Agentic-R: Learning to Retrieve for Agentic Search

Wenhan Liu[1], **Xinyu Ma**[2], **Yutao Zhu**[1], **Yuchen Li**[2], **Daiting Shi**[2]
Dawei Yin[2] and **Zhicheng Dou**[1*]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Baidu Inc., Beijing, China
lwh@ruc.edu.cn, xinyuma2016@gmail.com, dou@ruc.edu.cn

Abstract

Agentic search has recently emerged as a powerful paradigm, where an agent interleaves multi-step reasoning with on-demand retrieval to solve complex questions. Despite its success, how to design a retriever for agentic search remains largely underexplored. Existing search agents typically rely on similarity-based retrievers, while similar passages are not always useful for final answer generation. In this paper, we propose a novel retriever training framework tailored for agentic search. Unlike retrievers designed for single-turn retrieval-augmented generation (RAG) that only rely on local passage utility, we propose to use both local query-passage relevance and global answer correctness to measure passage utility in a multi-turn agentic search. We further introduce an iterative training strategy, where the search agent and the retriever are optimized bidirectionally and iteratively. Different from RAG retrievers that are only trained once with fixed questions, our retriever is continuously improved using evolving and higher-quality queries from the agent. Extensive experiments on seven single-hop and multi-hop QA benchmarks demonstrate that our retriever, termed `Agentic-R`, consistently outperforms strong baselines across different search agents. Our codes are available at: https://github.com/8421BCD/Agentic-R.

1 Introduction

Retrieval-augmented generation (RAG) (Asai et al., 2023; Gao et al., 2023; Jin et al., 2025c) has become a widely adopted approach to address the knowledge limitations of large language models (LLMs) by retrieving external information to support generation. Recently, advances in large reasoning models (DeepSeek-AI et al., 2025) have given rise to a new paradigm known as *agentic search* (Jin et al., 2025b; Li et al., 2025). This approach extends traditional RAG from a single-turn retrieval to a multi-step "search-during-think" process (as shown in Figure 1). Unlike traditional single-turn RAG, agentic search enables an agent to decompose complex questions into a sequence of sub-queries, interleave reasoning with retrieval across multiple turns, and progressively gather evidence before producing a final answer. By integrating reasoning with retrieval, agentic search achieves superior performance compared to traditional single-turn RAG on challenging NLP tasks.

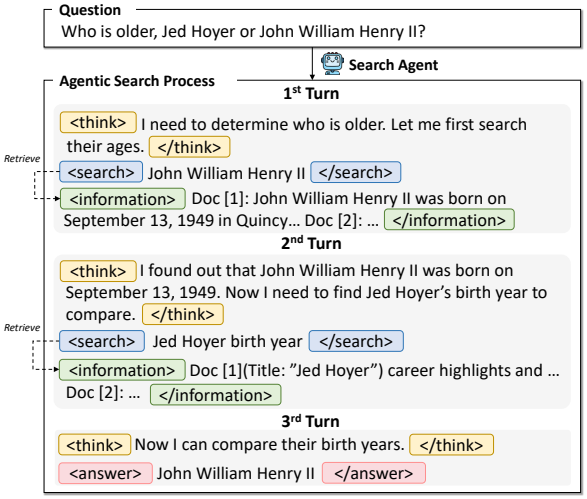

Figure 1: An example of agentic search.

Despite this progress, most existing research on agentic search focuses on designing more powerful agents (Shi et al., 2025; Jin et al., 2025b), with less attention paid to optimizing a critical component: **retriever**. This is a critical oversight because the quality of retrieved passages directly impacts both the training process and the final inference performance of the agent. Currently, most approaches employ *off-the-shelf* retrievers, such as E5 (Wang et al., 2022) or BGE (Xiao et al., 2024), which rely on semantic similarity. However, as demonstrated in previous studies (Xu et al., 2025; Wu et al., 2024), high semantic similarity does not guar-

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 15987–16005

antee that a passage is useful for generating answer.

To better align retrieval with downstream generation, prior studies have explored modeling passage utility for downstream generation and training utility-optimized retrievers for RAG (Shi et al., 2024; Xu et al., 2025; Zhang et al., 2024). These methods typically estimate utility by feeding retrieved passages into a generator and measuring downstream performance, such as generation likelihood or task-specific metrics. While effective for *single-turn* RAG, extending these approaches to *multi-turn* agentic search presents significant challenges: **First**, utility modeling in single-turn RAG relies on gold answers, which are available in existing datasets, to evaluate whether a passage addresses the question. In contrast, agentic search involves intermediate queries generated by the agent in each turn, but gold answers for these intermediate queries are not available, making it challenging to evaluate passage utility based on the gold answer. **Second**, passage utility in agentic search goes beyond local relevance. A passage relevant to current sub-query may contain misleading information that steers the subsequent reasoning process in the wrong direction, which ultimately leads to an incorrect final answer (Zeng et al., 2025; Cuconasu et al., 2024a,b). Thus, relying solely on local relevance to evaluate passage utility is insufficient.

Furthermore, existing retriever training methods for RAG (Shi et al., 2024; Xu et al., 2025) typically adopt a *one-way optimization* paradigm, where the retriever is trained only once based on fixed training queries (*i.e.*, user questions) and utility signals provided by a fixed generator. However, this approach is sub-optimal for training retrievers for agentic search. Unlike traditional RAG, the training queries for retrievers in agentic search are generated by the search agent itself. After the retriever is optimized, the search agent can further improve through reinforcement learning by interacting with this stronger retriever (Jin et al., 2025a). The improved search agent could generate new search trajectories with higher-quality queries, which can be leveraged to further optimize the retriever. Therefore, the optimization of the retriever and the search agent should be formulated as a *bidirectional and iterative* process.

In this paper, we propose the first retriever training framework designed specifically for agentic search. We introduce a passage utility modeling strategy that considers both the single-turn relevance and the correctness of the final answer. To evaluate single-turn relevance, we design an LLM-based listwise scoring approach that generates the relevance scores for multiple candidate passages of each intermediate query. To measure the correctness of the final answer, we evaluate whether the agent can derive the correct final answer when using a specific passage. Furthermore, we propose an iterative agent–retriever optimization framework that alternates between training the search agent and the retriever, allowing them to evolve together and finally resulting in a stronger retriever for agentic search. We conduct extensive experiments on seven benchmarks, including both multi-hop and single-hop question answering datasets. The results demonstrate that our method consistently outperforms strong baselines across different search agents. Further analysis shows that our `Agentic-R` also make the agent solve questions with fewer search queries.

Our contributions are threefold:

- We present the first retriever training framework specifically designed for search agents, addressing a critical yet underexplored component in existing agentic search systems.
- Based on the multi-turn nature of agentic search, we propose a novel passage utility modeling approach that considers both the relevance of the passage to current search query and its contribution to the correctness of the final answer.
- We introduce an iterative training framework, where the search agent and the retriever are optimized bidirectionally to progressively improve the retriever.

