the effect of retrieval parameters (#retrieved documents and #turns) in Table 3 and reward progression in Figure 5. s3 reaches peak performance with (k=8, turns=3), and adding more turns or broader retrieval brings limited improvement. This indicates that the policy rapidly learns to emit focused and early queries, capturing most useful content without unnecessary expansion.

Training Efficiency Table 4 shows that it takes 20 PPO steps (2.4k examples) to train s3, while Search-R1 requires 2,100 steps (170k examples). Even accounting for the higher per-step cost due to LLM-based reward computation, the total wall-clock time is reduced by ∼33×. Moreover, s3 avoids retriever pretraining and operates with a smaller 7B policy model, making it a practical method for low-resource RL training. s3 achieves

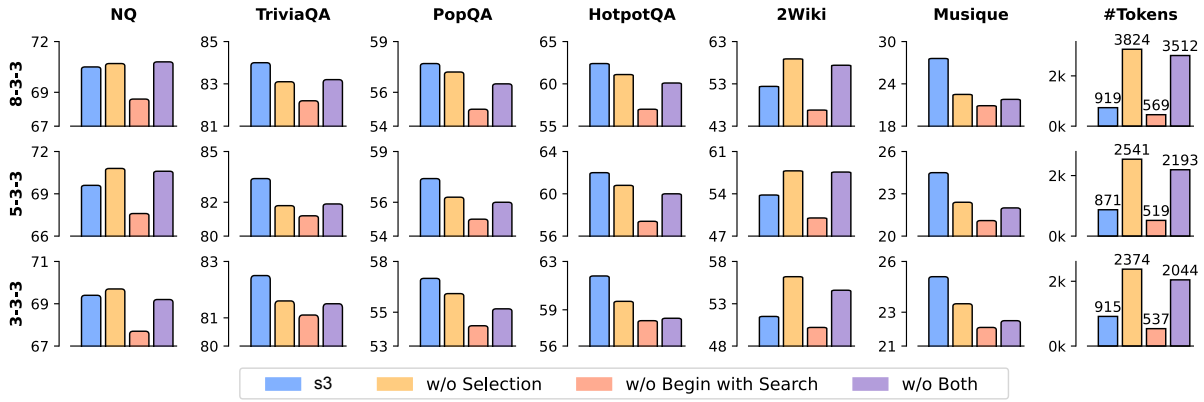

Figure 6: **Ablation study on s3 components.** Each row corresponds to a different configuration of Retrieval:Selection:Turns = 8:3:3, 5:3:3, and 3:3:3. The first six columns report generation accuracy. **"Begin with Search"** refers to initializing the first query with the original question. **"Document Selection"** refers to the selection step within the s3 loop (Step 3). We observe that removing Begin with Search leads to a significant drop in performance. While removing Document Selection sometimes yields better performance, the full s3 system still performs competitively—and most importantly, drastically reduces input token usage (2.6× ∼ 4.2× less tokens), improving overall efficiency.

	Time/Step	Training Steps	Total
Search-R1	1.8m	∼2,100	3,780m
DeepRetrieval$_{BM25}$	1.3m	∼1,600	2,080m
s3	5.7m	∼20	**114m**

Table 4: Comparison of Training Efficiency (tested with batch size=120 on five NVIDIA A100 GPUs). Note: s3 is slower stepwise since we need to conduct generation and evaluation by a frozen LLM for reward computation during training.

	GenAcc	LLMJudge	Span	EM
General QA	58.9	59.6	57.1	50.5
Medical QA	76.6	77.3	74.3	70.3

Table 5: Comparison of RAG performance under different reward functions. *LLMJudge* (judge_check) yields the highest scores but is computationally expensive. *GenAcc* offers a good balance of accuracy and efficiency, while *Span* (span_check) and *EM* underperform due to limited semantic coverage.

state-of-the-art performance with orders of magnitude less data and compute, suggesting a more sustainable path for RAG optimization.

Reward Function Comparison Table 5 compares different reward signals used for computing GBR. LLMJudge provides slightly higher final scores, but is too costly for scalable training. In contrast, GenAcc offers strong performance while remaining efficient and aligning better with human evaluation than EM or span-based heuristics. Appendix B shows that GenAcc matches human judgment on 96.4% of samples, while Exact Match used by Search-R1 captures only 15.8%.

Takeaway #3: Reward Choice directly shapes Search Quality

Using semantically or human preference aligned metrics like our GenAcc (§4.1) encourages the search policy to retrieve substantively helpful documents, rather than optimizing for brittle string overlap.

