

# Reinforcement Fine-Tuning for History-Aware Dense Retriever in RAG

Yicheng Zhang<sup>1</sup> Zhen Qin<sup>2</sup> Zhaomin Wu<sup>3</sup> Wenqi Zhang<sup>2</sup> Shuguang Deng<sup>1</sup>

## Abstract

Retrieval-augmented generation (RAG) enables large language models (LLMs) to produce evidence-based responses, and its performance hinges on the matching between the retriever and LLMs. Retriever optimization has emerged as an efficient alternative to fine-tuning LLMs. However, existing solutions suffer from objective mismatch between retriever optimization and the goal of RAG pipeline. Reinforcement learning (RL) provides a promising solution to address this limitation, yet applying RL to retriever optimization introduces two fundamental challenges: 1) the deterministic retrieval is incompatible with RL formulations, and 2) state aliasing arises from query-only retrieval in multi-hop reasoning. To address these challenges, we replace deterministic retrieval with stochastic sampling and formulate RAG as a Markov decision process, making retriever optimizable by RL. Further, we incorporate retrieval history into the state at each retrieval step to mitigate state aliasing. Extensive experiments across diverse RAG pipelines, datasets, and retriever scales demonstrate consistent improvements of our approach in RAG performance.

## 1. Introduction

Retrieval-Augmented Generation (RAG) has become a widely adopted framework for grounding large language models (LLMs) in external knowledge by retrieving supporting evidence and conditioning generation on the retrieved context (Fan et al., 2024; Lewis et al., 2020). This retrieval-conditioned generation paradigm can alleviate hallucinations (Niu et al., 2024) and has proven particularly beneficial for complex question answering (Trivedi et al., 2023), including multi-hop reasoning that requires com-

<sup>1</sup>College of Computer Science and Technology, Zhejiang University, Hangzhou, China <sup>2</sup>Ningbo Global Innovation Center, Zhejiang University, Ningbo, China <sup>3</sup>Department of Computer Science, National University of Singapore, Singapore. Correspondence to: Zhen Qin <zhenqin@zju.edu.cn>, Shuguang Deng <dengsg@zju.edu.cn>.

posing evidence across multiple sources. Nevertheless, the retriever and LLM-based generator are typically optimized in isolation, which can induce an objective mismatch between retrieval relevance and downstream answer correctness, thereby limiting end-to-end RAG performance.

Previous studies have investigated fine-tuning the LLM to better incorporate retrieved documents, i.e., adapting the LLMs to a given retriever. As LLMs continue to scale, their optimization is increasingly becoming a resource-intensive task (Naveed et al., 2025). In this context, a growing body of work instead focuses on improving the retriever to enhance RAG efficiency and effectiveness, i.e., adapting the retriever to the LLMs (Shi et al., 2024). Currently, retriever-centric optimization is commonly conducted via supervised fine-tuning (SFT) with synthetic or proxy targets, which may not faithfully reflect downstream answer correctness. As a result, the retriever can be optimized toward objectives that are only weakly aligned with the end task, limiting overall RAG performance.

To directly align retriever optimization with the end-to-end objective of RAG, we seek to optimize the retriever using reinforcement learning (RL) based on downstream task rewards; however, this setting gives rise to two key challenges: (1) Standard retrieval is typically realized via deterministic top- $k$  selection, which is incompatible with RL that requires a stochastic policy capable of exploring alternative actions through sampling. (2) In multi-hop reasoning, conditioning retrieval solely on the current query leads to state ambiguity, as identical queries may arise from different retrieval histories with distinct information needs, thereby hindering effective reward assignment.

To address the above two challenges, this work proposes HARR, an end-to-end reinforcement fine-tuning framework for **H**istory-**A**ware **R**einforced **R**etriever in RAG pipelines. HARR addresses the above two challenges through two strategic designs. To address challenge (1), we replace deterministic top- $k$  retrieval with probabilistic document sampling, allowing the retriever to be modeled as a stochastic policy that supports exploration, and then formulate retrieval-augmented question answering as a Markov decision process (MDP). These efforts make the optimization of retriever amenable to RL. To tackle challenge (2), we incorporate retrieval history into the state at each hop,

enabling the retriever to capture the evolving information need across retrieval steps and mitigating state ambiguity in multi-hop reasoning. With these designs, HARR aligns the optimization of retriever with end-to-end task rewards.

The contributions of this work are summarized as follows:

- We replace the conventional deterministic top- $k$  retrieval with probabilistic document sampling, and formulate retrieval-augmented question answering as an MDP, thereby making dense retrievers amenable to RL.
- We consider state aliasing as a key challenge in applying RL to optimize the retriever in multi-hop RAG and propose a history-aware state representation that conditions retrieval on the history, in order to reduce state ambiguity in multi-hop reasoning.
- We perform extensive experiments across diverse RAG pipelines, datasets, and retriever scales to validate the effectiveness of the proposed framework. The experimental results demonstrate that the reinforcement fine-tuning of retrievers consistently improves downstream RAG performance. Our code is available at <https://github.com/zyc140345/HARR>.

## 2. Related Work

### 2.1. RAG Workflow

RAG systems can be categorized by their retrieval-generation workflows, which define how retrieval is invoked and how retrieved evidence is consumed during generation. Existing workflows range from simple single-hop pipelines to more structured multi-hop and agentic designs, offering different trade-offs in reasoning capability, system complexity, and efficiency (Fan et al., 2024; Singh et al., 2025).

**Single-Hop RAG.** Single-hop RAG follows a retrieve-then-generate paradigm, where relevant documents are retrieved once based on the input query and directly provided to the LLM. This workflow is widely adopted in early and practical RAG systems due to its simplicity and efficiency (Lewis et al., 2020; Guu et al., 2020), and has been shown effective for factoid or shallow knowledge-intensive tasks (Izacard & Grave, 2021). However, performing retrieval only once limits its ability to support complex, multi-evidence reasoning. Accordingly, our work focuses on multi-hop and agentic settings to improve applicability.