2 Related Work

Agentic Search Retrieval-Augmented Generation (RAG) significantly enhances Large Language Models (LLMs) by incorporating external knowledge sources (Asai et al., 2023; Gao et al., 2023; Ram et al., 2023). A fundamental challenge in RAG systems involves determining the optimal timing and approach for retrieval (Shao et al., 2023; Trivedi et al., 2023a; Huang et al., 2025; Liu et al., 2026). Previous studies explored prompt-based approaches that enable interleaved reasoning and retrieval (Trivedi et al., 2023b; Yao et al., 2023) and supervised fine-tuning (SFT) approaches that learn to call the search engine (Asai et al., 2024; Schick et al., 2023). Recently, reinforcement learning (RL) has emerged as a powerful and scalable alternative, enabling agents to learn complex, multi-turn

search strategies directly from task-outcome rewards without relying on extensive supervision (Jin et al., 2025b; Song et al., 2025; Chen et al., 2025; Zheng et al., 2025). This paradigm allows LLMs to dynamically decompose questions, formulate sequential queries, and retrieve information across turns. Despite significant progress in optimizing the search agents, the retriever, which is another critical component and significantly influences the agent's performance (Jin et al., 2025a), remains largely underexplored. In this paper, we propose to train a retriever tailored for agentic search.

Training Retrievers for Generation A key challenge in RAG systems is the gap between retrieval objectives (topical similarity) and generation needs (passage utility). To align retrievers with downstream tasks, recent work focuses on training utility-oriented retrievers using feedback from the generator. To obtain the passage utility, existing studies explore various approaches, such as using the generation likelihood of ground-truth answers (Shi et al., 2024; Xu et al., 2025; Izacard et al., 2023), downstream task metrics (Zamani and Bendersky, 2024; Wang et al., 2023a), and LLM-based annotation (Zhang et al., 2025). However, such passage utility modeling is limited to single-turn RAG, and the corresponding retriever training follows a one-way optimization paradigm which is sub-optimal for agentic search. In this paper, we propose a passage utility modeling mechanism and an agent–retriever iterative optimization framework tailored for training retrievers in agentic search.

3 Preliminary: Agentic Search

We consider a search agent based on a Large Language Model (LLM) that alternates between reasoning and external retrieval over multiple turns. An example is shown in Figure 1. In each turn i, the agent first produces a reasoning trace t_i, to analyze the current context and assess what information is still missing. Conditioned on this reasoning, the agent generates a search query q_i. After that, the retriever returns a set of passages D_i that are incorporated into the agent's context for subsequent reasoning.

This reasoning-retrieval cycle repeats across iterations, allowing the agent to progressively refine its understanding and gather relevant information. Once the agent determines that the accumulated information is sufficient, it terminates the retrieval process and produces the final answer. During the whole process, the reasoning steps, search query, retrieved passages and the final answer are explicitly enclosed within tags, such as `<think> </think>`, `<search> </search>`, `<information> </information>` and `<answer> </answer>`, respectively.

4 Methodology

4.1 Agentic-R

In this section, we first describe how we construct training data for Agentic-R by modeling passage utility in agentic search, and then introduce the corresponding training approach.

4.1.1 Training Data Construction

In this section, we propose to measure passage utility in agentic search from both local and global perspectives, and to use this utility to distinguish positive and negative passages for subsequent retriever training. Given an original user question Q and a retriever $\mathcal{R}$, we first let the search agent generate the whole trajectory, denoted as $\mathcal{T} = \{t_1, q_1, D_1, \ldots, t_i, q_i, D_i, \ldots, t_n, A\}$, where t_i, q_i and D_i represent the reasoning content, search query and top passages returned by retriever $\mathcal{R}$ at the i-th turn, and A is the final generated answer. For each query q_i, we first retrieve a candidate passage set $\mathcal{P}_i = \{p_{i,1}, \ldots, p_{i,j}, \ldots\}$ from the passage corpus using retriever $\mathcal{R}$, where $|\mathcal{P}_i| = 20$. Then, we propose to measure the utility of each candidate passage $p_{i,j}$ from two different perspectives: (1) local relevance and (2) global answer correctness, based on which we construct the positive and negative passages.

Local Relevance. Local relevance measures whether a candidate passage $p_{i,j}$ can answer the query q_i in i-th turn. Unlike prior single-turn RAG methods (Shi et al., 2024; Zamani and Bendersky, 2024) that rely on gold answers to measure the relevance, we do not have gold answers for queries of each turn. In this part, we design an LLM-based listwise scoring approach to evaluate the local query-passage relevance.

Specifically, we input the q_i and all its candidate passages $\mathcal{P}_i$ into a strong LLM, Qwen2.5-72B-Instruct[1] and instruct the LLM to assign a relevance score in the range $[0, 100]$ to each passage $p_{i,j}$, where higher scores indicate stronger relevance. We divide the score range into five intervals with

[1] https://huggingface.co/Qwen/Qwen2.5-72B-Instruct

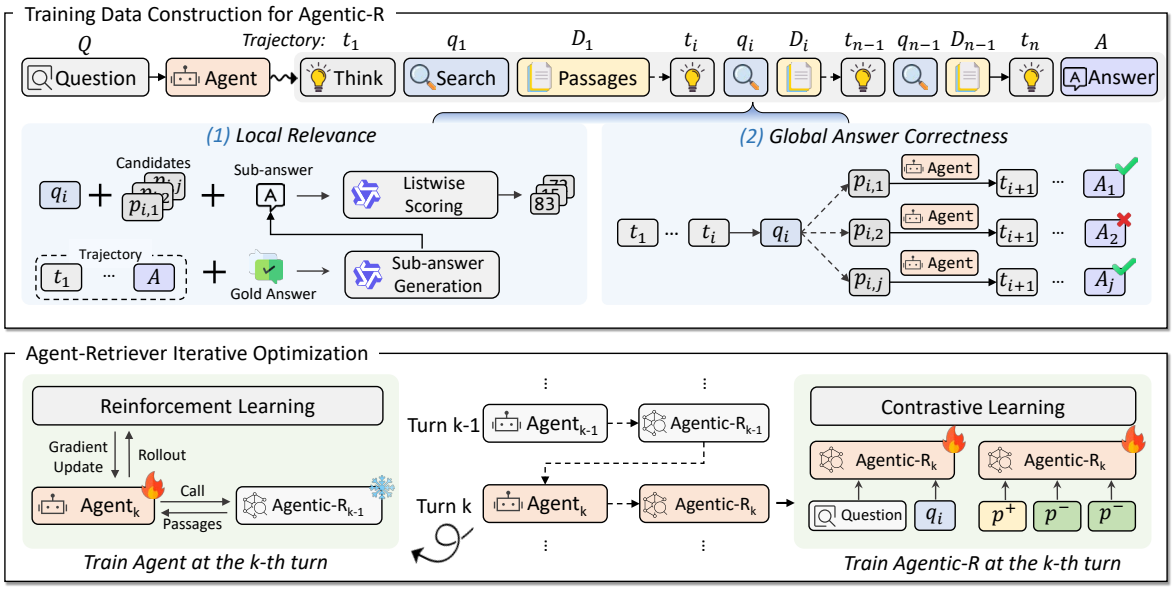

Figure 2: An overview of our training framework.

fine-grained scoring rules. For example, scores between 81 and 100 indicate that the passage directly addresses the search query or explicitly contains the required answer. The design of listwise scoring is inspired by listwise passage reranking in Information Retrieval (Sun et al., 2023; Liu et al., 2024, 2025a,b). Different from pointwise relevance scoring, which evaluates a single passage, listwise scoring compares multiple passages simultaneously and could yield more accurate relevance scores.