Effects of Selection and "Begin with Search". We investigate the role of two components in the s3 loop: document selection and initialization with the original question (Begin with Search"). As shown in Figure 6, removing the selection step degrades performance on four out of six datasets. This is expected, as passing all retrieved documents to the generator increases token length, up to 4× with $k = 8$, and introduces more noise. Still, performance improves slightly on NQ and 2Wiki, likely because broader context benefits multi-hop reasoning or compensates for overly aggressive pruning. Disabling "Begin with Search" consistently causes a significant drop, underscoring the importance of seeding the search process with a strong initial query. Interestingly, when both selection and initialization are removed, performance recovers slightly compared to removing only initialization. This suggests that selection and initialization interact conditionally—selection may amplify the downsides of poor initialization by prematurely filtering out useful context.

6 Conclusion

We present s3, a framework that trains a search-only agent using the Gain Beyond RAG reward. By decoupling search from generation and optimizing only the retriever, s3 outperforms strong baselines with just 2.4k examples. Our results show that targeted search policy learning yields substantial gains in both efficiency and generalization, offering a scalable path for improving RAG systems.

7 Limitations

While s3 demonstrates strong empirical performance with remarkable data efficiency, several limitations warrant discussion.

Dependency on Frozen Generators. Our framework assumes the availability of a capable frozen generator LLM. Although this enables model-agnostic training, it implicitly relies on the generator's ability to make use of improved context. For lower-capacity or instruction-weak generators, the gains from better retrieval may not fully translate into better outputs.

Reward Estimation Bottleneck. The use of generation-based rewards such as GenAcc necessitates LLM inference during training to compute reward signals. This introduces computational overhead compared to token-level or retrieval-only objectives, limiting scalability. Although we show that s3 achieves high performance with minimal steps, online reward computation remains more costly than offline retrieval optimization.

Broader Impacts. On the positive side, s3 reduces the data and compute burden for training effective retrieval agents, making RAG systems more accessible to low-resource communities. It may also benefit domains such as healthcare or scientific QA where labeled data is scarce. However, like all retrieval-augmented systems, s3 inherits the biases of both its searcher and generator. If deployed without careful curation of source corpora, it may propagate misinformation or reflect existing societal biases. We encourage practitioners to audit both retrieval sources and downstream outputs when applying this framework in sensitive domains.

Overall, while s3 advances the state of search-agent training, further work is needed to address these limitations and ensure safe, robust deployment in real-world settings.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Lu Dai, Yijie Xu, Jinhui Ye, Hao Liu, and Hui Xiong. 2025. Seper: Measure retrieval utility through the lens of semantic perplexity reduction. In *The Thirteenth International Conference on Learning Representations*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. 2025. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. *arXiv preprint arXiv:2503.00223*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arık, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. *arXiv preprint arXiv:1909.06146*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. 2023. Bioasq-qa: A manually curated corpus for biomedical question answering. *Scientific Data*, 10(1):170.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. 2022. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356–2362.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, and 1 others. 2023a. Ra-dit: Retrieval-augmented dual instruction tuning. *arXiv preprint arXiv:2310.01352*.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, and 1 others. 2023b. Ra-dit: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Yuanjie Lyu, Zihan Niu, Zheyong Xie, Chao Zhang, Tong Xu, Yang Wang, and Enhong Chen. 2024. Retrieve-plan-generation: An iterative planning and answering framework for knowledge-intensive llm generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4683–4702.

Xueguang Ma, Kai Sun, Ronak Pradeep, and Jimmy Lin. 2021. A replication study of dense passage retriever. *arXiv preprint arXiv:2104.05740*.

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

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, and 1 others. 2018. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, pages 561–577.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023b. 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)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, and 1 others. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16:1–28.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, and 1 others. 2025. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6233–6251.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Contents of Appendix

A Implementation Details

For static retrieval baselines running on MIRAGE, we use the question itself instead of question+options to retrieve.

A.1 Baselines Details

IRCoT (7B and 14B). IRCoT[2] (Trivedi et al., 2023b) is a prompting-based method that alternates between chain-of-thought reasoning and retrieval. It requires no fine-tuning: the model is instructed via prompt to iteratively reason about a question and issue retrieval queries, integrating newly retrieved evidence into its reasoning process. We apply IRCoT using both Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct.

DeepRetrieval-BM25-3B (Jiang et al., 2025). This baseline employs a 3B-parameter language model trained with reinforcement learning on retrieval metrics such as recall and NDCG. It learns to generate search queries that maximize the retrieval of relevant documents using a BM25 search engine. Training is conducted on 70k QA examples in NQ dataset with answer span reward (evidence-seeking task in (Jiang et al., 2025)), focusing exclusively on improving retrieval performance, not generation. We use its publicly released checkpoint[3].

