

Learning When to Search: Multi-Stage Reward Design for Efficient Agentic RAG with Reasoning Language Models

Anonymous ACL submission

Abstract

Reasoning Language Models (RLMs) provide strong step-by-step reasoning and evidence integration for agentic Retrieval-Augmented Generation (RAG), yet their retrieval behaviors remain unstable, often exhibiting over and under retrieval that degrades both efficiency and reliability. A controlled comparison under matched parameter scales shows that RLMs consistently outperform conventional LLMs across QA benchmarks, with particularly pronounced gains on multi-hop tasks. Further analysis indicates that RLMs tend to repeatedly verify retrieved evidence and reconcile it with internal knowledge, which helps reduce hallucinations but may also trigger confirmation heavy overthinking when internal knowledge is uncertain or external evidence is insufficient or conflicting, leading to redundant retrieval. To address this issue, we propose a reinforcement learning framework centered on a multi-stage reward design that explicitly couples answer quality rewards with the number of retrieval steps. Through staged optimization, the model learns when to retrieve and how many retrievals are necessary, balancing correctness and retrieval cost. Experiments across multiple benchmarks demonstrate that our approach improves average F1 and accuracy over prompt based and prior reinforcement learning baselines while significantly reducing the average number of retrievals, achieving a better performance efficiency trade-off.

1 Introduction

Over the past year, reinforcement learning (Zhang et al., 2025c) driven Reasoning Language Models (RLMs) (Xu et al., 2025) have achieved substantial progress. Representative models such as OpenAI o1 (Wu et al., 2024), Qwen (Qwen3, 2025), and DeepSeek-R1 (DeepSeek-AI et al., 2025) exhibit strong long horizon, step-by-step reasoning

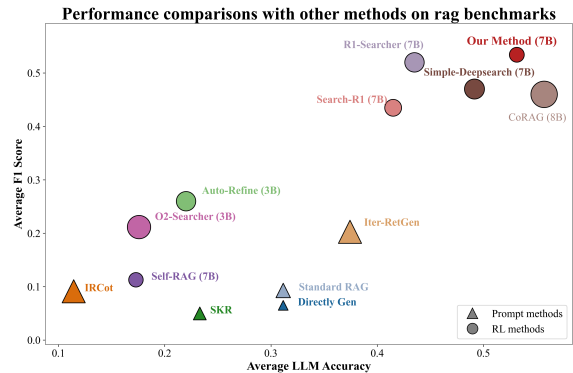


Figure 1: Performance comparisons with other methods on RAG benchmarks. The x-axis represents the average accuracy, the y-axis represents the average F1 score, and the size of the markers represents the average number of retrievals. The larger the marker, the more retrievals are performed.

capabilities, enabling test time reasoning expansion to handle increasingly complex tasks. While such capabilities are well aligned with information intensive applications, relying solely on internal model knowledge remains insufficient to meet requirements for factuality and timeliness. Retrieval-Augmented Generation (RAG) (Gao et al., 2024) addresses this limitation by incorporating external evidence to mitigate hallucinations and improve answer grounding. Agentic RAG (Singh et al., 2025) further formulates retrieval as an interactive decision making process, requiring models to dynamically decide whether, when, and what to retrieve during multi-step reasoning. However, existing agentic RAG systems are often constrained by sub-optimal retrieval strategies: over-retrieval leads to redundant reasoning and high computational costs, whereas under-retrieval (Wu et al., 2025b) deprives models of necessary external knowledge, jointly degrading efficiency and reliability.

Motivated by these challenges, we first conduct a controlled comparison between RLMs and con-

ventional LLMs under matched parameter scales across diverse question answering benchmarks. As illustrated in Figure 2, the results show that RLMs consistently outperform LLMs, with particularly pronounced gains on challenging multi-hop datasets. Further analysis reveals that the reasoning chains of RLMs enable repeated verification of retrieved evidence and its integration with internal knowledge, thereby improving information aggregation and suppressing hallucinations. However, this confirmation oriented reasoning paradigm may also induce overthinking in RAG settings: when internal knowledge is uncertain or external evidence is insufficient or conflicting, models tend to repeatedly retrieve and verify information, resulting in excessive retrieval and reduced efficiency.

To address this issue, we apply reinforcement learning with Dynamic sAmpling Policy Optimization (DAPO) (Yu et al., 2025) to train reasoning models and propose a multi-stage reward design that explicitly couples answer quality with retrieval cost. This design guides models toward efficient retrieval policies by balancing correctness and efficiency. As illustrated in Figure 1, experimental results demonstrate that compared with prompt based baselines and existing reinforcement learning methods, our approach achieves higher average F1 and accuracy across multiple benchmarks while significantly reducing the average number of retrievals, yielding a more favorable trade-off between performance and efficiency.

Our contributions are summarized as follows.