To help LLM better assess the relevance, besides q_i and $p_{i,j}$, we also incorporate a sub-answer into the relevance scoring prompt (shown in Figure 6). The sub-answer is generated by prompting the same LLM with the full search trajectory $\mathcal{T}$ and the final gold answer, and asking it to infer a correct sub-answer corresponding to each intermediate query q_i. If the LLM is unable to infer a reliable sub-answer, it will output "not sure", in which case no sub-answer will be provided in the listwise scoring prompt (prompt shown in Figure 7). The prompt for generating the sub-answer is shown in Figure 8. Formally, we define the local relevance score $\mathrm{LR}_{i,j}$ for passage $p_{i,j}$ in i-th search turn as:

$$\{\mathrm{LR}_{i,1}, \ldots, \mathrm{LR}_{i,j}\} = \mathrm{LLM}\big(q_i,\ \mathcal{P}_i,\ A_i^{\mathrm{sub}}\big)\,, \quad (1)$$

where $\mathcal{P}_i = \{p_{i,1}, \ldots, p_{i,j}\}$ denotes the candidate passages for query q_i and A_i^{sub} is an optional sub-answer for q_i. If no reliable sub-answer can be inferred, we set $A_i^{\mathrm{sub}} = \varnothing$.

Final Answer Correctness. As discussed in Section 1, passage with high local relevance does not guarantee that it could lead the search agent to generate a correct final answer. In this part, we explicitly incorporate final answer correctness as another passage utility.

Given query q_i, we concatenate each passage $p_{i,j}$ with q_i and let the agent continue generation conditioned on $p_{i,j}$ together with the preceding search trajectory, until a final answer $A_{i,j}$ is produced.

$$\{t_{i+1}, \ldots, A_{i,j}\} = \mathrm{Agent}(\{t_1, \ldots, q_i, p_{i,j}\})\,. \quad (2)$$

After that, we compute the exact match (EM) metric between the generated answer $A_{i,j}$ and the gold answer A^{gold} as the global answer correctness $\mathrm{GAC}_{i,j}$:

$$\mathrm{GAC}_{i,j} = \mathrm{EM}(A_{i,j}, A^{\mathrm{gold}}). \quad (3)$$

Although the final answer is influenced not only by $p_{i,j}$ but also by subsequent turns, all candidate passages $p_{i,j}$ are evaluated under the same preceding trajectory and search agent. Therefore, differences in final answer correctness can be a fair and valid metric to measure how $p_{i,j}$ steers the subsequent reasoning and generate the final answer.

Positive and Negative Passages Selection. After computing the local relevance $\mathrm{LR}_{i,j}$ (defined in Eq. 1) and global answer correctness $\mathrm{GAC}_{i,j}$ (defined in Eq. 3) for each candidate passage $p_{i,j}$ of query q_i, we rank all candidates based on two

sorting keys. Specifically, passages are first ranked by global answer correctness GAC (the first key) in descending order, and passages with identical GAC are further ordered by LR (the second key) in descending order. We prioritize global answer correctness since it directly reflects final task success: for example, a passage that leads to a correct final answer should be preferred over one that does not, regardless of their local relevance.

After ranking, we select the top-ranked passage as the positive passage, while passages ranked below are sampled as negatives. The total number of passages (positive plus negatives) is fixed to $N = 16$. To ensure positive quality, we require its $\mathrm{GAC}_{i,j} = 1$ and $\mathrm{LR}_{i,j} \geq 60$. If no passage satisfies both conditions, the training instance for query q_i is discarded. For all q_i in trajectory $\mathcal{T}$, we use the same method to construct training data.

4.1.2 Training Approach

After constructing positive and negative passages, we use contrastive learning (Karpukhin et al., 2020a) to train our retriever model `Agentic-R`.

Training Input. When modeling passage utility, we consider not only the relevance of a passage to the current query q_i, but also its contribution to answering the original question Q and generating the final answer. Thus, besides q_i, we also incorporate the original question Q as auxiliary information and concatenate them together as the input x_i to the query encoder:

$$x_i = Q \ [\mathrm{SEP}] \ q_i, \tag{4}$$

where [SEP] denotes a separator token. Note that we do not include queries from previous turns as input. This is because, in agentic search, agent queries are typically self-contained and do not involve anaphoric references (*e.g.*, terms such as "it") that require contextual disambiguation. This differs from conversational search (Mao et al., 2022) or session search (Wang et al., 2023b) tasks, where previous queries are necessary to understand the current query. Empirically, we find that incorporating previous queries introduces retrieval noise and degrades final performance. Detailed analysis is provided in Appendix C.2.

Training Loss. In addition to the sampled negatives described in Section 4.1.1, we also incorporate in-batch negatives and cross-device negatives (Qu et al., 2021; Zhang et al., 2024) to expand the scale of negative passages. Consequently, our method yields $(B \times G \times N - 1)$ negative samples in total, where B is the batch size, G is the number of GPU devices, and N is the number of constructed samples per query. For each training instance x_i, the contrastive learning loss is defined as:

$$\mathcal{L} = -\log \frac{\exp\big(\mathrm{sim}(x_i, z^+)\big)}{\sum_{z \in \mathcal{Z}} \exp\big(\mathrm{sim}(x_i, z)\big)}, \tag{5}$$

where z^+ denotes the positive passage embedding, $\mathcal{Z}$ includes the sampled positives and negatives as well as in-batch and cross-device negatives, and $\mathrm{sim}(\cdot, \cdot)$ denotes the embedding similarity.

4.2 Agent-Retriever Iterative Optimization

As discussed in Section 1, the optimized `Agentic-R` could further be used to improve the search agent by providing more relevant passages during the RL process and the improved search agent could generate new training queries to further train the retriever, which is a bidirectional and iterative process. Motivated by this, we propose an iterative training framework that iteratively optimizes the search agent and `Agentic-R`. We first describe the training of the search agent, followed by the iterative optimization procedure.

Agent Training. We apply the same RL training approach as Search-R1 (Jin et al., 2025b) for our search agent training. Specifically, we adopt the RL algorithm PPO (Schulman et al., 2017) to train the agent. During RL training, the agent generates trajectories by performing multiple turns of interactions with the retriever until generating the final answer. Then, we use exact match (EM) between the generated answer and the gold answer as the final reward. The prompt we used for training is shown in Figure 5. Additional training details are provided in Appendix B.

Iterative Optimization. We adopt an iterative optimization mechanism to iteratively train the search agent and the retriever. At iteration i, we first optimize the search agent Agent_i, based on our retriever from the previous iteration $\texttt{Agentic-R}_{i-1}$. Note that for the first iteration, $\texttt{Agentic-R}_0$ is initialized by embedding model E5. During agent's training, the retriever is kept fixed and treated as part of the RL environment. After training Agent_i, we use it to generate training queries, retrieve candidate passages using $\texttt{Agentic-R}_{i-1}$ and construct positive and negative passages based on the passage utility modeling described in Section 4.1.1,

Algorithm 1 Iterative Optimization Process

Input: Training questions $\mathcal{Q}$; initial retriever $\texttt{Agentic-R}_0$; number of iterations K
Output: Optimized Agent_K and $\texttt{Agentic-R}_K$
for $i = 1$ **to** K **do**
  Agent Training:
  Train the search agent Agent_i using PPO by interacting with retriever $\texttt{Agentic-R}_{i-1}$
  Retriever Training:
  (1) Use Agent_i to generate trajectories of $\mathcal{Q}$.
  (2) Retrieve candidate passages using $\mathcal{R}_{i-1}$ and construct positive and negative passage following Section 4.1.1.
  (3) Train $\texttt{Agentic-R}_i$ following Section 4.1.2.
end
return Agent_K, $\texttt{Agentic-R}_K$

which will be used to train $\texttt{Agentic-R}_i$. The overall procedure is summarized in Algorithm 1.