**Multi-Hop RAG.** To address queries requiring compositional or multi-step reasoning, multi-hop RAG extends the workflow to perform retrieval iteratively or through structured reasoning paths. Representative designs include iterative retrieval-generation loops (Trivedi et al., 2023; Shao et al., 2023) as well as tree- (Yao et al., 2023a) or

graph- (Besta et al., 2024) structured workflows that explore and aggregate multiple reasoning branches. However, these workflows typically rely on independently pretrained components composed heuristically at inference time. Our method addresses this limitation by optimizing the retriever to better support downstream reasoning.

**Agentic and Adaptive Workflows.** Recent work further generalizes multi-hop RAG into agentic workflows, where autonomous agents dynamically decide when and how to retrieve information. These systems incorporate planning, reflection, or tool use to adapt retrieval strategies based on intermediate states (Yao et al., 2023b; Schick et al., 2023), and include methods such as FLARE (Jiang et al., 2023) that trigger retrieval based on generation uncertainty. However, retrieval decisions in agentic workflows are often governed by fixed heuristics or separately trained modules. Our approach complements these workflows by learning retrieval policies directly optimized for downstream task performance.

Overall, existing RAG workflows improve reasoning capability through more elaborate retrieval control, but largely rely on heuristics or fixed policies to orchestrate retrieval and generation. This leaves open the question of how to systematically learn retrieval decisions within RAG pipelines.

### 2.2. Optimization for RAG Components

Most RAG systems are constructed by independently pre-training retrievers and LLMs with different objectives, and then integrating them into a single pipeline at inference time. This loose coupling often leads to suboptimal cooperation. Recent work seeks to reduce this mismatch by optimizing different components of the RAG pipeline.

**Optimizing the LLMs.** A first line of work improves the LLM’s ability to plan, retrieve, and utilize evidence. Supervised fine-tuning (SFT) methods rely on annotated or synthesized trajectories to teach iterative retrieval and reasoning, such as ITER-RETEGEN (Shao et al., 2023), Self-RAG (Asai et al., 2024), and CoRAG (Wang et al., 2025). Reinforcement learning (RL) further optimizes generation or search behaviors with outcome-based rewards, as explored in Search-R1 (Jin et al., 2025), DeepRetrieval (Jiang et al., 2025), and ZeroSearch (Sun et al., 2025a). While effective, LLM-centric optimization is often expensive due to the large parameter scale of LLMs and the cost of repeated rollouts, and does not directly address cases where critical evidence is not recalled by the underlying retriever. Our work instead focuses on lightweight retriever optimization to improve evidence recall while keeping the LLM fixed.

**Joint Optimization.** Another direction jointly optimizes retrievers and LLMs to improve end-to-end performance. Early work such as RAG (Lewis et al., 2020) treats retrieved

documents as latent variables and maximizes marginal likelihood, while subsequent methods improve optimization efficiency or differentiability through sampling-based training (Zamani & Bindersky, 2024), staged optimization (Lin et al., 2024), or preference-based objectives (Li et al., 2024). Despite strong empirical results, joint optimization typically incurs high engineering and computational cost, and often requires both the retriever and LLM to be white-box models, which limits flexibility. In contrast, our approach focuses on retriever-only optimization to reduce training cost and improve deployment flexibility.

**Inserting Connecting Modules.** Instead of updating large models, some approaches introduce lightweight modules between retrieval and generation to better mediate their interaction. Examples include external policies for retrieval control (Kulkarni et al., 2024), multi-agent memory selection (Wang et al., 2024), and context selection (Ke et al., 2024) or reconstruction (Li & Ramakrishnan, 2025). These methods are training-efficient and flexible across different backbones, but add extra system components and inference steps, and their effectiveness is constrained by the quality of the retrieved evidence. Our work addresses this limitation by directly optimizing the retriever to influence which evidence is retrieved, without introducing additional inference modules or overhead.

**Optimization of Retriever.** Motivated by the limitations of LLM-centric optimization, joint training, and lightweight bridging modules, recent work explores optimizing the retriever to better balance resource cost, optimization headroom, and deployment flexibility. Compared to LLMs and intermediate modules, retriever optimization is lightweight, can directly alter retrieved evidence, and naturally supports black-box LLMs. Representative approaches include RE-PLUG (Shi et al., 2024), which distills LLM document preferences into the retriever, and Reward-RAG (Nguyen et al., 2024) and R3 (Zhou & Chen, 2026), which leverage answer-level feedback via contrastive or reinforcement-inspired objectives. However, most existing methods rely on proxy objectives rather than directly optimizing downstream generation performance, leaving room for more direct retriever-generation objective matching. Related retrieval methods outside the RAG setting also explore reinforcement learning for dense retrieval (Liu et al., 2025) or memory selection (Zhou et al., 2025), but differ in task formulation or retrieval substrates, thus do not address lightweight, train-inference objective consistency within RAG pipelines. Our work fills this gap by enabling lightweight retriever optimization that directly matches retrieval behavior to downstream generation objectives.

### 3. Problem Formulation

We consider a multi-hop retrieval-augmented generation (RAG) system comprising an LLM  $\pi_{\text{LLM}}$ , a dense retriever  $\pi_{\text{ret}}$ , and an external knowledge corpus  $\mathcal{D} = \{d_i\}_{i=1}^N$ . The interaction is modeled as a sequential decision-making process starting with an initial query  $q_0$ .