- Reasoning models can integrate evidence effectively via internal reasoning chains, yet under dynamic retrieval they often call tools unnecessarily, resulting in over-retrieval. We therefore introduce a reinforcement learning-based multi-stage framework that balances reasoning and retrieval, improving efficiency while strengthening reasoning performance.
- To further mitigate the problem of excessive retrieval and enhance the model’s reasoning abilities, we design a multi-stage reward function. This function operates through a dynamic adjustment mechanism, wherein rewards are applied differently at each stage to promote more efficient retrieval. By incentivizing the model to perform targeted retrieval only when necessary, we achieve a significant reduction in unnecessary retrieval be-

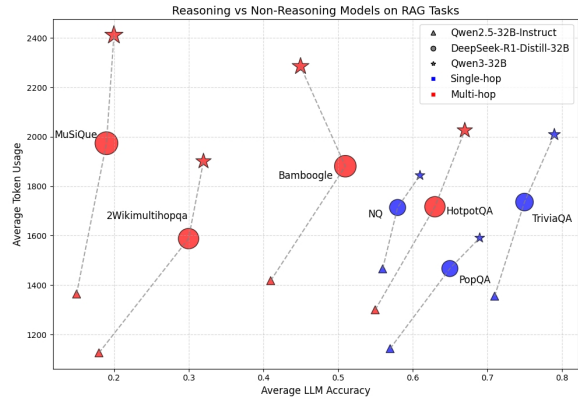


Figure 2: Reasoning models benefit from deeper thought in the reasoning chain, improving task accuracy.

havior while enhancing the model’s internal reasoning skills.

- Our experimental results show that the proposed method leads to substantial improvements in both retrieval count and accuracy. We compare our approach with other existing reinforcement learning based methods, demonstrating that our framework outperforms them in terms of both efficiency and accuracy. These results highlight the effectiveness of our approach in improving model performance by better balancing retrieval and reasoning.

2 Related work

2.1 Retrieval Augmented Generation

Early structural innovations LLM such as IRCoT (Trivedi et al., 2023) interleave retrieval with chain-of-thought reasoning, allowing each reasoning step to guide the next retrieval action. Iter-RetGen (Shao et al., 2023) is a method that synergizes retrieval and generation in an iterative manner to improve relevance modeling and performance on text tasks. Self-Knowledge guided Retrieval augmentation (SKR) (Wang et al., 2023) is a simple yet effective method which can let LLMs refer to the questions they have previously encountered and adaptively call for external resources when dealing with new questions.

2.2 Reinforcement Learning

The R1-Searcher (Song et al., 2025b) and Search-R1 (Jin et al., 2025a) series, which explicitly train LLMs to autonomously generate search queries during reasoning. R1-Searcher adopts a two stage

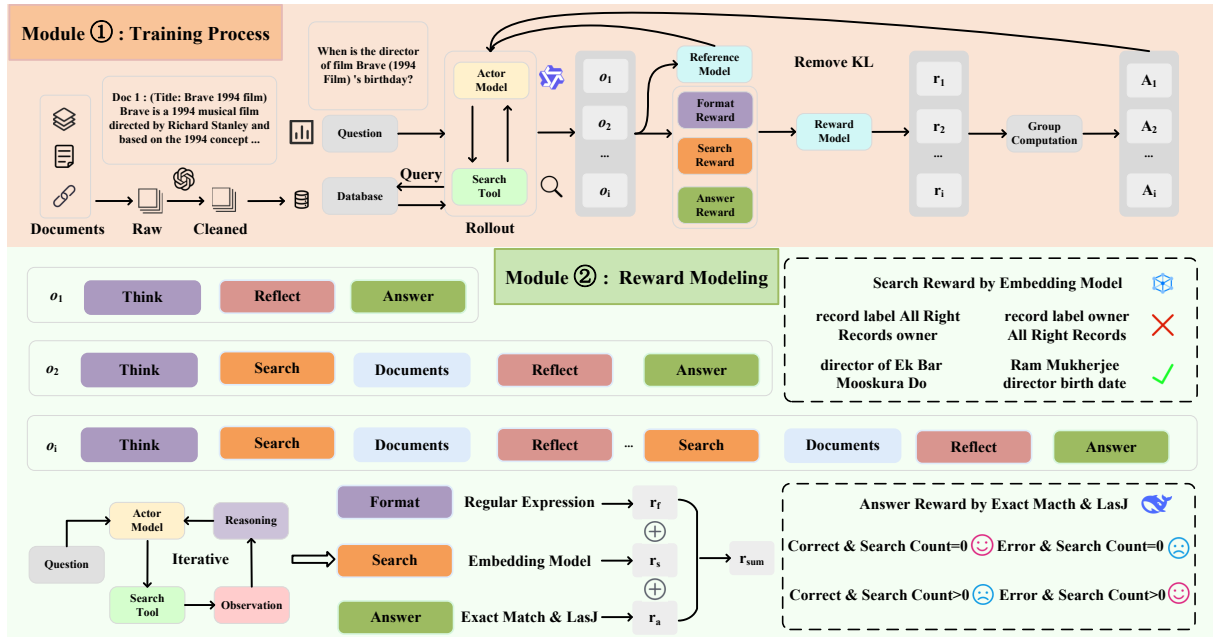


Figure 3: Overall framework of our proposed approach. Module 1 describes the entire workflow of reinforcement learning, and Module 2 introduces the reward function design scheme.