5 Experiment

5.1 Settings

Evaluation Datasets. We conduct experiments on seven question answering (QA) benchmarks covering both multi-hop and single-hop datasets. For multi-hop QA, we evaluate on HotpotQA (Yang et al., 2018), 2WikiMultihopQA (2Wiki) (Ho et al., 2020), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). The single-hop QA datasets include Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023). Exact match (EM) is used as the evaluation metric.

Baselines. We compare our approach with two categories of retriever baselines:

General-purpose embedding models. These retrievers are trained as universal text encoders to support a broad range of retrieval tasks and are widely adopted in agentic search systems. We include BGE (Xiao et al., 2024) and E5 (Wang et al., 2022) as representative baselines, both of which have demonstrated strong performance on standard retrieval benchmarks and are widely used as off-the-shelf retrievers in agentic search.

RAG-specific retrievers. These methods optimize retrievers using feedback from downstream generation in single-turn RAG settings. We compare against LLM-Embedder (Zhang et al., 2024), SCARLet (Xu et al., 2025), and REPLUG (Shi et al., 2024), which leverage generation likelihood or task-level signals to train utility-aware retrievers. Additional details of the baseline retrievers are provided in Appendix A.

Implementation Details. Throughout the iterative training process, we construct training data using the training splits of TriviaQA and HotpotQA. For agent training, we initialize the search agent with Qwen2.5-7B-Base. During training and inference, the maximum search turns n and passage retrieval number m of our agent are set as 5 and 3, respectively. For retriever training, we initialize our Agentic-R with E5[2]. We use the December 2018 Wikipedia dump (Karpukhin et al., 2020b) as the retrieval corpus for all experiments. We set the iteration number K of agent–retriever optimization as 2, as additional iterations do not yield further improvements (detailed analysis is provided in Section 5.5). Additional implementation details are provided in Appendix B.

5.2 Overall Performance

We evaluate Agentic-R's performance on three search agents: our trained search agent, as well as two other search agents, R1-Searcher (Song et al., 2025) and SimpleDeepSearcher (Sun et al., 2025), to evaluate the generalization across different search agents. The results are reported in Table 1. From the results, we have the following observations:

(1) Agentic-R consistently achieves the best average EM score across all three search agents. Agentic-R outperforms the second-best baseline by about 3.2 points on our trained search agent and by roughly 2 points on R1-Searcher and SimpleDeepSearcher. This demonstrates that Agentic-R not only performs well on our in-domain search agent, but also generalizes effectively to other search agents.

(2) Agentic-R yields larger improvements on multi-hop QA than on single-hop QA. For example, based on our search agent, the average performance gap between Agentic-R and REPLUG on multi-hop QA datasets is about 3 points, higher than the 2 points on single-hop QA datasets. This indicates that Agentic-R is particularly effective for search agents in multi-hop scenarios.

(3) RAG-specific retrievers do not consistently outperform general-purpose retrievers in agentic search. For example, LLM-Embedder and SCARLet are often inferior to E5 across all three

[2] https://huggingface.co/intfloat/e5-base-v2

Methods	Multi-Hop QA				General QA			
	HotpotQA	2Wiki	Musique	Bamboogle	NQ	TriviaQA	PopQA	Avg.
Our Search Agent (in-domain)								
LLM-Embedder	39.35	39.75	17.33	36.00	41.32	62.33	42.69	39.82
BGE	41.71	39.30	16.21	38.40	40.36	63.66	41.92	40.22
SCARLet	42.34	39.35	16.38	40.80	40.60	63.46	42.04	40.71
E5	41.68	40.02	17.74	44.00	42.18	64.72	41.82	41.74
REPLUG	42.63	40.23	18.90	41.60	41.46	65.78	41.85	41.78
Agentic-R	**45.82**	**45.30**	**20.27**	**48.00**	**42.43**	**69.02**	**44.14**	**45.00**
R1-Searcher (out-of-domain)								
LLM-Embedder	41.39	45.72	18.36	33.60	38.80	56.29	40.05	39.17
BGE	44.36	45.86	18.36	34.40	36.34	57.19	38.95	39.35
SCARLet	44.11	45.94	18.53	35.19	36.34	57.64	38.73	39.50
E5	43.56	46.33	21.39	**44.00**	39.39	58.69	38.31	41.67
REPLUG	40.68	39.66	18.32	41.60	**40.94**	62.39	42.11	40.81
Agentic-R	**47.68**	**49.07**	**22.54**	41.60	39.63	**62.52**	**42.43**	**43.64**
SimpleDeepSearcher (out-of-domain)								
LLM-Embedder	35.01	32.52	13.90	40.80	33.62	59.59	37.72	36.17
BGE	37.31	33.95	14.06	**42.40**	32.82	60.73	36.83	36.87
SCARLet	37.39	33.44	14.39	35.19	32.38	60.81	36.93	35.79
E5	36.61	34.01	15.55	40.80	33.65	62.48	37.09	37.17
REPLUG	37.39	34.08	15.80	41.60	33.71	63.59	37.68	37.69
Agentic-R	**39.75**	**38.04**	**17.29**	41.60	**34.62**	**65.61**	**39.09**	**39.43**

Table 1: The performance of retrievers on our trained search agent (in-domain) and two other search agents (out-of-domain). Both our search agent and Agentic-R are trained after two iterations. The top two rerankers are highlighted in **bold** and underlined.

agents. This may be because: (1) the passage utility derived from single-turn RAGs is not directly applicable to multi-turn agentic search; (2) RAG-specific retrievers are trained on user questions instead of agent-generated queries, resulting in a distribution gap of training queries.

5.3 Ablation Study

We conduct ablation studies to examine the impact of two key components in our framework: (1) agent–retriever iterative optimization and (2) passage utility modeling. Results are summarized in Table 2. Here, "Agent_2 + Agentic-R_2" denotes our final agent and retriever trained after two iterations.

Effect of Iterative Optimization. We first analyze the effect of iterative training on both the retriever and the search agent. Replacing Agentic-R_2 with the retriever Agentic-R_1 obtained after the first iteration leads to an average performance drop of about 0.9 points. This indicates that the second iteration further improves the retriever by leveraging higher-quality agent-generated queries. Similar trends are also observed when using R1-Searcher (see Table 4). We further replace Agent_2 with the agent trained after the first iteration (*i.e.*, Agent_1), which results in an additional drop of about 1.9 points. This suggests that compared with E5, optimized Agentic-R_1 can better improve the RL training of the search agent.

Effect of Passage Utility Modeling. We next study the effectiveness of different utility signals, global answer correctness (GAC) and local relevance (LR), based on combination "Agent_1 + Agentic-R_1". Removing GAC ("w/o GAC") or LR ("w/o LR") from the utility modeling causes clear performance degradation, with average drops of about 1.1 and 1.7 points, respectively. This confirms that both signals are essential for identifying positive passages for retriever training. The larger performance drop of "w/o LR" indicates that local relevance plays a more important role in measuring passage utility. We also ablate the use of the original question in the retriever input ("w/o Question"). The 0.7-points performance drop suggests that the original question helps the retriever better assess whether a passage could contribute to answering the question and generating the correct final answer.