At each step  $t$ , the LLM  $\pi_{\text{LLM}}$  conditions on the accumulated retrieval history  $\mathcal{H}_{t-1}$  ( $\mathcal{H}_0 = (q_0)$ ) to determine the next output, which is either a termination signal or a sub-query  $q_t$ . Given a sub-query  $q_t$ , the retriever  $\pi_{\text{ret}}$  selects a ranked list of top- $k$  documents  $D_t = (d_t^1, \dots, d_t^k)$  from  $\mathcal{D}$ . Subsequently, the LLM synthesizes an intermediate observation  $o_t \sim \pi_{\text{LLM}}(\cdot | q_t, D_t)$  to summarize the evidence, updating the history to  $\mathcal{H}_t = \mathcal{H}_{t-1} \circ (q_t, o_t)$ . Upon termination after  $T$  steps, the LLM produces a final answer  $y \sim \pi_{\text{LLM}}(\cdot | \mathcal{H}_T)$ .

The overarching learning objective is to optimize the system parameters such that the generated answer  $y$  maximizes a utility function  $\text{Score}(y, y^*)$  (e.g., F1 score or Exact Match) with respect to the ground-truth answer  $y^*$ . While standard approaches often optimize this via proxy losses, our goal is to align the retrieval policy  $\pi_{\text{ret}}$  directly with this end-to-end generation metric.

## 4. Approach

### 4.1. Overview

In this section, we present the proposed reinforcement learning framework for matching retriever behavior with LLMs’ preferences in multi-hop RAG systems. We formulate retrieval as an MDP with a history-aware state representation to address state aliasing (§4.2), and parameterize the retriever as a stochastic ranking policy over ordered document lists (§4.3). The retriever is optimized using Group Relative Policy Optimization (GRPO) with sparse terminal rewards (§4.4), together with practical approximations that make training feasible on large-scale corpora (§4.5).

### 4.2. MDP Formulation

To enable RL of the retriever in multi-hop RAG systems, we formulate the interaction between the retriever and the rest of the RAG pipeline as a Markov decision process (MDP)  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r)$ . In this formulation, the retriever acts as the policy, while the LLM and the document corpus define the environment dynamics. We now specify each component of the MDP.

**History-Aware State Space  $\mathcal{S}$ .** Given the retriever as the policy, a straightforward state design defines the state solely by the current sub-query  $q_t$ , which matches the retriever’s pretraining objective. However, this state representation

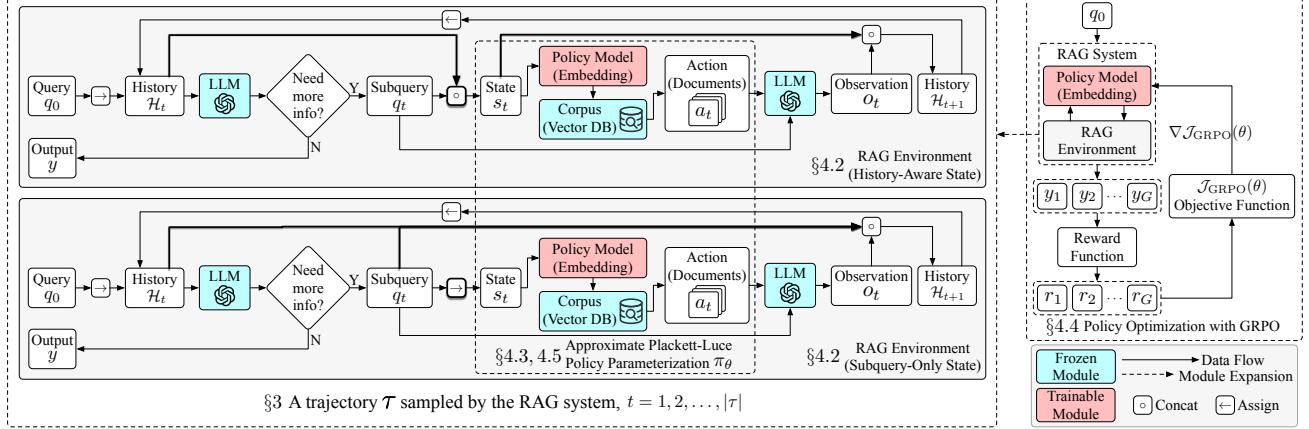


Figure 1. Overview of HARR

suffers from state aliasing. In multi-hop RAG, identical sub-queries  $q_t$  may arise from different retrieval histories  $\mathcal{H}_{t-1}$ , yet be mapped to the same state under a query-only representation. Consequently, executing the same retrieval action in this apparent state can lead to different downstream reasoning trajectories and final rewards, yielding inconsistent learning signals for the same state–action pair. This leads to high-variance policy gradient estimates and hinders effective learning.

To mitigate the state aliasing issue, we define each state  $s_t \in \mathcal{S}$  at step  $t$  as a history-aware representation

$$s_t = (\mathcal{H}_{t-1}, q_t), \quad (1)$$

where  $\mathcal{H}_{t-1}$  denotes the retrieval history up to step  $t - 1$ . By incorporating the retrieval history into the state, retrieval decisions made under identical sub-queries but different histories are distinguished. As a result, state-action pairs that would otherwise be aliased under a query-only state formulation are separated, leading to more consistent downstream outcomes and learning signals for each state–action pair.

**Action Space  $\mathcal{A}$ .** At each step  $t$ , the action is to select a ranked list of  $k$  documents from the corpus  $\mathcal{D}$ , denoted  $a_t = D_t = (d_t^1, \dots, d_t^k)$ . Thus, the action space  $\mathcal{A}$  comprises all such ordered  $k$ -documents, with  $|\mathcal{A}| = \frac{|\mathcal{D}|!}{(|\mathcal{D}|-k)!}$ .