148 pipeline SFT for format acquisition followed by
 149 outcome based RL to encourage effective external
 150 search usage. ReSearch (Chen et al., 2025a) trains
 151 tool augmented reasoning agents in executable
 152 multi-step environments. Auto-Refine (Shi et al.,
 153 2025) enabling the model to iteratively filter, distill,
 154 and organize evidence before generating an answer.
 155 O2-Searcher (Mei et al., 2025) a novel search agent
 156 leveraging reinforcement learning to effectively
 157 tackle both open-ended and closed-ended questions
 158 in the open domain. Simple-Deepsearch (Sun et al.,
 159 2025b) a lightweight yet effective framework that
 160 bridges this gap through strategic data engineering
 161 rather than complex training paradigms. CoRAG
 162 (Wang et al., 2025) allows the model to dynami-
 163 cally reformulate the query based on the evolving
 164 state.

165 Existing studies primarily focus on enhancing
 166 internal reasoning capabilities, while systematic
 167 exploration of how to tightly couple step-by-step
 168 reasoning with external knowledge retrieval so that
 169 models not only reason effectively but also re-
 170 trieve judiciously and efficiently remains an open
 171 research challenge.

172 3 Methodology

173 As shown in the figure 3, we present the overall
 174 framework of our method, which consists of two
 175 core components: the training pipeline and the re-
 176 ward modeling.

177 3.1 Problem Definition

178 We formulate RAG as a sequential decision-making
 179 process. Given a question q , the model incremen-
 180 tally performs internal reasoning and may option-
 181 ally invoke an external retriever multiple times be-
 182 fore producing a final answer y . At step t , the
 183 system maintains a state

$$184 s_t = (q, x_t, e_t), \quad (1)$$

185 where x_t denotes the current intermediate reason-
 186 ing context and e_t is the set of evidence retrieved
 187 so far. At each step, the model selects an action

$$188 a_t \in \{Search, Answer\}. \quad (2)$$

189 When $a_t = Search$, the model issues a retrieval
 190 query and appends the newly obtained evidence to
 191 e_t ; when $a_t = Answer$, the process terminates and
 192 the model outputs a final answer.

193 3.2 Training Process

194 In this study, we adopt the DAPO algorithm. Un-
 195 like PPO, which trains a separate critic model,
 196 DAPO optimizes the policy by estimating the base-
 197 line from a group of rollouts. Given an existing
 198 policy $\pi_{\theta_{old}}$ and a reference policy $\pi_{\theta_{ref}}$, based on
 199 G rollouts for each input, the objective of DAPO
 200 is to optimize the policy π_{θ} by maximizing the
 201 following objective:

$$E_{(q,a_t) \sim D, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G} \sum_{i=1}^G |o_i| \sum_{t=1}^{|o_i|} \min \left(r_{i,t}(\theta) A_{i,t}, \text{clip}(r_{i,t}(\theta), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}}) A_{i,t} \right) \right] \quad (3)$$

Explanation of Symbols:

- $\pi_{\theta_{\text{old}}}$: Old policy used to generate trajectories.
- o_i : A trajectory representing the sequence of actions and states taken.
- $r_{i,t}(\theta)$: Reward at time step t , used to update the policy based on the reward. r_i is the return for the i -th rollout, representing the total accumulated reward from a given time step onward.

$$r_{i,t}(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \quad (4)$$

- $A_{i,t}$: Advantage at time step t , compared to the expected return. A_i is the advantage value of the i -th rollout. It represents the deviation of the model’s return r_i in that rollout compared to the average return of all rollouts.

$$A_i = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)} \quad (5)$$

- $\epsilon_{\text{low}}, \epsilon_{\text{high}}$: The clipping range to prevent large updates.
- G : Number of rollouts generated in a batch.
- D : Dataset contains questions and answers.

When t is included, the advantage and reward are calculated at each time step of the trajectory, whereas without t , the advantage and reward are computed over the entire rollout, reflecting the overall performance across the whole sequence.

In contrast to Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which uses a KL divergence penalty to prevent the policy from deviating too much from the reference policy, DAPO removes the KL divergence term, focusing instead on decoupling the clip operation and dynamically sampling rollouts to optimize the policy more effectively. We have encountered training crashes when using GRPO. Therefore, we choose DAPO.