5.4 Search Turns Analysis

In this section, we analyze the number of search turns taken by the agent to examine whether our

Methods	Multi-Hop QA				General QA			
	HotpotQA	2Wiki	Musique	Bamboogle	NQ	TriviaQA	PopQA	Avg.
Agent_2 + $\texttt{Agentic-R}_2$	**45.82**	**45.30**	**20.27**	**48.00**	**42.43**	**69.02**	44.14	**45.00**
• Agent_2 + $\texttt{Agentic-R}_1$	44.88	44.31	19.23	44.80	42.16	68.59	**44.99**	44.14
• Agent_1 + $\texttt{Agentic-R}_1$	40.44	44.49	16.63	44.00	39.69	65.80	44.75	42.26
• w/o GAC	38.91	43.42	16.71	43.20	39.19	63.91	42.66	41.14
• w/o LR	38.73	41.34	15.80	40.00	40.36	64.12	43.86	40.60
• w/o Question	39.52	44.02	16.13	43.20	39.30	64.96	43.89	41.57

Table 2: Ablation studies over key components.

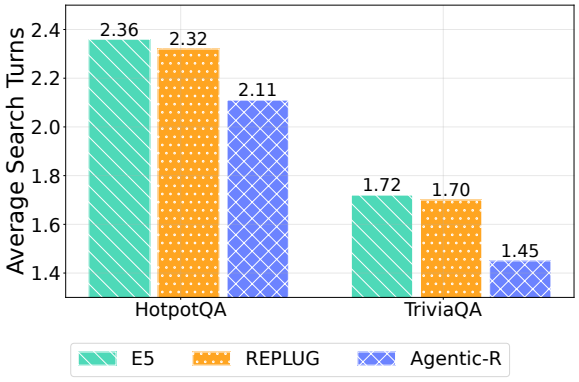

Figure 3: The comparison of search turns on different retrievers.

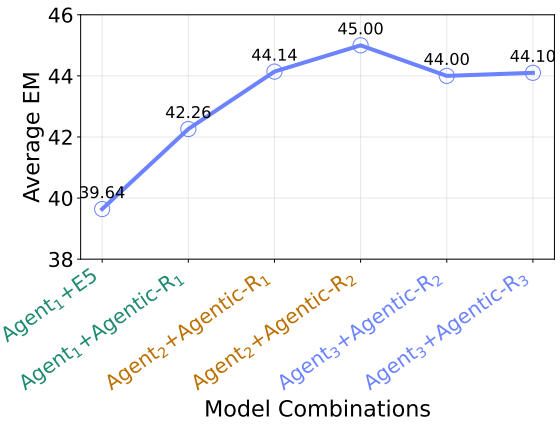

Figure 4: Effect of different iteration numbers (K) in agent–retriever optimization. Different colors on the x-axis indicate different iteration number.

`Agentic-R` can also make the agent generate the answer with fewer search queries. We conduct experiments on HotpotQA and TriviaQA using our search agent and `Agentic-R` trained after two iterations, and compare against two baselines, E5 and REPLUG. The results are shown in Figure 3. From the results, `Agentic-R` consistently reduces the number of search turns compared to E5 and REPLUG on both datasets. For example, `Agentic-R` reduces the average search turns by approximately 10% on HotpotQA and 15% on TriviaQA compared to REPLUG. These results indicate that `Agentic-R` enables the agent to acquire more useful information per retrieval, allowing it to solve questions with fewer search turns.

5.5 Different Iteration Number K

In our main experiments, we set the round K of agent–retriever iterative optimization as 2. To examine whether the iterative training process converges, we further perform a third iteration and compare the performance after each round. We report the average EM score of all 7 datasets used in Table 1 and show the results in Figure 4.

From the results, we observe consistent performance improvements during the first two iterations. Specifically, Agent_1 + $\texttt{Agentic-R}_1$ outperforms Agent_1 + E5 by approximately 2.6 points, and Agent_2 + $\texttt{Agentic-R}_1$ further improves over Agent_1 + $\texttt{Agentic-R}_1$ by about 1.9 points. However, no further improvement is observed in the third iteration, where performance slightly degrades. This indicates that the iterative optimization converges after two iterations, and additional iterations do not provide further benefits.

6 Conclusion

In this work, we propose a retriever training framework specifically designed for agentic search. We first introduce a passage utility modeling strategy that captures both local relevance and global answer correctness, providing effective supervision for retriever training for multi-turn agentic search. Furthermore, we develop an iterative agent–retriever training framework that continuously enhances the retriever's ability based on higher-quality agent-generated queries. We conduct extensive experiments on multi-hop and single-hop QA benchmarks. Results demonstrate the superior performance of our retriever `Agentic-R`. Further anal-

yses show that Agentic-R also reduces the number of search turns required by the agent, improving the efficiency of agentic search.

Limitations

Despite the strong empirical results, this work has several limitations.

First, our evaluation is conducted on widely used single-hop and multi-hop question answering benchmarks. While these datasets cover a range of reasoning difficulties, they do not fully represent more challenging settings that require deeper or more abstract reasoning, such as expert-level or scientific reasoning tasks such as GPQA (Rein et al., 2023). Future work could extend our framework to broader tasks.

Second, due to computational and memory constraints, we conduct our experiments with moderately sized search agents and retriever backbones. Although our results across different backbones suggest favorable scaling trends, validating the effectiveness of our Agentic-R in larger agents and retriever backbones remains an important direction for future work.

Acknowledgments

The work was supported by National Natural Science Foundation of China No. 62272467, and was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.

References

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 41–46. Association for Computational Linguistics.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025. Research: Learning to reason with search for llms via reinforcement learning. *CoRR*, abs/2503.19470.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024a. The power of noise: Redefining retrieval for RAG systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 719–729. ACM.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024b. Rethinking relevance: How noise and distractors impact retrieval-augmented generation. In *Proceedings of the 14th Italian Information Retrieval Workshop, Udine, Italy, September 5-6, 2024*, volume 3802 of *CEUR Workshop Proceedings*, pages 95–98. CEUR-WS.org.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

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

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, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Lisheng Huang, Yichen Liu, Jinhao Jiang, Rongxiang Zhang, Jiahao Yan, Junyi Li, and Wayne Xin Zhao. 2025. Manusearch: Democratizing deep search in large language models with a transparent and open multi-agent framework. *CoRR*, abs/2505.18105.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43.

Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan Ö. Arik, and Jiawei Han. 2025a. An empirical study on reinforcement learning for reasoning-search interleaved LLM agents. *CoRR*, abs/2505.15117.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025b. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025c. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pages 737–740. ACM.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020a. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020b. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

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. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438, Suzhou, China. Association for Computational Linguistics.

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. 2025a. Reasonrank: Empowering passage ranking with strong reasoning ability. *arXiv preprint arXiv:2508.07050*.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Lixin Su, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2025b. CoRanking: Collaborative ranking with small and large ranking agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5098–5110, Suzhou, China. Association for Computational Linguistics.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Ziliang Zhao, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2024. Sliding windows are not the end: Exploring full ranking with long-context large language models. *CoRR*, abs/2412.14574.

Wenhan Liu, Yutao Zhu, Zhicheng Dou, and Yujia Zhou. 2026. Demorank: Selecting effective demonstrations for large language models in ranking task. *ACM Transactions on Information Systems*, 44(3):1–25.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9802–9822. Association for Computational Linguistics.