Search-R1-3B and Search-R1-7B (Jin et al., 2025). These baselines[4] use 3B and 7B parameter models, respectively, and are trained end-to-end to jointly retrieve and generate answers. Reinforcement learning is applied on 170k training examples, using an exact match (EM) reward to guide both retrieval query formulation and answer generation. The model directly integrates search results into its reasoning steps within a single retrieval round.

Search-o1. Search-o1 (Li et al., 2025) is an inference-time retrieval controller designed to enhance long-form reasoning in o1-style models such as QwQ and OpenAI's o1-preview. It is not trained with reinforcement learning or fine-tuned at all. Instead, Search-o1 leverages frozen LLMs and augments them with retrieval by prompting the model to emit search queries mid-reasoning, enclosed in special tokens (e.g., `<|begin_search_query|>`...). Retrieved documents are then post-processed using a Reason-in-Documents module before being injected back into the reasoning flow.

RAG-BM25 and RAG-E5 (Lewis et al., 2020). These are naive retrieval-augmented generation baselines with no model training. RAG-BM25 uses top-k documents retrieved from a BM25 index, while RAG-E5 retrieves passages using dense retrieval based on E5 embeddings. In both settings, the retrieved documents are prepended to the input prompt and fed into a frozen generator LLM. We set $k = 3$, following prior study (Lin et al., 2023b; Jin et al., 2025).

SFT and R1. On general-domain RAG datasets, we train an SFT model with Qwen2.5-3B-Instruct using the same dataset as Search-R1's 170k NQ+HotpotQA with TRL (von Werra et al., 2020) framework. R1 is the "no search" version of Search-R1 (Jin et al., 2025), replicating Deepseek-R1-Zero (Guo et al., 2025) with a small LLM. We use its publicly released checkpoint[5].

CoT (Wei et al., 2022) and Direct Inference. CoT (Chain-of-Thought) prompting instructs the LLM to generate intermediate reasoning steps before producing an answer, without any external retrieval. Direct Inference simply feeds the raw question into the LLM. Neither baseline involves any form of training or finetuning.

To ensure a fair comparison, we set the maximum number of turns to 4 and limit the context to 3 documents per turn for all multi-turn baselines (IRCoT, Search-R1, and Search-o1) and `s3`, aligning with prior study (Jin et al., 2025).

A.2 Setup Details

Hardware. All training and evaluation processes are run on five NVIDIA A100 80GB PCIe on a system with an AMD EPYC 7513 32-Core Processor and 1.0 TB of RAM.

[2] https://github.com/StonyBrookNLP/ircot

[3] https://huggingface.co/DeepRetrieval/DeepRetrieval-NQ-BM25-3B

[4] https://huggingface.co/PeterJinGo/SearchR1-nq_hotpotqa_train-qwen2.5-3b-em-ppo, https://huggingface.co/PeterJinGo/SearchR1-nq_hotpotqa_train-qwen2.5-7b-em-ppo

[5] https://huggingface.co/PeterJinGo/R1-nq_hotpotqa_train-qwen2.5-7b-em-ppo-v0.2

Software. We built `s3` using Python 3.9, leveraging the VERL framework (Sheng et al., 2024)[6] (v0.1) as the backbone for reinforcement learning with language models, and RAGEN (Wang et al., 2025)[7] as the underlying multi-turn RL architecture. Our implementation uses vLLM (v0.8.5) (Kwon et al., 2023) for fast LLM inference and evaluation, PyTorch (v2.4.0) with CUDA 12.1 for deep learning, and Ray (Moritz et al., 2018) for distributed training and serving. To improve performance, we integrate Flash Attention 2 (Dao, 2023) for efficient attention computation, PySerini (v0.22.1) (Lin et al., 2021) for retrieval and evaluation, and FAISS-GPU (v1.7.2) (Douze et al., 2024) for high-speed dense retrieval.

Model parameters. We fine-tune Qwen2.5-7B-Instruct using Proximal Policy Optimization (PPO) via VERL. Training is conducted with a total batch size of 120, using micro-batches of size 15 for the actor and 10 for the critic, and a rollout temperature of 0.6. The actor and critic learning rates are set to 1×10^{-6} and 1×10^{-5}, respectively, with no warm-up for the actor and a 1% warm-up ratio for the critic. Both models use gradient checkpointing and parameter offloading to reduce memory overhead. Following prior work (Jin et al., 2025), we adopt XFORMERS (Lefaudeux et al., 2022) as the attention backend in vLLM and enable state masking to prevent incorrect supervision signals. KL regularization is applied with a coefficient of 0.001. For answer generation and LLM-based `judge_check` during training, we run Qwen2.5-14B-Instruct-GPTQ-Int4[8] on a dedicated A100 80GB GPU with vLLM. The retriever (E5-base) is deployed alongside PySerini on the same five GPUs used for PPO training. The context window is set to 8,000 tokens, with a maximum of 1,400 tokens allocated to the top-k retrieved documents per turn.