Rollout with Search

Unlike traditional rollouts that only contain text-based reasoning, the rollouts in our method also incorporate search queries and retrieval results. We use `<search>` and `</search>` to denote the search queries, and `<information>` and `</information>` to denote the retrieved information. Additionally, reflection is facilitated through the `<reflect>` and `</reflect>` tags. These instructions are defined within the prompt templates, which will be introduced in the Appendix.

The rollout process forms an iterative cycle between text-based reasoning, search queries, retrieval results, and reflection. Specifically, when the generation process encounters the `</search>` tag, the query between the preceding `<search>` and the current `</search>` tags will be used as a search query to retrieve relevant factual information, and the retrieval results will be enclosed within `<information>` and `</information>` tags. The existing rollout, concatenated with the retrieval results, will be used as the next input to generate subsequent responses, and reflection will be incorporated through the `<reflect>` tags until the generation process encounters an end-of-sentence tag.

3.3 Reward Modeling

In the reinforcement learning process, the reward function serves as a supervisory signal to adjust and optimize the behavior of the model. Therefore, given a question q and the i -th generated response o_i , we design three types of rewards: **format reward**, **answer reward**, and **search reward**. The following section introduces the details of the reward function.

Format Reward

The format reward ensures that the model’s output adheres to the expected structural format. In many cases, deviations from the required format can lead to inefficiencies or errors in further processing. This function encourages the model to generate responses that are both syntactically correct and semantically accurate, thereby ensuring that the output meets the necessary structural standards.

We define two types of output structures: when retrieval is not required, the output consists of think, reflect, answer; when retrieval is needed, the output includes multiple iterations of think, search, information, reflect, answer. The format reward is

calculated based on the alignment of the model’s output with these structures.

The format reward is calculated as follows:

$$r_f = \begin{cases} 1.0, & \text{they fully match} \\ -1.0, & \text{otherwise} \end{cases} \quad (6)$$

This structure ensures that the model prioritizes generating outputs in the correct format, while penalizing deviations, thereby guiding the model to produce structurally consistent and correct responses.

Search Reward

The search reward module evaluates the quality of search queries rather than directly scoring the retrieved documents. Traditional methods often score documents based on relevance, but this approach is heavily dependent on the retriever’s performance, which is not the focus of our study. Instead, we focus on evaluating the search queries and their impact on the retrieval process.

We use a rule-based approach to determine whether the query is concise. Specifically, we apply rules to avoid redundant question words, prepositions, and irrelevant modifiers in the query, ensuring that it remains as brief and direct as possible.

For example, for a simple question, if relevant results are not found using keywords, the model should refrain from continuing to search, as this could be due to issues with the retriever. For complex queries, the model is encouraged to perform multiple rounds of retrieval in different directions. Our goal is to promote this behavior, enabling the model to demonstrate flexibility and diversity in handling multi-hop queries. The cosine similarity between two vectors m and n is calculated as follows:

$$\text{cosine_similarity}(m, n) = \frac{m \cdot n}{\|m\| \|n\|} \quad (7)$$

Where m and n are the embedding vectors of two search queries. $\|\cdot\|$ represents the Euclidean norm of the vectors. The average similarity between all search queries is computed as the mean of pairwise cosine similarities:

$$\text{avg_sim} = \frac{\sum_{i=1}^N \sum_{j=i+1}^N \text{cosine_similarity}(m_i, n_j)}{N(N-1)} \quad (8)$$

Where N is the total number of search queries. m_i and m_j are the embedding vectors of the i -th and j -th search queries. The search reward is calculated based on the number of retrieval counts (RC), expressed as a piecewise function:

$$r_s = \begin{cases} 0, & \text{concise \& RC} \leq 1 \\ -1.0, & \text{no concise \& RC} \leq 1 \\ -\text{avg_sim}, & \text{if RC} > 1 \end{cases} \quad (9)$$

This structure ensures that the model optimizes the retrieval process by promoting diversity and effectiveness while minimizing redundant search behaviors.

Multi-Stage Answer Reward

The reward function adjusts the model’s behavior based on the correctness of its answer and the number of retrieval counts it performs.

In **Stage 1**, for correct answers, the reward is independent of the number of retrieval counts. For incorrect answers, the reward is calculated based on the number of retrieval counts, encouraging the model to perform more retrievals in an attempt to improve the answer. The reward for an incorrect answer is given by where we set β to 0.3 :

$$r_a = \begin{cases} 1.0, & \text{if } y \text{ is correct} \\ -1.0 + \beta \cdot \text{RC}, & \text{otherwise} \end{cases} \quad (10)$$

In **Stage 2**, the reward for correct answers is adjusted to account for the number of retrieval cycles, encouraging the model to perform retrievals when necessary. The reward for correct answers is calculated as:

$$r_a = \begin{cases} 1.0 - \beta \cdot \text{RC}, & \text{if } y \text{ is correct} \\ -1.0, & \text{otherwise} \end{cases} \quad (11)$$

This staged reward structure guides the model’s behavior by first encouraging more retrievals for incorrect answers and then optimizing retrievals for correct answers in the later stage.