**State Transition  $P$ .** Given state  $s_t = (\mathcal{H}_{t-1}, q_t)$  and action  $a_t = D_t$ , the state transition is induced by the LLM. Conditioned on the current sub-query and retrieved documents, the LLM produces an intermediate observation

$$o_t \sim \pi_{\text{LLM}}(\cdot | q_t, D_t), \quad (2)$$

which summarizes the retrieved evidence. The retrieval history is then updated as  $\mathcal{H}_t = \mathcal{H}_{t-1} \circ (q_t, o_t)$ , and the LLM emits the next sub-query

$$q_{t+1} \sim \pi_{\text{LLM}}(\cdot | \mathcal{H}_t). \quad (3)$$

The resulting next state is  $s_{t+1} = (\mathcal{H}_t, q_{t+1})$ .

Under this formulation, the state transition probability can be expressed as

$$\begin{aligned} P(s_{t+1} | s_t, a_t) &= \sum_{o_t} P(s_{t+1}, o_t | s_t, a_t) \\ &= \sum_{o_t} P(o_t | s_t, a_t) \cdot P(s_{t+1} | s_t, a_t, o_t) \quad (4) \\ &= \sum_{o_t} \underbrace{\pi_{\text{LLM}}(o_t | q_t, D_t)}_{\text{Observation Generation}} \cdot \underbrace{\pi_{\text{LLM}}(q_{t+1} | \mathcal{H}_t)}_{\text{Sub-query Generation}}. \end{aligned}$$

**Reward  $r$ .** To optimize the retriever for final question answering performance, we use a sparse terminal reward:

$$r_t = \begin{cases} \text{F1}(y, y^*), & t = T \\ 0, & t < T, \end{cases} \quad (5)$$

where  $y$  is the answer generated by conditioning the LLM on the full retrieval history  $\mathcal{H}_T$ , and  $y^*$  is the ground-truth answer. The F1 score is computed as

$$\text{F1}(y, y^*) = \frac{2 |\mathcal{T}(y) \cap \mathcal{T}(y^*)|}{|\mathcal{T}(y)| + |\mathcal{T}(y^*)|}, \quad (6)$$

where  $\mathcal{T}(\cdot)$  denotes the set of tokens in a given text.

### 4.3. Policy Parametrization

We parameterize the retriever as a stochastic policy  $\pi_\theta$  that maps a state  $s_t$  to a probability distribution over ordered lists of  $k$  documents  $D_t$ , from which an action  $a_t = D_t$  is sampled at each retrieval step.

To model this distribution, we view retrieval as sequential sampling without replacement from the corpus, which is modeled using the Plackett-Luce ranking model (Plackett, 1975). Under this model, the probability of selecting an ordered document list  $D_t = (d_t^1, \dots, d_t^k)$  factorizes into a

product of conditional selection probabilities

$$\pi_\theta(a_t = D_t \mid s_t) = \prod_{i=1}^k \pi_\theta(d_t^i \mid s_t, d_t^{1:i-1}). \quad (7)$$

At each position  $i$ , the next document is sampled from the remaining candidate set  $\mathcal{D} \setminus \{d_t^1, \dots, d_t^{i-1}\}$  according to a softmax distribution

$$\pi_\theta(d_t^i \mid s_t, d_t^{1:i-1}) = \frac{\exp(f_\theta(s_t, d_t^i)/\alpha)}{\sum_{d \in \mathcal{D} \setminus \{d_t^j\}_{j=1}^{i-1}} \exp(f_\theta(s_t, d)/\alpha)}, \quad (8)$$

where  $\alpha > 0$  is a temperature parameter. The scoring function  $f_\theta(s, d)$  represents the policy score of document  $d$  under state  $s$  and is defined as

$$f_\theta(s, d) = \text{sim}(\text{enc}_\theta(s), \text{enc}_\theta(d)), \quad (9)$$

where  $\text{enc}_\theta(\cdot)$  is a text encoder parameterized by  $\theta$ , and  $\text{sim}(\cdot, \cdot)$  denotes a similarity function (e.g., inner product).

#### 4.4. Policy Optimization

Given the MDP formulation and the policy parametrization, our objective is to optimize the retriever policy to maximize the expected terminal reward,

$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} [r_{|\tau|}], \quad (10)$$

where  $\tau = (s_1, a_1, \dots, s_{|\tau|}, a_{|\tau|})$  denotes a retrieval trajectory induced by the policy, and  $r_{|\tau|}$  is the terminal reward defined in § 4.2.

In our formulation, rewards are sparse and only observed at the end of a trajectory, while the state space is history-dependent and high-dimensional. Estimating accurate value functions in this setting is challenging and introduces additional model complexity. We therefore adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which optimizes the policy using relative outcome-based advantages without explicit value function estimation.

Specifically, for a given query  $q_0 \sim P(Q)$ , we sample a group of  $G$  retrieval trajectories  $\{\tau_i\}_{i=1}^G$  from the old policy  $\pi_{\theta_{\text{old}}}$ . Each trajectory  $\tau_i$  receives a terminal reward  $r_i$ , which is normalized within the group to obtain the advantage

$$A_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}, \quad \mathbf{r} = \{r_1, \dots, r_G\}. \quad (11)$$

The retriever policy is then optimized by maximizing the GRPO objective

$$\begin{aligned} \mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E} \left[ \frac{1}{G} \sum_{i=1}^G \frac{1}{|\tau_i|} \sum_{t=1}^{|\tau_i|} \min \left( \frac{\pi_\theta(a_{i,t} \mid s_{i,t})}{\pi_{\theta_{\text{old}}}(a_{i,t} \mid s_{i,t})} A_i, \right. \right. \\ \left. \left. \text{clip} \left( \frac{\pi_\theta(a_{i,t} \mid s_{i,t})}{\pi_{\theta_{\text{old}}}(a_{i,t} \mid s_{i,t})}, 1 - \epsilon, 1 + \epsilon \right) A_i \right) \right], \end{aligned} \quad (12)$$

where  $\epsilon$  is the clipping parameter.