A.3 Datasets & Corpora

Datasets. We evaluate on six general-domain QA datasets and five medical-domain QA datasets.

General-domain datasets include Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2022), HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), and Musique (Trivedi et al., 2022). [9]

For ***medical-domain***, we adopt the MIRAGE benchmark (Xiong et al., 2024), which includes five datasets: MedQA-US (Jin et al., 2021), MedMCQA (Pal et al., 2022), PubMedQA* (Jin et al., 2019), BioASQ-Y/N (Tsatsaronis et al., 2015; Krithara et al., 2023), and MMLU-Med (Hendrycks et al., 2020).[10]

Corpora. ***For general-domain QA***, we follow prior work (Jin et al., 2025) and use the Wikipedia 2018 dump (Karpukhin et al., 2020) as the sole knowledge source.[11] ***For medical-domain QA***, we evaluate under two corpus settings: (1) the Wikipedia 2018 dump (Karpukhin et al., 2020) alone, and (2) a composite biomedical corpus introduced by (Xiong et al., 2024), which combines Wikipedia, PubMed, and textbook documents to provide broader domain coverage.[12]

Use of Artifacts. All datasets and models are used strictly within research contexts, consistent with their intended use and licensing. Our derived artifacts (e.g., retrieved documents, trained models) are likewise restricted to non-commercial academic use.

A.4 Generation Accuracy Computation

To evaluate the effectiveness of retrieval strategies in improving answer generation, we adopt a composite metric called **Generation Accuracy (GenAcc)**, which is designed to better reflect semantic correctness than surface-form exact match.

Overview. Given a model prediction p and a set of gold answers $\mathcal{A}$, GenAcc is defined in Eq. 3. This metric returns 1 if either a string-normalized token span of any $a \in \mathcal{A}$ is found within p, or if a frozen LLM judge deems the answer semantically correct. It returns 0 otherwise.

1. Span-Based Matching. We first apply a deterministic span check using normalized string comparison. Specifically, we:

[6] https://github.com/volcengine/verl

[7] https://github.com/RAGEN-AI/RAGEN

[8] https://huggingface.co/Qwen/Qwen2.5-14B-Instruct-GPTQ-Int4

[9] All the general-domain QA datasets are available at https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets.

[10] All the medical-domain QA datasets are available at https://github.com/Teddy-XiongGZ/MIRAGE/blob/main/benchmark.json.

[11] Wikipedia 2018 dump is available at https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets (retrieval-corpus folder)

[12] All the corpora for medical RAG are available at https://huggingface.co/MedRAG.

- Convert both prediction and gold answers to lowercase.
- Remove punctuation and articles (*a, an, the*).
- Apply whitespace normalization.

We then use a tokenizer to compare whether any token span in the prediction matches any normalized gold answer. If a match is found, the score is 1.

Examples:

- *Success Case:*
  Prediction: `"The 44th President of the United States was Barack Obama."`
  Gold Answer: `"Barack Obama"`
  Result: Span match succeeds because the normalized gold answer is a token span in the prediction.
- *Failure Case (Negation):*
  Prediction: `"That statement is not true."`
  Gold Answer: `"true"`
  Result: Span match incorrectly succeeds due to token overlap, despite the semantic meaning being opposite. We exclude such yes/no cases from training to avoid this issue.
- *Failure Case (Paraphrase):*
  Prediction: `"He led the civil rights movement in the 1960s."`
  Gold Answer: `"Martin Luther King Jr."`
  Result: Span match fails because the gold answer does not appear verbatim in the response, even though the answer is implied.

2. LLM-Based Semantic Judging. If the span check fails (0), we invoke a lightweight correctness check using a frozen LLM (e.g., `Qwen2.5-14B-Instruct-GPTQ-Int4` for training or `Claude-3-Haiku` for evaluation). We prompt the model with:

```
Please check if any of the golden answers
is contained in the following response:
{p}
Golden answers: {str(𝒜)}
Directly answer with 'yes' or 'no'.
```

If the LLM outputs `yes`, we consider the prediction correct and set the score to 1.

Examples:

- *Success Case (Numerical Format):*
  Prediction: `"The answer is twenty-five."`
  Gold Answer: `"25"`
  Result: Span match fails due to different formats, but the LLM outputs `yes` based on numerical equivalence.
- *Success Case (Units and Symbols):*
  Prediction: `"It weighs 3 kilograms."`
  Gold Answer: `"3 kg"`
  Result: Span match fails due to token mismatch, but the LLM recognizes them as equivalent and answers `yes`.
- *Failure Case (Incorrect Entity):*
  Prediction: `"The capital of France is Marseille."`
  Gold Answer: `"Paris"`
  Result: Span match fails, and the LLM also outputs `no`, indicating semantic disagreement.