The reason for stage-based training is that in dynamic retrieval, the model may suffer from insufficient retrieval, which can reduce task accuracy. The final reward consists of the following components:

$$r_{\text{sum}} = r_a + r_s + r_f \quad (12)$$

r_{sum} is used to evaluate the overall performance and guide the optimization of the model’s policy.

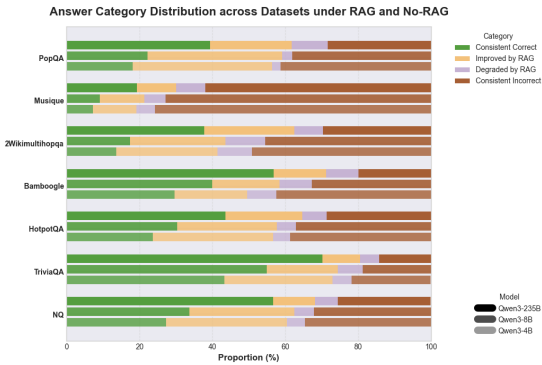


Figure 4: Answer category distribution across datasets under RAG and non-RAG conditions. The answers are divided into four categories. The proportion of each category is shown for three models across different datasets.

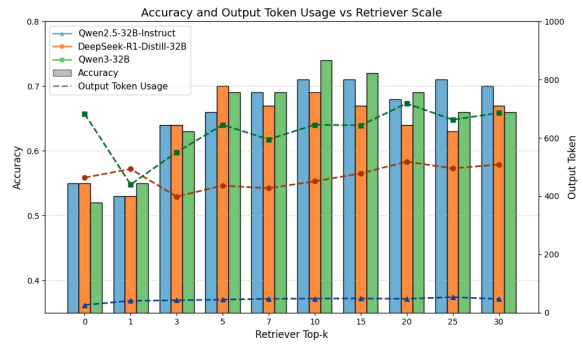


Figure 5: Accuracy and output token usage as a function of retriever top-k. Accuracy increases with top-k and stabilizes after a certain point. Experiments are conducted on both standard and reasoning models.

4 Experiments

4.1 Datasets and Metrics

We evaluate our approach on seven established open-domain question answering, Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023) and multi-hop reasoning benchmarks, HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2023), Bamboogle (Shen et al., 2023).

As shown in the figure 4 we classified the dataset into four categories: (1) correctly answered with and without RAG, (2) correctly answered with RAG indicating that the retrieved documents contributed to the answer, (3) correctly answered without RAG indicating that the retrieved documents contain noise, and (4) incorrectly answered in both cases indicating issues with the dataset itself. These four categories allow us to assess the impact of retrieval documents on model performance.

We report three evaluation metrics: F1 score, LLM as a Judge (using a large model for scoring), and Retrieval Counts. Throughout both the training and testing phases, we used the same evaluation large model to ensure the accuracy of the results.

4.2 Preliminary

We provide a detailed description of the retriever setup. We utilized the work of FlashRAG (Jin et al., 2025b) to prepare our retriever, as our task focuses on training and optimizing the generator rather than working on the retriever. As a result, all experiments in this paper are conducted using the same retriever. For the corpus, we use the September 2025 Wikipedia data. The pooling method selected

for the embeddings is the mean method for the e5 model. For the search reward, we use the latest Qwen3-Embedding (Zhang et al., 2025g). Our approach does not involve rerank models, and all model deployments are initiated using the VLLM method. We provide a detailed description of experiments in the appendix.

To investigate the impact of the number of retrieved documents on model performance, we conducted detailed experiments. We compared the results between a LLM and a RLM with the same parameter size. As shown in the figure 5, the results demonstrate that increasing the number of retrieved documents does not necessarily improve performance. While retrieving more documents increases the number of input tokens, it does not lead to an improvement in accuracy. To better understand the relationship between the model’s context, retrieval count, and document length, we standardized the retrieval setting with top k equal to 5.