Kelong Mao, Zhicheng Dou, and Hongjin Qian. 2022. Curriculum contrastive context denoising for few-shot conversational dense retrieval. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 176–186. ACM.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 5687–5711. Association for Computational Linguistics.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and

Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5835–5847. Association for Computational Linguistics.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Trans. Assoc. Comput. Linguistics*, 11:1316–1331.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9248–9274. Association for Computational Linguistics.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. 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), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 8371–8384. Association for Computational Linguistics.

Yaorui Shi, Sihang Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. 2025. Search and refine during think: Autonomous retrieval-augmented reasoning of llms. *CoRR*, abs/2505.11277.

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. *CoRR*, abs/2503.05592.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen. 2025. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *CoRR*, abs/2505.16834.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14918–14937. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023a. 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), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10014–10037. Association for Computational Linguistics.

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), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10014–10037. Association for Computational Linguistics.

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. *CoRR*, abs/2212.03533.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023a. Simlm: Pre-training with representation bottleneck for dense passage retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2244–2258. Association for Computational Linguistics.

Shuting Wang, Zhicheng Dou, and Yutao Zhu. 2023b. Heterogeneous graph-based context-aware document ranking. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pages 724–732. ACM.

Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. 2024. How easily do irrelevant inputs skew the responses of large language models? *CoRR*, abs/2404.03302.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 641–649. ACM.

Yilong Xu, Jinhua Gao, Xiaoming Yu, Yuanhai Xue, Baolong Bi, Huawei Shen, and Xueqi Cheng. 2025. Training a utility-based retriever through shared context attribution for retrieval-augmented language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 629–648, Suzhou, China. Association for Computational Linguistics.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Hamed Zamani and Michael Bendersky. 2024. 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, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2641–2646. ACM.

Linda Zeng, Rithwik Gupta, Divij Motwani, Diji Yang, and Yi Zhang. 2025. Worse than zero-shot? A fact-checking dataset for evaluating the robustness of RAG against misleading retrievals. *CoRR*, abs/2502.16101.

Hengran Zhang, Keping Bi, Jiafeng Guo, Jiaming Zhang, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. 2025. Llm-specific utility: A new perspective for retrieval-augmented generation. *CoRR*, abs/2510.11358.

Peitian Zhang, Zheng Liu, Shitao Xiao, Zhicheng Dou, and Jian-Yun Nie. 2024. A multi-task embedder for retrieval augmented llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3537–3553. Association for Computational Linguistics.

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

A Baselines

We compare our approach with two categories of retriever baselines: general-purpose embedding models and RAG-specific retrievers.

General-purpose embedding models. These retrievers are trained as universal text encoders and are widely adopted as off-the-shelf components in agentic search systems. We include **E5** (Wang et al., 2022) and **BGE** (Xiao et al., 2024) as representative baselines. For E5, we use the `e5-base-v2`, and for BGE, we adopt `bge-base-en-v1.5`, both of which demonstrate strong performance on standard retrieval benchmarks.

RAG-specific retrievers. These methods train retrievers using generation feedback in single-turn RAG settings, aiming to capture passage utility beyond semantic similarity.

LLM-Embedder (Zhang et al., 2024) trains a unified embedding model with supervision from LLM-generated feedback to support retrieval-augmented generation across tasks. We use the publicly available checkpoint[3].

SCARLet (Xu et al., 2025) estimates passage utility via perturbation-based attribution, explicitly modeling inter-passage interactions to better reflect a passage's contribution to downstream generation. We use the authors' publicly released checkpoint for all experiments.

REPLUG (Shi et al., 2024) defines passage utility by measuring how well each passage supports generating the ground-truth answer, using answer likelihood as the supervision signal. As the official checkpoint is not released, we re-implement REPLUG and train it on the same datasets (HotpotQA and TriviaQA) with the same generator backbone, `Qwen2.5-7B-Base`, to ensure fair comparison.

B Implementation Details

In this part, we will introduce the implementation details of our paper.

Retriever Training. We initialize our retriever from e5-base-v2 and train it using contrastive learning with agent-generated supervision. Training data are constructed following the passage utility modeling procedure described in Section 4.1.1. For each query, we use $N = 16$ training passages in total, including one positive passage and multiple negatives.

We adopt mean pooling over token embeddings and apply ℓ_2 normalization. The retriever is trained for 2 epochs with a learning rate of 2×10^{-5} and a per-device batch size of 32. We use a temperature of 0.01 in the contrastive loss and enable in-batch and cross-device negatives, resulting in a large and diverse negative set. The maximum input length is set to 512 tokens for both queries and passages. All retriever models are trained on the December 2018 Wikipedia dump, which serves as the retrieval corpus.

Agent Training. We train the search agent following the same reinforcement learning approach of Search-R1 (Jin et al., 2025b). Specifically, the policy LLM is initialized from Qwen2.5-7B-Base, and the agent is optimized using Proximal Policy Optimization (PPO) (Schulman et al., 2017). Unlike standard PPO settings, where all tokens are generated by the policy model, the rollout sequence in agentic search contains both LLM-generated tokens and retrieved tokens from external passages. We therefore apply token-level loss masking to ensure that policy optimization is performed only on LLM-generated tokens, while retrieved tokens are excluded from gradient updates. This design allows the agent to learn effective reasoning and search behaviors while preventing spurious updates on retrieved tokens, resulting in more stable training for search-augmented generation. Refer to Search-R1 (Jin et al., 2025b) for more details.

During RL training, the learning rate is set to 1×10^{-6} for the policy model and 1×10^{-5} for the value model. We train the agent for 500 optimization steps, using Generalized Advantage Estimation (GAE) with $\lambda = 1$ and $\gamma = 1$. The total batch size is 512, with a mini-batch size of 256 and a micro-batch size of 64. The maximum sequence length is 4,096 tokens, including up to 500 tokens for the generated response and up to 500 tokens for retrieved passages.

To reduce memory consumption, we enable gradient checkpointing and employ Fully Sharded Data Parallel (FSDP) with CPU offloading. Model checkpoints are saved every 50 steps. If training instability is observed, we select the most recent stable checkpoint based on the reward curve; otherwise, the final checkpoint is used for evaluation.

Iterative Training Setup. During iterative agent–retriever optimization, we alternate between training the search agent and updating the retriever. At each iteration, the retriever is fixed while train-