#### 4.5. Practical Training Approximations

The GRPO objective optimizes the retriever policy over the full document corpus. However, computing the policy probabilities in (12) is prohibitively expensive for large-scale corpora, as it requires normalization over the entire corpus at each retrieval step and recomputation of document embeddings after each policy update. We therefore introduce the following approximations to enable efficient training.

**Candidate Pool Restriction.** At each retrieval step  $t$ , instead of normalizing over the full corpus  $\mathcal{D}$ , we first construct a candidate pool  $\mathcal{C}_t \subset \mathcal{D}$  by retrieving the top- $K$  documents using the scoring function in (9), where  $k < K \ll |\mathcal{D}|$ . Policy sampling and probability normalization are then restricted to this candidate pool. Under this approximation, the conditional selection probability in (8) is given by

$$\pi_\theta(d_t^i \mid s_t, d_t^{1:i-1}) \approx \frac{\exp(f_\theta(s_t, d_t^i)/\alpha)}{\sum_{d \in \mathcal{C}_t \setminus \{d_t^j\}_{j=1}^{i-1}} \exp(f_\theta(s_t, d)/\alpha)}. \quad (13)$$

**Document Encoder Freezing.** To further reduce computational overhead, we freeze the document encoder during training and only update the state encoder. The scoring function in (9) then takes the form

$$f_\theta(s, d) = \text{sim}(\text{enc}_\theta(s), \text{enc}(d)), \quad (14)$$

where document embeddings  $\text{enc}(d)$  are precomputed and reused across policy updates, while the retriever adapts through learned state representations.

## 5. Experiments

The experiments aim at demonstrating: (1) HARR improves end-to-end question answering performance across different RAG pipelines, datasets, and retriever encoders (§5.2); (2) the proposed RL-based fine-tuning and history-aware state representation both contribute positively to performance gains (§5.3); and (3) HARR achieves favorable training efficiency in terms of convergence speed, training time, and memory usage (§5.4).

### 5.1. Experimental Setup

We position HARR as a *plug-in* retriever optimization module compatible with existing RAG systems. To enable a fair assessment, we fix the RAG pipeline configurations—including LLMs, prompts, and interaction workflows—and evaluate performance gains solely by varying the retriever optimization strategy.

**RAG Pipelines.** We evaluate our method on two representative RAG pipelines to demonstrate its versatility. (1)

*ReAct Agent* (Yao et al., 2023b): a ReAct-style pipeline using Qwen3-8B<sup>1</sup> as the LLM, where the retriever is invoked as an external tool. (2) *Search-R1* (Jin et al., 2025): a pipeline with an RL-fine-tuned Qwen2.5-7B<sup>2</sup> LLM that interleaves reasoning and retrieval, with the retriever serving as its search engine.

**Baselines.** We compare against the following baselines under the same fixed RAG pipelines. (1) *Frozen Retriever*: the pre-trained retriever is used directly without adaptation, serving as a lower-bound reference. (2) *REPLUG*: a strong baseline based on the LSR method from REPLUG (Shi et al., 2024), which fine-tunes the retriever by matching its retrieval score distribution to the LLM’s perplexity distribution over the top- $k$  documents.

**Datasets & Evaluation.** To provide a comprehensive evaluation, we conduct experiments on both multi-hop and single-hop question answering benchmarks, using HotpotQA (Yang et al., 2018) and Natural Questions (NQ) (Kwiatkowski et al., 2019), respectively. Following prior work (Guan et al., 2025), we report SQuAD (Rajpurkar et al., 2016) Exact Match (EM) and F1 scores as evaluation metrics.

**Implementation.** All experiments are conducted on a single node equipped with 8 NVIDIA A100 GPUs. Our method is implemented using the TRL (von Werra et al., 2020) and LlamaIndex (Liu, 2022) libraries, while REPLUG follows the FedRAG (Fajardo & Emerson, 2025) implementation, with all LLMs deployed via vLLM (Kwon et al., 2023). Unless otherwise specified, we employ the 2018 Wikipedia dump (Karpukhin et al., 2020) as the retrieval corpus, indexed in Qdrant (Qdrant Team, 2023) using an INT8-quantized HNSW index ( $m = 16$ ,  $\text{ef\_construct} = 100$ ). We use Qwen3-Embedding-4B<sup>3</sup> and -0.6B<sup>4</sup> as retriever encoders, fine-tuned via LoRA (Hu et al., 2022) ( $r = 32$ ,  $\alpha = 64$ , dropout = 0.05) using the AdamW (Loshchilov & Hutter, 2017) optimizer with FP16 precision, a constant learning rate of  $1 \times 10^{-5}$  without warmup, a batch size of 16, and 100 training steps. For our method, we filter the training set to ensure non-zero reward variance under the initial policy across 8 rollouts and set the training retrieval top- $k$  to 3, with a temperature  $\alpha = 0.05$ , a candidate pool size  $K = 30$ , a GRPO group size  $G = 8$ ,

<sup>1</sup>Adopt checkpoint from <https://huggingface.co/Qwen/Qwen3-8B>.

<sup>2</sup>Adopt checkpoint from [https://huggingface.co/PeterJinGo/SearchR1-nq\\_hotpotqa\\_train-qwen2.5-7b-em-ppo](https://huggingface.co/PeterJinGo/SearchR1-nq_hotpotqa_train-qwen2.5-7b-em-ppo).

<sup>3</sup>Adopt checkpoint from <https://huggingface.co/Qwen/Qwen3-Embedding-4B>.

<sup>4</sup>Adopt checkpoint from <https://huggingface.co/Qwen/Qwen3-Embedding-0.6B>.

and a clipping ratio  $\epsilon = 0.2$  (no KL regularization). At inference time, all methods use  $k = 3$  and are evaluated on the full, unfiltered development splits following the original dataset partitions for a fair comparison.