4.3 Main Results

As shown in Table 1, the reasoning model’s reasoning chain significantly enhances accuracy across various datasets. In the NQ dataset, the Qwen3-4B model initially achieved a LasJ score of 0.38 and an RC score of 0.18 in Stage one, demonstrating basic retrieval capabilities. This indicates the model’s early stage of learning how to retrieve relevant information, albeit with limited efficiency. As the training progresses within Stage one, the model further improves, with the LasJ score rising to 0.45 and the RC score increasing to 1.09, signifying the model’s ability to perform multi-round retrieval. At this stage, the model becomes more proficient at conducting multiple retrievals to yield

Methods	Single-hop QA						Multi-hop QA							
	NQ		PopQA		TriviaQA		HotpotQA		MuSiQue		2WIKI		Bamboogle	
	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC
<i>Qwen3-4B</i>														
Zero-shot	0.32	0.00	0.20	0.00	0.48	0.00	0.28	0.00	0.12	0.00	0.23	0.00	0.37	0.00
Naive	0.55	1.00	0.51	1.00	0.73	1.00	0.56	1.00	0.19	1.00	0.41	1.00	0.50	1.00
Naive w/o think	0.53	1.00	0.43	1.00	0.58	1.00	0.42	1.00	0.15	1.00	0.41	1.00	0.37	1.00
	0.38	0.18	0.32	0.33	0.67	0.13	0.35	0.30	0.16	0.41	0.37	0.45	0.39	0.14
Stage one	0.45	1.09	0.49	1.12	0.67	1.14	0.47	1.35	0.18	1.52	0.53	1.65	0.45	1.35
	0.48	1.83	0.51	1.82	0.72	1.88	0.54	2.25	0.22	2.28	0.54	2.75	0.46	2.05
Stage two	0.57	1.07	0.54	1.06	0.83	1.03	0.55	1.11	0.24	1.25	0.61	1.33	0.61	1.44
<i>Qwen3-8B</i>														
Zero-shot	0.39	0.00	0.25	0.00	0.61	0.00	0.35	0.00	0.14	0.00	0.28	0.00	0.48	0.00
Naive	0.62	1.00	0.52	1.00	0.70	1.00	0.54	1.00	0.18	1.00	0.43	1.00	0.54	1.00
Naive w/o think	0.47	1.00	0.48	1.00	0.58	1.00	0.42	1.00	0.13	1.00	0.37	1.00	0.27	1.00
	0.51	0.43	0.46	0.62	0.66	0.26	0.48	0.65	0.21	0.70	0.39	0.85	0.49	0.36
Stage one	0.57	1.13	0.53	1.27	0.72	1.01	0.59	1.79	0.24	2.29	0.58	1.98	0.51	1.20
	0.64	1.52	0.56	1.64	0.79	1.65	0.62	2.26	0.31	2.61	0.64	2.19	0.65	2.01
Stage two	0.64	1.13	0.53	1.01	0.81	1.01	0.55	1.15	0.27	2.15	0.51	1.41	0.56	1.22

Table 1: Multi-stage capability evolution of a reasoning model during training. We report results for Qwen3-4B and Qwen3-8B across single-hop and multi-hop QA benchmarks, comparing Zero-shot and Naive using the LLM as a Judge (LasJ) and Retrieval Count metrics.

Checkpoints	NQ		PopQA		TriviaQA		HotpotQA		MuSiQue		2WIKI		Bamboogle	
	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC
Qwen3-4B														
Step 30	0.38	0.18	0.32	0.33	0.67	0.13	0.35	0.30	0.16	0.41	0.37	0.45	0.39	0.14
Step 90	0.48	0.49	0.47	0.80	0.75	0.39	0.44	0.72	0.19	0.97	0.47	1.08	0.41	0.29
Step 180	0.49	0.35	0.49	0.61	0.74	0.23	0.42	0.64	0.18	0.76	0.48	0.78	0.38	0.28
Qwen3-8B														
Step 30	0.38	0.18	0.33	0.33	0.66	0.13	0.37	0.30	0.13	0.41	0.34	0.45	0.45	0.14
Step 90	0.47	0.44	0.45	0.79	0.74	0.26	0.45	0.60	0.18	0.82	0.50	1.03	0.48	0.22
Step 180	0.47	0.40	0.47	0.77	0.74	0.20	0.43	0.43	0.18	0.69	0.50	0.86	0.48	0.10

Table 2: Performance of RLMs at training steps, showing inadequate retrieval behavior under the unified dynamic reward function, without stage-based training.

more accurate and comprehensive answers.

In Stage two, further optimization leads to improved accuracy and efficiency. The model not only learns appropriate retrieval behavior but also avoids excessive retrieval. For instance, on the NQ dataset, the Qwen3-4B model reaches a LasJ score of 0.57 and an RC score of 1.07, indicating a better balance between retrieval accuracy and efficiency.

Through multi-stage training, the model transitions from learning basic retrieval to mastering multi-round retrieval, and ultimately optimizing retrieval efficiency. This process leads to significant improvements in both accuracy and retrieval efficiency, achieving optimization in multi-hop QA tasks.

During the model training process, we conducted tests on the checkpoints at various training steps. As shown in Table 2, we observed that training without stage-based learning where answer rewards for both correct and incorrect responses are di-

rectly tied to the number of retrievals results in lower accuracy. This is because the model has not learned multi-round retrieval, leading to inadequate retrieval performance.