Motivation. This design avoids brittle behavior from exact match metrics and aligns more closely with human judgments. For instance, if the gold answer is *"Einstein"* and the model prediction is *"Albert Einstein was the scientist who developed the theory of relativity"*, our metric returns 1, while exact match fails due to surface mismatch. Empirically, GenAcc matches human labels on 96.4% of samples (see Appendix B), whereas EM only achieves 15.8%.

Implementation. The full reward computing pipeline is implemented through the following components:

- `span_check`: This function (1) normalizes both prediction and gold answers by applying case-folding, punctuation and article removal, and whitespace normalization; and (2) performs token-level span matching using a tokenizer. This step follows the evaluation strategy introduced in prior work (Ma et al., 2021) and leverages the `has_answer` utility from PySerini[13].
- `judge_check`: If the span check fails, this fallback invokes a frozen LLM to assess whether the prediction semantically entails any gold answer. The LLM is prompted to respond with a binary judgment ("yes" or "no").
- `check_answer_correct`: This function coordinates the evaluation process. It first applies `span_check`; if that fails, it falls back to `judge_check` for semantic validation. **Note:** For the medical RAG benchmark (MIRAGE (Xiong et al., 2024)) evaluation, we exclusively use `judge_check`, as most questions are multiple-choice and `span_check` can incorrectly accept wrong answers due to its strict matching criteria.

[13] https://github.com/castorini/pyserini/blob/master/pyserini/eval/evaluate_dpr_retrieval.py

This hybrid strategy combines the efficiency of lexical matching with the robustness of LLM-based semantic evaluation, ensuring reliable and scalable answer correctness assessment.

A.5 Document Extraction Logic

We extract document titles and texts from information blocks using a structured approach that prioritizes important documents. Our extraction algorithm processes text with the following format:

```
<information>
Doc 1 (Title: "Document Title 1") ...
Doc 2 (Title: "Document Title 2") ...
</information>
<important_info>
[1, 3]
</important_info>
```

The algorithm follows these key rules:

- `<important_info>` tags apply only to the most recent `<information>` block
- If no `<important_info>` tag exists for a `<information>` block, all documents from that block are included
- Documents are deduplicated based on content

The implementation uses regular expressions to:

1. Identify all information blocks and important document tags
2. Associate each important info tag with its corresponding information block
3. Extract document IDs, titles, and text content
4. Filter documents based on importance markers

The document pattern is matched using a regex that handles variations in spacing and optional quotes around titles. Our implementation includes appropriate error handling to manage parsing failures and maintains the original order of documents. The algorithm has $O(n)$ time complexity where n is the input string length, with additional factors related to the number of documents and information blocks.

B Human Alignment Study of Evaluation Metrics (GenAcc and EM)

To assess the alignment of our primary evaluation metric, **Generation Accuracy**, with human judgment, we conducted a human annotation study. We randomly sampled 1,000 answer generations from the general-domain QA test set. Each sample was labeled as `Correct (1)` or `Incorrect (0)` by human annotators, consisting of two Ph.D. students and one M.S. student majoring in computer science who evenly divided the annotation workload. Figure 7 shows the instruction, and the anonymous sheet[14] shows the raw results.

Human Evaluation Instruction

You are an evaluator for question-answering systems. Your task is to determine whether the system-generated answer aligns with the provided gold (reference) answers.
Evaluation Criteria: An answer should be marked as **correct (1)** if it:

- Contains the same key information as the golden answers;
- Expresses the same meaning, even if using different wording;
- Is factually consistent with the golden answers.

Please input **only**:

- `"1"` if the system's answer aligns with the golden answers;
- `"0"` if it does not.

Figure 7: Instruction for human evaluation of LLM generation.

We then compared these human labels against the binary decisions made by **Generation Accuracy** and **Exact Match**. As shown in Figure 8, Generation Accuracy demonstrates strong alignment with human evaluation, correctly identifying 96.4% of answers that were judged correct by humans. In contrast, Exact Match only captures 15.8% of such answers, largely due to its strict reliance on string matching.