## 5.2. Comparison of Accuracy

Table 1 reports the end-to-end question answering performance across different RAG pipelines, datasets, and retriever encoders, while Figure 2 depicts the corresponding training reward trajectories on HotpotQA. We analyze the results from four complementary perspectives.

**Comparison with Frozen Retrievers.** From Table 1, our approach improves performance in 18 out of 20 reported metrics compared to the Frozen Retriever baseline, demonstrating consistent gains across diverse experimental settings. We attribute these improvements to end-to-end reinforcement learning, which aligns the retriever with the LLM and enables more effective coordination between retrieval and reasoning components.

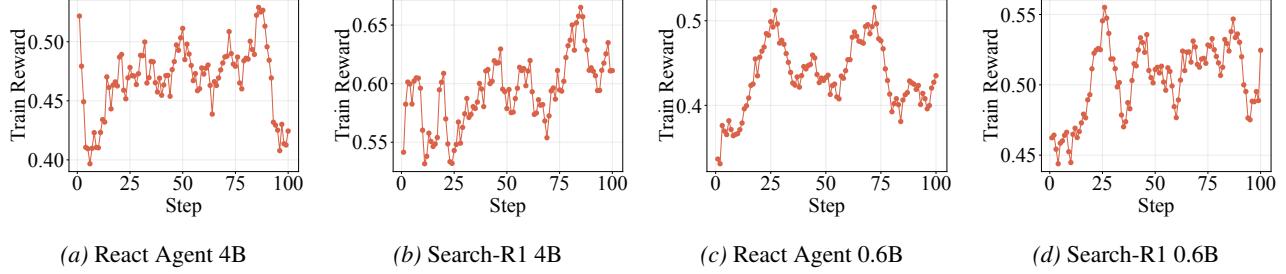
**Superiority over Supervised Fine-Tuning.** From Table 1, our method also outperforms REPLUG in 17 out of 20 metrics. While REPLUG optimizes proxy objectives that approximate LLM preferences, such objectives do not necessarily translate into improved end-to-end question answering performance. In contrast, our approach directly optimizes the downstream task reward, reducing the mismatch between training objectives and inference-time behavior and allowing the retriever to learn document selection strategies that better support the overall reasoning pipeline.

**Training Dynamics.** Figure 2 illustrates the training reward trajectories of HARR. We observe an initial decrease in reward followed by a steady increase. This early decline is likely caused by a distribution shift from the history-aware state representation, which differs from the retriever’s pre-training regime. After an adaptation phase, rewards increase consistently, indicating that the retriever learns to exploit historical context to improve retrieval decisions.

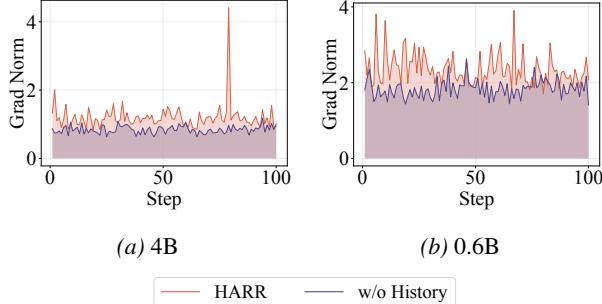
**Analysis of Performance Variations.** Despite the overall gains, several trends are worth noting. First, reward trajectories exhibit oscillations during training (Figure 2), which can be attributed to sparse terminal rewards and the off-policy approximations introduced for scalability (see §4.5). Second, improvements are more pronounced on the multi-hop HotpotQA dataset than on the single-hop NQ benchmark, and when using the larger 4B retriever encoder compared to the 0.6B model. These observations suggest that history-aware state representations are most beneficial when retrieval decisions depend on longer reasoning contexts, and that sufficient model capacity is required to effec-

**Table 1.** End-to-end QA performance (EM/F1; higher is better) across two RAG agents (ReAct vs. Original), two datasets (HotpotQA, NQ), and two retriever encoders (Qwen3-Embedding-4B, Qwen3-Embedding-0.6B). For each base agent, we compare the frozen-retriever baseline (ReAct Agent / Search-R1), a plug-in retrieval variant (REPLUG), and our method HARR applied within the same agent pipeline. **Bold** denotes the best result in each column, and underline denotes the second-best. *Gains* reports the relative improvement (%) of HARR over the strongest non-HARR baseline under the same base agent for that column.

Base Agent	Approach	Qwen3-Embedding-4B				Qwen3-Embedding-0.6B				Average	
		HotpotQA		NQ		HotpotQA		NQ			
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
ReAct	Original	31.28	40.42	32.82	43.86	28.74	37.47	30.10	40.46	30.73	40.55
	REPLUG	31.91	40.91	32.60	43.77	28.53	37.04	30.12	40.58	30.79	40.57
	<b>HARR (Ours)</b>	<b>32.34</b>	<b>41.55</b>	<b>33.46</b>	<b>44.56</b>	<b>29.60</b>	<b>38.14</b>	<b>30.16</b>	<b>40.72</b>	<b>31.39</b>	<b>41.24</b>
	<i>Gains</i>	+1.35%	+1.56%	+1.95%	+1.60%	+3.01%	+1.76%	+0.11%	+0.36%	+1.61%	+1.32%
Search-R1	Original	40.04	50.48	43.91	53.61	37.88	47.81	40.92	50.08	40.69	50.50
	REPLUG	<b>40.39</b>	50.90	43.95	53.52	37.66	47.67	<b>41.12</b>	<b>50.27</b>	<b>40.78</b>	<b>50.59</b>
	<b>HARR (Ours)</b>	40.30	<b>51.02</b>	<b>44.92</b>	<b>54.45</b>	<b>38.34</b>	<b>48.63</b>	40.87	50.07	<b>41.11</b>	<b>51.04</b>
	<i>Gains</i>	-0.23%	+0.24%	+2.21%	+1.57%	+1.21%	+1.72%	-0.61%	-0.40%	+0.64%	+0.78%



**Figure 2.** Training reward trajectories on the HotpotQA dataset. Subplot titles denote the specific RAG pipeline and retriever encoder used. All curves are smoothed using Exponential Moving Average (EMA) with a window size of 8.