[3] https://huggingface.co/BAAI/llm-embedder

Methods	Multi-Hop QA				General QA			Avg.
	HotpotQA	2Wiki	Musique	Bamboogle	NQ	TriviaQA	PopQA	
Our Search Agent (in-domain)								
LLM-Embedder	35.92	39.79	14.97	38.40	39.58	59.01	40.93	38.37
BGE-base	37.51	39.86	14.27	40.00	37.39	59.94	39.53	38.36
SCARLet	37.82	39.71	13.98	39.20	37.31	59.98	39.55	38.22
E5-base	36.32	42.04	16.13	36.80	39.77	64.35	42.09	39.64
REPLUG	38.05	40.78	16.38	37.60	39.08	63.00	39.72	39.23
E5-large	38.69	40.53	16.88	**44.00**	**41.10**	63.49	40.98	40.81
Agentic-R$_{\text{E5-base}}$	40.44	**44.49**	16.63	**44.00**	39.69	65.80	**44.75**	42.26
Agentic-R$_{\text{BGE-base}}$	40.28	43.82	15.97	42.40	38.39	64.00	43.30	41.17
Agentic-R$_{\text{E5-large}}$	**41.49**	44.40	**18.08**	**44.00**	39.97	**66.26**	44.40	**42.66**
SimpleDeepSearcher (out-of-domain)								
LLM-Embedder	35.01	32.52	13.90	40.80	33.62	59.59	37.72	36.17
BGE-base	37.31	33.95	14.06	42.40	32.82	60.73	36.83	36.87
SCARLet	37.39	33.44	14.39	35.19	32.38	60.81	36.93	35.79
E5-base	36.61	34.01	15.55	40.80	33.65	62.48	37.09	37.17
REPLUG	37.39	34.08	15.80	41.60	33.71	63.59	37.68	37.69
E5-large	37.59	33.20	15.72	43.20	**34.90**	63.58	37.63	37.97
Agentic-R$_{\text{E5-base}}$	39.98	37.43	15.18	44.80	33.98	65.16	**39.46**	39.43
Agentic-R$_{\text{BGE-base}}$	38.83	35.92	15.26	40.80	33.21	64.31	38.38	38.10
Agentic-R$_{\text{E5-large}}$	**40.25**	**38.12**	**16.09**	**46.40**	34.65	**65.58**	39.27	**40.05**

Table 3: The performance of Agentic-R using different backbones. Our search agent and Agentic-R are trained for one iteration.

Methods	Multi-Hop QA				General QA			Avg.
	HotpotQA	2Wiki	Musique	Bamboogle	NQ	TriviaQA	PopQA	
R1-Searcher + Agentic-R$_2$	**47.68**	**49.07**	**22.54**	**41.60**	39.63	**62.52**	42.43	**43.64**
R1-Searcher + Agentic-R$_1$	46.86	48.64	21.39	35.19	**39.94**	62.33	**43.07**	42.49

Table 4: The performance comparison between Agentic-R$_1$ and Agentic-R$_2$ based on R1-Searcher.

ing the agent, and the trained agent is then used to generate new trajectories for retriever training. We perform two iterations in total, as additional iterations do not yield further improvements (see Section 5.3). All experiments are conducted on a single node with 8*A800 80G GPUs.

C Additional Experiments

C.1 Different Backbone of Agentic-R

During our previous experiments, we used E5-base as the backbone of Agentic-R. In this section, we evaluate whether our retriever training framework generalizes to other backbone models by training Agentic-R on BGE-base[4] and E5-large[5]. For simplicity, both the search agent and the retriever are trained for a single iteration in this experiment.

As shown in Table 3, Agentic-R consistently outperforms all baseline retrievers across all tested backbones and search agents, demonstrating that our method generalizes well beyond a specific embedding model. Moreover, Agentic-R yields substantial improvements over its corresponding backbone retriever; for example, under our search agent, Agentic-R$_{\text{BGE-base}}$ improves upon BGE-base by approximately 2.8 average EM points. We further observe a clear scaling trend with respect to retriever capacity: the larger backbone E5-large consistently outperforms E5-base. A similar trend is also observed between Agentic-R$_{\text{E5-base}}$ and Agentic-R$_{\text{E5-large}}$.

C.2 Training Input

In the main experiments, we construct the retriever input using only the original question Q and the current-turn query q_i. In this part, we further investigate whether incorporating historical queries from previous turns can benefit retriever training.

Specifically, we concatenate the original question Q, all historical queries $\{q_1, \ldots, q_{i-1}\}$, and the current query q_i with separator tokens as the

[4] https://huggingface.co/BAAI/bge-base-en-v1.5
[5] https://huggingface.co/intfloat/e5-large-v2

Methods	Multi-Hop QA				General QA			Avg.
	HotpotQA	2Wiki	Musique	Bamboogle	NQ	TriviaQA	PopQA	
Our Search Agent								
E5	36.32	42.04	16.13	36.80	39.77	64.35	42.09	39.64
REPLUG	38.05	40.78	16.38	37.60	39.08	63.00	39.72	39.23
Agentic-R	40.44	44.49	16.63	**44.00**	39.69	**65.80**	44.75	**42.26**
Agentic-R (w/ historical queries)	**40.47**	**44.65**	**17.04**	38.40	**40.02**	65.67	**44.87**	41.59
R1-Searcher								
E5	43.56	46.33	**21.39**	**44.00**	39.39	58.69	38.31	41.67
REPLUG	40.68	39.66	18.32	41.60	**40.94**	**62.39**	42.11	40.81
Agentic-R	**46.86**	**48.64**	**21.39**	35.19	39.94	62.33	43.07	**42.49**
Agentic-R (w/ historical queries)	43.13	42.95	15.30	31.20	39.63	62.18	**43.17**	39.65

Table 5: The performance of Agentic-R when using historical queries as additional training input.

retriever input:

$$x_i = Q \text{ [SEP] } q_1 \text{ [SEP] } \cdots q_{i-1} \text{ [SEP] } q_i. \quad (6)$$

The same input format is also used at inference time. We conduct this ablation using a single iteration of agent–retriever training and compare it with the default setting that excludes historical queries.

The results are shown in Table 5. We observe that incorporating historical queries (*i.e.*, Agentic-R (w/ historical queries)) degrades average performance for Agentic-R under both the in-domain search agent and the out-of-domain R1-Searcher. The performance drop is particularly obvious under R1-Searcher, where Agentic-R with historical queries even underperforms general-purpose retrievers such as E5 and REPLUG.

This behavior can be attributed to the nature of agentic search queries. In agentic search, the current query q_i is typically generated with a clear and explicit intent, and does not involve anaphoric references or contextual ambiguity. As a result, historical queries provide limited additional information, while introducing extra noise that may distract the retriever from the core intent of the current query. These results further justify our design choice of using only (Q, q_i) as the retriever input in Agentic-R.

D Case Study

To better understand how Agentic-R improves the performance of the search agent, we conduct a case study comparing the trajectories produced by E5 and Agentic-R based on our trained agent. Note that both Agentic-R and our trained agent are trained for 2 iterations. The result is shown in Table 6.

E Use of AI Assistants

We use ChatGPT to improve the presentations of this paper.[6]

[6] https://chatgpt.com/

Table 6: Case study comparing E5 and Agentic-R on a multi-hop query of HotpotQA. The red-highlighted text contains irrelevant information that misled the generation of the correct answer, while the green-highlighted text represents key evidence that contributed to generating the correct answer.

Query

Who founded the honky tonk that is at the center of John Travolta's third major acting role?