4.4 Baselines

We compared our approach with other methods, which can be divided into prompt-based methods and RL methods. Our approach uses a different RL method and strategy. To ensure fairness, we conducted training on the standard model. As shown in the table 3, our method demonstrates significant advantages over other approaches across various benchmarks.

Moreover, retrieval efficiency is a key strength of our method. On Bamboogle, our model achieves an RC of 1.03 with an acc score of 0.49, while R1-Searcher requires 2.11 retrievals for an acc score of 0.39, highlighting a significant improvement in efficiency. These results indicate that our approach

Type	Methods	NQ			PopQA			TriviaQA			HotpotQA [‡]			MuSiQue [‡]			2WIKI [‡]			Bamboogle [‡]		
		F1	ACC	RC	F1	ACC	RC	F1	ACC	RC	F1	ACC	RC	F1	ACC	RC	F1	ACC	RC	F1	ACC	RC
Prompt	Directly Gen	0.09	0.32	0.00	0.05	0.25	0.00	0.17	0.52	0.00	0.07	0.26	0.00	0.04	0.13	0.00	0.08	0.43	0.00	0.08	0.27	0.00
	Standard RAG	0.14	0.41	1.00	0.11	0.34	1.00	0.07	0.32	1.00	0.15	0.40	1.00	0.05	0.18	1.00	0.11	0.41	1.00	0.02	0.12	1.00
	IRCoT	0.08	0.07	3.16	0.04	0.08	3.79	0.15	0.31	3.10	0.12	0.11	2.70	0.05	0.04	2.68	0.08	0.08	2.52	0.12	0.11	3.03
	Iter-RetGen	0.15	0.49	3.00	0.20	<u>0.65</u>	3.00	0.19	0.17	3.00	0.18	0.43	3.00	0.07	0.23	3.00	0.30	0.33	3.00	0.33	0.32	3.00
	SKR	0.04	0.18	0.78	0.02	0.13	0.56	0.09	0.37	0.75	0.03	0.20	0.72	0.02	0.06	0.74	0.06	0.39	0.74	0.09	0.30	0.83
RL	Auto-Refine (3B)	0.25	0.22	2.00	0.16	0.17	2.04	0.43	0.39	1.99	0.26	0.20	2.02	0.14	0.08	2.04	0.30	0.26	2.02	0.28	0.22	2.04
	O2-Searcher (3B)	0.20	0.18	3.04	0.11	0.10	2.70	0.38	0.35	2.08	0.21	0.16	3.15	0.10	0.05	2.70	0.19	0.17	4.89	0.29	0.22	2.76
	Search-R1 (7B)	0.42	0.38	1.43	0.50	0.56	1.31	0.63	0.60	1.48	0.47	0.53	1.39	0.47	0.41	1.91	0.53	0.52	1.18	0.21	0.19	1.46
	Self-RAG (7B)	0.05	0.08	1.00	0.09	0.12	1.00	0.23	0.38	1.00	0.13	0.18	1.00	0.06	0.04	1.00	0.17	0.32	1.00	0.06	0.09	1.00
	R1-Searcher (7B)	0.43	0.50	1.87	0.48	0.53	1.54	<u>0.63</u>	0.61	1.84	0.53	0.46	2.22	0.34	0.30	2.40	<u>0.63</u>	<u>0.61</u>	2.20	0.48	0.39	2.11
	Simple-Deepsearch (7B)	<u>0.44</u>	0.42	2.10	<u>0.52</u>	0.60	1.60	0.66	0.67	1.80	0.47	0.44	2.30	0.31	0.30	2.45	0.55	0.57	2.70	0.52	0.44	2.26
	CoRAG (8B)	0.39	0.47	4.00	0.40	0.69	4.00	0.62	<u>0.69</u>	4.00	0.49	<u>0.51</u>	4.00	0.38	0.38	4.00	0.64	0.67	4.00	0.49	0.53	4.00
RL	Our Method (3B)	0.27	0.39	0.59	0.37	0.33	0.85	0.40	0.57	0.39	0.39	0.38	0.75	0.21	0.14	1.10	0.49	0.28	0.68	0.45	0.29	0.87
	Our Method (7B)	0.46	<u>0.48</u>	0.82	0.57	0.59	1.04	0.60	0.73	0.66	<u>0.52</u>	0.53	1.09	<u>0.34</u>	<u>0.37</u>	1.65	0.55	0.53	1.20	<u>0.49</u>	<u>0.49</u>	1.03

Table 3: Performance comparisons with other methods across seven datasets with three metrics (F1/ACC/RC). Best and second-best can be highlighted with **bold** and underline. 3B represents Qwen2.5-3B-Instruct and 7B represents Qwen2.5-7B-Instruct.