**Figure 3.** Gradient norm trajectories on HotpotQA using the ReAct Agent pipeline. Subplot titles denote the retriever encoder size.

tively leverage such contextual information.

### 5.3. Ablation Study

We conduct ablation studies on the ReAct Agent pipeline to analyze the contributions of the proposed history-aware state representation and the RL-based retriever fine-tuning within HARR. Table 2 reports the results across two datasets and two retriever encoder sizes, while Figure 3 illustrates

the corresponding gradient norm trajectories.

**Effect of RL and History-Aware State.** Table 2 demonstrates that both proposed components are essential for achieving optimal performance. Specifically, removing the history-aware state degrades performance in 9 out of 10 metrics, while excluding the RL-based optimization leads to drops across all 10 metrics. The notably larger degradation caused by removing RL identifies it as the primary driver of performance gains, while the history-aware state serves as a critical complementary module.

**RL versus History-Aware State in Isolation.** We further analyze the behavior of each component in isolation. When employing the history-aware state without RL-based fine-tuning, the average performance fails to surpass that of the frozen retriever. We attribute this to the distribution shift introduced by the augmented state, which the pre-trained encoder cannot effectively leverage without end-to-end adaptation. Conversely, applying RL without the history-aware state still yields improvements over the frozen baseline, indicating that our RL framework is robust and capable of enhancing performance even in the presence of

Table 2. Ablation study on the ReAct Agent pipeline across two datasets and two retriever encoders. “w/o History” and “w/o RL” refer to removing the history-aware state space and the RL-based retriever fine-tuning from HARR, respectively. Notations follow Table 1.

Approach based on ReAct Agent	Qwen3-Embedding-4B				Qwen3-Embedding-0.6B				Average	
	HotpotQA		NQ		HotpotQA		NQ			
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Original	31.28	40.42	32.82	43.86	28.74	37.47	30.10	40.46	30.73	40.55
HARR w/o History	<u>31.56</u>	<u>40.74</u>	32.68	43.86	<u>29.37</u>	<u>37.99</u>	<b>30.19</b>	<u>40.48</u>	<u>30.95</u>	<u>40.77</u>
HARR w/o RL	31.42	40.63	<u>32.98</u>	<u>44.14</u>	27.52	35.74	29.80	40.20	30.43	40.18
HARR	<b>32.34</b>	<b>41.55</b>	<b>33.46</b>	<b>44.56</b>	<b>29.60</b>	<b>38.14</b>	<u>30.16</u>	<b>40.72</b>	<b>31.39</b>	<b>41.24</b>

state aliasing.

**Impact of Model Capacity.** The relative contribution of each component varies with retriever capacity. For the smaller 0.6B encoder, adding RL-based fine-tuning on top of the frozen retriever yields larger marginal gains. This suggests that smaller models may initially suffer from a larger gap in matching LLM preferences, thereby benefiting more from the RL-driven optimization. In contrast, for the larger 4B encoder, introducing the history-aware state on top of RL leads to more pronounced improvements, indicating that higher-capacity models are better equipped to exploit historical context for informed retrieval decisions.

**Gradient Analysis.** Finally, Figure 3 provides insight into the optimization dynamics. Compared to the variant trained without history, HARR exhibits consistently larger gradient norms throughout training. This observation is consistent with our hypothesis that the history-aware state mitigates state aliasing, thereby reducing reward ambiguity and yielding stronger learning signals for policy optimization.

#### 5.4. Training Efficiency

Since HARR exclusively fine-tunes the retriever encoder while keeping the LLM and inference workflow unchanged, it introduces no additional inference overhead; therefore, we focus on training efficiency in terms of convergence speed, wall-clock training time, and memory usage. As shown by the training reward curves in Figure 2, RL-based retriever optimization converges rapidly, typically within 100 training steps across different settings. Consistent with this observation, Table 3 shows that the total training time of a single experiment is at most around three hours, which is substantially lower than the cost of fine-tuning LLMs, often reported to require over ten hours even with comparable hardware (Sun et al., 2025b). In terms of memory consumption, training the 0.6B retriever requires less than 32 GB GPU memory, making it feasible on consumer-grade GPUs. While the 4B retriever incurs higher memory usage under our default configuration, this is mainly due to the absence

Table 3. Training efficiency analysis. We report the peak GPU memory usage (Mem) and total training time (Time) for a single experiment run across varying RAG pipelines, datasets, and retriever encoders. All models are fine-tuned with DP=1, TP=1, gradient checkpointing enabled, and no gradient accumulation.

RAG Pipeline	Qwen3-Embedding-4B			
	HotpotQA		NQ	
	Mem/GB	Time/h	Mem/GB	Time/h
ReAct Agent	61.70	3.10	31.42	3.14
Search-R1	65.82	1.36	41.72	1.12
RAG Pipeline	Qwen3-Embedding-0.6B			
	HotpotQA		NQ	
	Mem/GB	Time/h	Mem/GB	Time/h
ReAct Agent	26.84	2.40	17.70	2.00
Search-R1	31.48	2.42	16.47	1.51

of aggressive memory optimization; in practice, techniques such as gradient accumulation can be applied to reduce peak memory footprint without significantly increasing training time. Overall, these results demonstrate that HARR offers a favorable efficiency–effectiveness trade-off for retriever optimization in RAG systems.