Gold Answer

Mickey Gilley

E5 (Failed)

<think> John Travolta is an actor, so I will search for his filmography to find his third major acting role. </think>
<search> John Travolta third major acting role </search>
<information>
Doc 1(Title: John Travolta) John Travolta first became known in the 1970s, after appearing on the television series Welcome Back, Kotter (1975–1979) and starring in the box office successes Saturday Night Fever (1977) and Grease (1978). His acting career declined through the 1980s, but enjoyed a resurgence in the 1990s with his role in Pulp Fiction (1994), and he has since starred in films such as Get Shorty, Broken Arrow, Face/Off, Swordfish, Be Cool, Wild Hogs, Hairspray, ...
</information>

<think> Assuming the movie is Get Shorty, I now search for the honky tonk. </think>
<search> Get Shorty honky tonk </search>
<information>
Doc1: ... Doc2 (Honky-tonk): A honky-tonk (also called honkatonk, honkey-tonk, or tonk) is both a bar that provides country music for the entertainment of its patrons and the style of music played in such establishments. Bars of this kind are common in the South and Southwest United States...
</information>

<think> Search for the founder of the assumed honky tonk. </think>
<search> The Big Enchilada honky tonk founder </search>
<information>
Doc 1(Title: Honky-tonk) Honky-tonk: A honky-tonk (also called honkatonk, honkey-tonk, or tonk) is both a bar that provides country music for the entertainment of its patrons and the style of music played in such establishments...
</information>

<answer> Eddie Wilson </answer>

Agentic-R (Successful)

<think> John Travolta is an actor, so I will search for his filmography to find his third major acting role. </think>
<search> John Travolta third major acting role </search>
<information>
Doc1 (Urban Cowboy): Urban Cowboy is a 1980 American romantic drama film about the love-hate relationship between Buford Uan "Bud" Davis (John Travolta) and Sissy (Debra Winger). It was John Travolta's third major film role after Saturday Night Fever and Grease. Much of the action centers around activities at Gilley's Club, a football-field-sized honky tonk in Pasadena, Texas...
</information>

<think> Now that I have the name of the honky tonk, I will search for information about Gilley's Club and its founder. </think>
<search> Gilley's Club founder </search>
<information>
Doc2 (Gilley's Club): Gilley's was a bar/honky tonk founded in 1971 by country singer Mickey Gilley in Pasadena, Texas. It was the central location in the 1980 movie Urban Cowboy. It was a huge building with a corrugated steel roof that housed multiple bars and mechanical bulls. Connected to the club was a small rodeo arena that would also host both bicycle and motorcycle motocross races...
</information>

<answer> Mickey Gilley </answer>

Prompt for Search Agent Training

Answer the given question. You must conduct reasoning inside <think> and </think> first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by <search> query </search>, and it will return the top searched results between <information> and </information>. You can search as many times as you want. If you find no further external knowledge needed, you can directly provide the answer inside <answer> and </answer> without detailed illustrations. For example, <answer> xxx </answer>. Question: {question}.

Figure 5: The input prompt for training our search agent.

Single-turn Relevance Scoring Prompt (with reference answer)

You are an expert evaluator whose role is to determine how effectively a passage answers a search query.
Given a Search Query and a list of passages, you need to assign a utility score (from a scale of 0 to 100) for each passage. The utility score should reflect how well each passage **directly answers** or **contains the necessary information** to resolve the information need of the Search Query.
I will also provide you with a reference answer to the Search Query to help you assess the utility score of each passage.

The scoring criteria is shown as follows:
81-100 (Excellent Utility): The passage directly and comprehensively addresses the Search Query or completely contains the answer required by the current Search Query.
61-80 (High Utility): The passage contains the majority of the information needed to directly answer the Search Query but might miss some minor details.
41-60 (Moderate Utility): The passage is on-topic and addresses a part of the query's intent, but it is not a comprehensive answer.
21-40 (Low Utility): The passage mentions keywords from the query, but its main topic is different. It offers very limited value.
0-20 (Very Low Utility): The passage is completely irrelevant to the Search Query. It contains no useful facts or is actively distracting.
Now, I will give you the Search Query and {num} passages (each indicated by a number identifier []). Please compare all passages globally before outputting the utility scores, ensuring relative scoring consistency and more precise utility determination for each passage.

Search Query:
{query}
Passages:
Passage [1] {$passage_1$}
Passage [2] {$passage_2$}
…
Reference Answer:
{reference answer}

Please output the utility scores of the passages (strictly as integers between 0 and 100) in the order of the passage identifiers, separating the scores with a single space. If there is only one passage, output a single number. Example output for 3 passages: "68 42 97". Only output the utility scores, without any words or explanations.

Figure 6: The prompt for single-turn relevance scoring with sub-answer.

Single-turn Relevance Scoring Prompt (without reference answer)

```
You are an expert evaluator whose role is to determine how effectively a passage answers a
search query.
Given a Search Query and a list of passages, you need to assign a utility score (from a scale
of 0 to 100) for each passage. The utility score should reflect how well each passage
**directly answers** or **contains the necessary information** to resolve the information need
of the Search Query.

The scoring criteria is shown as follows:
**81-100 (Excellent Utility):** The passage directly and comprehensively addresses the Search
Query or completely contains the answer required by the current Search Query.
**61-80 (High Utility):** The passage contains the majority of the information needed to
directly answer the Search Query but might miss some minor details.
**41-60 (Moderate Utility):** The passage is on-topic and addresses a part of the query's
intent, but it is not a comprehensive answer.
**21-40 (Low Utility):** The passage mentions keywords from the query, but its main topic is
different. It offers very limited value.
**0-20 (Very Low Utility):** The passage is completely irrelevant to the Search Query. It
contains no useful facts or is actively distracting.
Now, I will give you the Search Query and {num} passages (each indicated by a number
identifier []). Please compare all passages globally before outputting the utility scores,
ensuring relative scoring consistency and more precise utility determination for each passage.

**Search Query:**
{query}
**Passages:**
Passage [1] {passage_1}
Passage [2] {passage_2}
…

Please output the utility scores of the passages (strictly as integers between 0 and 100) in
the order of the passage identifiers, separating the scores with a single space. If there is
only one passage, output a single number. Example output for 3 passages: "68 42 97". Only
output the utility scores, without any words or explanations.
```

Figure 7: The prompt for single-turn relevance scoring without sub-answer.

Sub-answer Generation Prompt

```
You are an **Expert Trajectory Analyst and Answer Generator**. Your task is to analyze Agentic
Search Agent Search-R1's complete reasoning trajectory for answering a given Question. Based
on the Question and its Gold Answer(s), you must infer the **precise sub-answer** intended to
be found by each generated search query in the trajectory.
**Search-R1's Multi-Turn Reasoning-Search Process:**
To answer a question, Search-R1 needs to conduct reasoning first every time it gets new
information. After reasoning, if Search-R1 finds it lacks some knowledge, it will call a
search engine by `<search> query </search>` and obtain the top searched results between
`<information>` and `</information>`. Search-R1 can search as many times as it wants. If
Search-R1 finds no further external knowledge needed, it can directly provide the answer
inside `<answer>` and `</answer>` without detailed illustrations. For example, `<answer> xxx
</answer>`.

Given the Question, Gold Answer(s) List and the Search-R1's Trajectory, you need to analyze
the provided Trajectory step-by-step. For **every** search query, determine the **precise sub-
answer** it was intended to find. The required sub-answer **MUST** be a direct fact or short
piece of information that answers or resolves the `<search>` query and **CRITICALLY** must
align with one of the provided Gold Answers.

If you can find such a sub-answer for each query, output it. If the sub-answer of a certain
query cannot be reliably inferred (which often occurs when the Search-R1 ultimately predicts
an incorrect answer), output "Not Sure".
**Strictly** wrap each generated sub-answer (or "Not Sure") using the tag `<sub-answer> </sub-
answer>`. Generate at most one sub-answer for each query. Output the results according to the
order of the corresponding queries in the trajectory. Output the answers **contiguously** and
separated only by a single space. **Output only the answers wrapped with tags, nothing else.**
For example: `<sub-answer> Beijing </sub-answer> <sub-answer> Not Sure </sub-answer>`.

Here are the Question, Gold Answer(s) List and the Agent Trajectory:
**Question:**
{question}

**Gold Answer(s) List:**
[{gold_answers}]

**Agent Trajectory:**
{trajectory}
```

Figure 8: The prompt for generating sub-answers.