These results confirm that Generation Accuracy is a more reliable and human-aligned metric, especially for evaluating free-form and abstractive answers where surface forms may differ despite

[14](Anonymous) raw results of human-metric alignment study: https://docs.google.com/spreadsheets/d/e/2PACX-1vQ-aAC6FNJYFJk1Ca8-EGN1zHa5z8WoF0Fm2VIHoWO_CA0Gaa-f_uy8JGX-NiRO9l2yDaJTxU0nObjG/pubhtml

Configuration	NQ†	TriviaQA	PopQA	HotpotQA†	2Wiki	Musique
Retrieval: 8, Selection: 3, Turns: 3						
Full Implementation	70.5$_{(3.2)}$	84.0$_{(24.6)}$	57.7$_{(5.9)}$	62.4$_{(11.1)}$	55.1$_{(8.3)}$	26.2$_{(7.9)}$
w/o Selection	70.7$_{(2.7)}$	83.1$_{(18.0)}$	57.2$_{(8.1)}$	61.1$_{(8.4)}$	58.9$_{(3.3)}$	22.5$_{(1.6)}$
w/o Begin with Search	68.6$_{(3.6)}$	82.2$_{(25.5)}$	55.0$_{(7.7)}$	57.0$_{(11.8)}$	46.8$_{(7.9)}$	20.9$_{(2.3)}$
w/o Both	70.8$_{(2.5)}$	83.2$_{(18.2)}$	56.5$_{(7.8)}$	60.1$_{(8.7)}$	57.4$_{(3.5)}$	21.8$_{(1.7)}$
Retrieval: 5, Selection: 3, Turns: 3						
Full Implementation	69.6$_{(3.5)}$	83.4$_{(24.3)}$	57.4$_{(5.8)}$	62.0$_{(11.9)}$	53.8$_{(7.8)}$	24.5$_{(2.3)}$
w/o Selection	70.8$_{(2.6)}$	81.8$_{(19.6)}$	56.3$_{(9.6)}$	60.8$_{(9.4)}$	57.8$_{(3.0)}$	22.4$_{(2.0)}$
w/o Begin with Search	67.6$_{(4.0)}$	81.2$_{(26.6)}$	55.0$_{(8.3)}$	57.4$_{(12.0)}$	50.0$_{(9.3)}$	21.1$_{(2.2)}$
w/o Both	70.6$_{(2.6)}$	81.9$_{(19.4)}$	56.0$_{(8.6)}$	60.0$_{(9.1)}$	57.6$_{(3.2)}$	22.3$_{(1.7)}$
Retrieval: 3, Selection: 3, Turns: 3						
Full Implementation	69.4$_{(3.5)}$	82.3$_{(24.4)}$	57.0$_{(5.7)}$	61.8$_{(11.7)}$	51.5$_{(8.2)}$	25.1$_{(2.3)}$
w/o Selection	69.7$_{(5.0)}$	81.6$_{(27.8)}$	56.1$_{(12.5)}$	59.7$_{(11.4)}$	56.2$_{(4.3)}$	23.5$_{(2.2)}$
w/o Begin with Search	67.7$_{(3.8)}$	81.1$_{(25.5)}$	54.2$_{(6.7)}$	58.1$_{(11.9)}$	50.2$_{(7.4)}$	22.1$_{(2.5)}$
w/o Both	69.2$_{(3.4)}$	81.5$_{(24.4)}$	55.2$_{(8.9)}$	58.3$_{(10.4)}$	54.6$_{(3.3)}$	22.5$_{(2.4)}$

Table 6: Ablation Studies of s3 on General Domain RAG. We show generation accuracy as the main results and exact match scores in brackets.

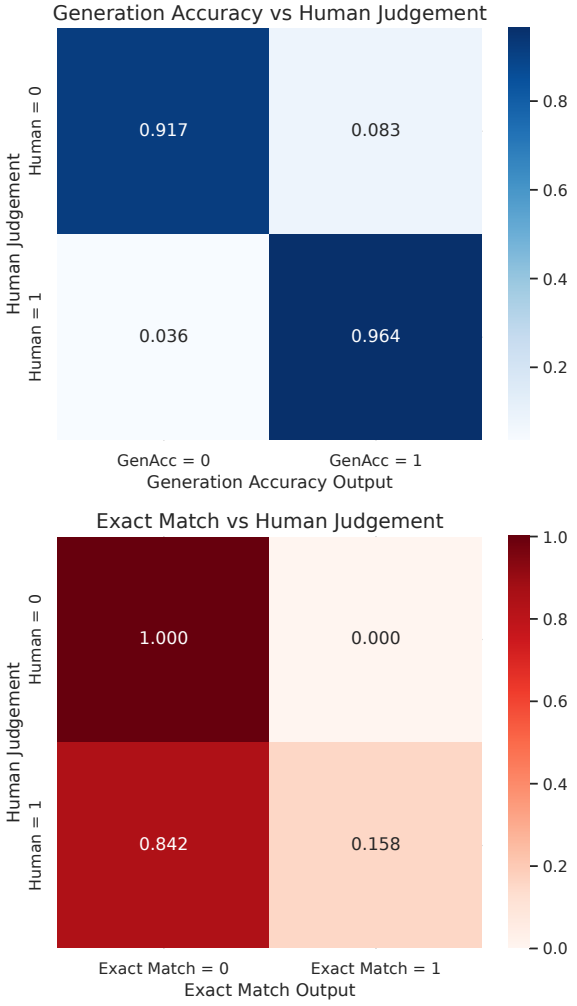

Figure 8: Confusion matrices comparing **Generation Accuracy** (top) and **Exact Match** (bottom) against human judgment. Each cell indicates the proportion of samples falling into the corresponding category.