## 6. Conclusion

This work targets the core mismatch between supervised retriever objectives and end-to-end QA in RAG, and proposes HARR: a history-aware RL framework that fine-tunes dense retrievers by (i) replacing deterministic top- $k$  with stochastic sampling to expose action distributions and (ii) encoding retrieval history in the state to reduce aliasing in multi-hop reasoning. Across datasets, RAG pipelines, and retriever scales, HARR consistently improves end-to-end RAG performance, positioning RL as a retriever-centric alternative that optimizes retrieval behavior directly for downstream task reward rather than proxy relevance signals.

## Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References

- Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. 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>.
- Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawska, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.
- Fajardo, A. and Emerson, D. fed-rag, March 2025. URL <https://github.com/VectorInstitute/fed-rag>.
- Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 6491–6501, 2024.
- Guan, X., Zeng, J., Meng, F., Xin, C., Lu, Y., Lin, H., Han, X., Sun, L., and Zhou, J. Deeprag: Thinking to retrieve step by step for large language models. *arXiv preprint arXiv:2502.01142*, 2025.
- Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.
- Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LORA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZeV KeeFYf9>.
- Izacard, G. and Grave, E. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pp. 874–880, 2021.
- Jiang, P., Lin, J., Cao, L., Tian, R., Kang, S., Wang, Z., Sun, J., and Han, J. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. *arXiv preprint arXiv:2503.00223*, 2025.
- Jiang, Z., Xu, F. F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., and Neubig, G. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023.
- Jin, B., Zeng, H., Yue, Z., Yoon, J., Arik, S., Wang, D., Zamani, H., and Han, J. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.
- Karpukhin, V., Oguz, B., Min, S., Lewis, P. S., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pp. 6769–6781, 2020.
- Ke, Z., Kong, W., Li, C., Zhang, M., Mei, Q., and Bendersky, M. Bridging the preference gap between retrievers and llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10438–10451, 2024.
- Kulkarni, M., Tangarajan, P., Kim, K., and Trivedi, A. Reinforcement learning for optimizing rag for domain chatbots. *arXiv preprint arXiv:2401.06800*, 2024.
- Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.
- Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.
- Li, S. and Ramakrishnan, N. Oreo: A plug-in context reconstructor to enhance retrieval-augmented generation. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR)*, pp. 238–253, 2025.
- Li, X., Mei, S., Liu, Z., Yan, Y., Wang, S., Yu, S., Zeng, Z., Chen, H., Yu, G., Liu, Z., et al. Rag-ddr: Optimizing retrieval-augmented generation using differentiable data rewards. *arXiv preprint arXiv:2410.13509*, 2024.

- Lin, X. V., Chen, X., Chen, M., Shi, W., Lomeli, M., James, R., Rodriguez, P., Kahn, J., Szilvassy, G., Lewis, M., et al. Ra-dit: Retrieval-augmented dual instruction tuning. In *ICLR*, 2024.
- Liu, J. LlamaIndex, 11 2022. URL [https://github.com/jerryjliu/llama\\_index](https://github.com/jerryjliu/llama_index).
- Liu, X., Li, D., Wen, T., Wan, J., Ling, G., Lv, F., Ou, D., and Tang, H. Taosearchemb: A multi-objective reinforcement learning framework for dense retrieval in taobao search. *arXiv preprint arXiv:2511.13885*, 2025.
- Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72, 2025.
- Nguyen, T., Chin, P., and Tai, Y.-W. Reward-rag: Enhancing rag with reward driven supervision. *arXiv preprint arXiv:2410.03780*, 2024.
- Niu, C., Wu, Y., Zhu, J., Xu, S., Shum, K., Zhong, R., Song, J., and Zhang, T. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10862–10878, 2024.
- Plackett, R. L. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.
- Qdrant Team. Qdrant: Vector search engine. <https://github.com/qdrant/qdrant>, 2023. Accessed: 2026-01.
- Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In Su, J., Duh, K., and Carreras, X. (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL <https://aclanthology.org/D16-1264/>.
- Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.
- Shao, Z., Gong, Y., Shen, Y., Huang, M., Duan, N., and Chen, W. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9248–9274, 2023.
- Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., and Yih, W.-t. 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, 2024.
- Singh, A., Ehtesham, A., Kumar, S., and Khoei, T. T. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*, 2025.
- Sun, H., Qiao, Z., Guo, J., Fan, X., Hou, Y., Jiang, Y., Xie, P., Zhang, Y., Huang, F., and Zhou, J. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*, 2025a.
- Sun, Y., Shen, J., Wang, Y., Chen, T., Wang, Z., Zhou, M., and Zhang, H. Improving data efficiency for llm reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay, 2025b. URL <https://arxiv.org/abs/2506.05316>.
- Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.
- von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., and Gallouédec, Q. TRL: Transformers Reinforcement Learning, 2020. URL <https://github.com/huggingface/trl>.
- Wang, L., Chen, H., Yang, N., Huang, X., Dou, Z., and Wei, F. Chain-of-retrieval augmented generation. *arXiv preprint arXiv:2501.14342*, 2025.
- Wang, Z., Teo, S., Ouyang, J., Xu, Y., and Shi, W. M-rag: Reinforcing large language model performance through retrieval-augmented generation with multiple partitions. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1966–1978, 2024.

- Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023a.
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL [https://openreview.net/forum?id=WE\\_vluYUL-X](https://openreview.net/forum?id=WE_vluYUL-X).
- Zamani, H. and Bendersky, M. Stochastic rag: End-to-end retrieval-augmented generation through expected utility maximization. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2641–2646, 2024.
- Zhou, H., Chen, Y., Guo, S., Yan, X., Lee, K. H., Wang, Z., Lee, K. Y., Zhang, G., Shao, K., Yang, L., et al. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.
- Zhou, J. and Chen, L. Optimizing retrieval for rag via reinforcement learning, 2026. URL <https://arxiv.org/abs/2510.24652>.