semantic correctness, which also syncs with findings by prior studies applying similar evaluation methods (Song et al., 2025).

C Prompts

To train and evaluate the s3 framework effectively, we design three system prompts targeting distinct modules: the search policy (Searcher), answer generation, and judge-based evaluation. Each prompt is carefully constructed to ensure modularity, interpretability, and compatibility with frozen LLMs.

Searcher Prompt. The prompt for the Searcher (Figure 11) guides a trained policy to perform structured multi-turn search. It defines a loop-based instruction set that mimics real-world decision-making: the model emits a search query, inspects results, selects key documents, and decides whether to continue searching. This design supports iterative refinement and selection via:

- `<query>`: the generated search query in JSON format.
- `<information>`: the retrieved documents returned by the search engine.
- `<important_info>`: a subset of documents deemed most relevant (up to 3).
- `<search_complete>`: a binary decision on whether to stop searching.

Importantly, only selected documents in `<important_info>` are visible to the generator, encouraging the policy to focus on high-quality

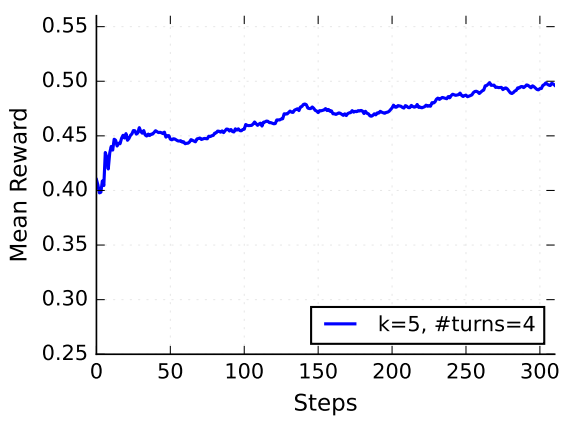

Figure 9: Scalability study: mean reward curve when training s3 (5-3-4) for 300 steps.

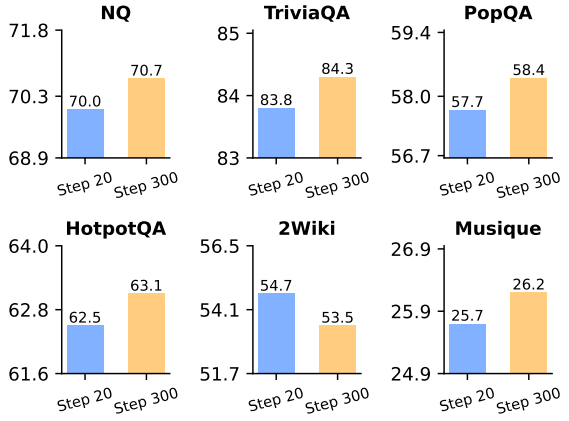

Figure 10: Performance comparison at Step 20 vs. Step 300 across datasets.

evidence rather than breadth. By isolating retrieval behavior from generation, this prompt allows reinforcement learning with a frozen black-box LLM using downstream answer quality as a reward.

Answer Generation Prompt. Figure 12 shows the prompt used for final answer generation. It provides the accumulated context from selected documents along with the user's original question. The generator is instructed to produce a direct, succinct answer without verbosity. This format simplifies reward computation and ensures generation outputs are consistent and easy to evaluate.

Judge_Check Prompt. To enable scalable, automated evaluation during training and inference, we employ a lightweight correctness prompt shown in Figure 13. This prompt asks an LLM to verify whether any gold answer appears in the predicted response. Unlike brittle exact-match metrics, this approach captures semantically valid completions even if they differ in surface form. During training, a quantized Qwen2.5-14B model is used for cost-effective inference, while evaluation employs Claude-3-Haiku for higher reliability.

Together, these prompts form a coherent pipeline that supports modular training and evaluation of retrieval-augmented generation systems. The clear separation of roles allows s3 to focus learning solely on the search agent, and our prompt designs play a key role in realizing this clean decoupling.

D Scalability Study

While s3 demonstrates strong performance with just 20 training steps (i.e., 2.4k examples), we investigate how performance evolves with additional data and training. Specifically, we train the "5-3-4" configuration for up to 300 steps.

Figure 9 shows the reward curve over training steps. We observe a consistent upward trend, indicating that the search policy continues to improve with more data and training iterations.

To quantify this improvement, Figure 10 compares the model's QA performance at step 20 and step 300 across six datasets. The results show that s3 scales gracefully: most datasets exhibit steady gains, with improvements particularly noticeable on PopQA, HotpotQA, and Musique.

These findings suggest that s3 can also benefit from larger-scale training, making it a flexible framework that performs well both in low-resource and high-resource settings.

Prompt Instructions for Searcher

You are a search copilot for a generation model. Based on a user's query and initial searched results, you will first determine if the searched results are enough to produce an answer.

If the searched results are enough, you will use `<search_complete>True</search_complete>` to indicate that you have gathered enough information for the generation model to produce an answer.

If the searched results are not enough, you will go through a loop of `<query>` → `<information>` → `<important_info>` → `<search_complete>` → `<query>` (if not complete) ..., to help the generation model to generate a better answer with more relevant information searched.

You should show the search query between `<query>` and `</query>` in JSON format.

Based on the search query, we will return the top searched results between `<information>` and `</information>`. You need to put the doc ids of the important documents (up to 3 documents, within the current information window) between `<important_info>` and `</important_info>` (e.g., `<important_info>[1, 4]</important_info>`).

A search query **must** be followed by a `<search_complete>` tag if the search is not complete.

After reviewing the information, you must decide whether to continue searching with a new query or indicate that the search is complete. If you need more information, use `<search_complete>False</search_complete>`. Otherwise, use `<search_complete>True</search_complete>` to terminate the search.

Note: Only the content between `<important_info>` will be used by the generation model to produce an answer.

[An abstractive example of `s3` loop]
`<question>` {question} `</question>`

`<information>` {initial_search_result} `</information>`

Figure 11: The prompt used for search policy (searcher).

Prompt for Answer Generation

Use the following contexts (some might be irrelevant) on demand:

Contexts: {context}

Question: {question}

Important: You MUST directly answer the question without any other text.

Figure 12: The prompt used for answer generation by frozen LLM.

Prompt for `Judge_Check`

Please check if any of the golden answers is contained in the following response:

{response}

Golden answers: {golden_answers}

Please directly answer with 'yes' or 'no'.

Figure 13: The prompt used for `judge_check`. Qwen2.5-14B-Instruct-GPTQ-Int4 is used in training, and Claude-3-Haiku is used in evaluation.

Question	<question> What year was the film made that was about the inventor of a type of passive solar house that is made of both natural and upcycled materials such as earth-packed tires? </question>
s3 Search Trajectory	<question> What year was the film made that was about the inventor of a type of passive solar house that is made of both natural and upcycled materials such as earth-packed tires? </question> <information> **Doc 1 (Earthship)**: An Earthship is a type of passive solar house... pioneered by architect Michael Reynolds... **Doc 2 (Mike Reynolds)**: Known for Earthship houses using recycled materials like tires... **Doc 3 (Don Stephens)**: Discusses earth-integrated designs and upcycled materials... </information> <search_complete> **False** </search_complete> <query> What year was the film made about **the inventor of Earthship**?" </query> <information> **Doc 1 (Garbage Warrior)**: Garbage Warrior is a 2007 film about architect Mike Reynolds, inventor of the Earthship style of building... **Doc 2 (Garbage Warrior)**: A 2007 film following Reynolds and his Earthship building... Doc 3 (Earthship): Mentions the film Garbage Warrior and its focus on off-grid housing... </information> <important_info>**[1, 2]**</important_info> <search_complete> **True** </search_complete>
s3 Searched Docs	**Doc 1 (Earthship)**: An Earthship is a type of passive solar house... pioneered by architect Michael Reynolds... **Doc 2 (Mike Reynolds)**: Known for Earthship houses using recycled materials like tires... **Doc 3 (Don Stephens)**: Discusses earth-integrated designs and upcycled materials... **Doc 4 (Garbage Warrior)**: Garbage Warrior is a 2007 film about architect Mike Reynolds, inventor of the Earthship style of building... **Doc 5 (Garbage Warrior)**: A 2007 film following Reynolds and his Earthship building...
Golden Answer	2007
RAG Answer	*There is no specific year mentioned for a film made about the inventor of the Earthship, which is a type of passive solar house made of natural and upcycled materials like earth-packed tires. The information provided does not include details about a particular film or its release year.*
s3 Answer	2007

Table 7: An example showing how s3 searches and selects correct evidence after issuing a focused search query. RAG fails to answer correctly without temporal grounding.