Settings	NQ		PopQA		TriviaQA		HotpotQA		MuSiQue		2WIKI		Bamboogle	
	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC	LasJ	RC
Qwen3-4B														
Ours	0.57	1.07	0.54	1.06	0.83	1.03	0.55	1.11	0.24	1.25	0.61	1.33	0.61	1.14
w/o r_s	0.57	1.33	0.53	1.14	0.81	1.14	0.52	1.21	0.22	1.34	0.54	1.44	0.59	1.19
w/o stage one	0.36	0.12	0.25	0.31	0.65	0.61	0.31	0.48	0.14	0.23	0.34	0.39	0.40	0.31
w/o stage two	0.55	1.71	0.54	1.72	0.81	1.23	0.53	1.75	0.23	2.33	0.57	2.52	0.59	1.49
w/o r_a	0.48	2.02	0.52	1.93	0.69	2.03	0.55	2.09	0.19	2.48	0.54	1.93	0.38	1.96

Table 4: Ablation study of reward components on Qwen3-4B across multiple RAG benchmarks.

achieves a better balance between accuracy and retrieval cost, demonstrating superior performance across both single-hop and multi-hop QA tasks.

Our method consistently outperforms other RL-based models in both accuracy and retrieval efficiency. By optimizing retrieval behavior, it achieves higher performance with fewer retrievals, making it a more efficient solution for RAG tasks.

4.5 Ablation Study

Table 4 reports an ablation study on the reward design for Qwen3-4B. Overall, removing any single reward component consistently degrades performance, confirming that the full objective is necessary for stable improvement. In particular, the search reward is crucial for learning effective tool use: without it, the model becomes less reliable in leveraging external evidence and the retrieval behavior is less well-controlled. Similarly, removing different stages affects the model’s training process. For instance, removing Stage one means that in the reward function for incorrect answers during this stage, the number of retrievals is no longer linked to the answer, which results in inadequate retrieval behavior. On the other hand, removing Stage two implies that in the reward function for correct answers during this stage, the number of retrievals is no longer associated with the answer, leading to

over-retrieval behavior.

Removing the answer level reward most directly harms end task quality, indicating that explicit supervision on final correctness is essential to align retrieval actions with answering goals. Together, these results validate that the proposed reward composition provides complementary signals which encourage accurate answers while discouraging unnecessary retrieval.

5 Conclusion

In conclusion, we show that reasoning language models deliver stronger evidence integration for agentic RAG, but their retrieval policies are often unstable between under- and over-retrieval, which harms both efficiency and reliability. Our analysis attributes this to confirmation-heavy reasoning: repeated verification improves faithfulness yet can trigger redundant retrieval when knowledge is uncertain or evidence is noisy. To resolve this problem, we propose a reinforcement learning framework with a multi-stage reward that jointly optimizes answer quality and retrieval cost, enabling the model to learn when to retrieve and how much retrieval is sufficient. Across diverse benchmarks, our method improves overall answer quality while substantially reducing unnecessary retrieval, achieving a better performance efficiency trade-off.

528 Limitations

529 We acknowledge three limitations in our study.
530 First, our experiments are conducted exclusively
531 on the Qwen series of language models. We do
532 not explore or evaluate models from other families
533 (e.g., Llama or DeepSeek) due to the architectural
534 incompatibilities: the verl-tool reinforcement learn-
535 ing (RL) training framework employed in this work
536 currently supports only the Qwen architecture.

537 Furthermore, all datasets used in our study are in
538 English. Consequently, the model’s performance
539 may differ when applied to tasks in other lan-
540 guages—a direction beyond the scope of this work.

541 Finally, while reinforcement learning is known
542 to suffer from issues such as low data efficiency and
543 slow training convergence, addressing these chal-
544 lenges lies outside the focus of the present study.

545 Potential Risks

546 Our systems integrate external knowledge sources
547 to enhance model outputs, yet this dependency in-
548 troduces critical risks that require systematic anal-
549 ysis. First, external knowledge repositories may
550 contain outdated, biased, or adversarially injected
551 content, which can compromise the factual accu-
552 racy and reliability of generated responses.

553 Second, privacy and data leakage risks are in-
554 herent in RAG architectures. Sensitive information
555 embedded in private documents or user queries
556 may be inadvertently exposed during retrieval or
557 generation processes.

558 References

559 Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou,
560 Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen
561 Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and
562 Weipeng Chen. 2025a. [Research: Learning to reason with search for llms via reinforcement learning](#). *Preprint*, arXiv:2503.19470.

565 Wang Chen, Guanqiang Qi, Weikang Li, Yang
566 Li, Deguo Xia, and Jizhou Huang. 2025b. [Pairs: Parametric-verified adaptive information retrieval and selection for efficient rag](#). *Preprint*, arXiv:2508.04057.

570 Yiqun Chen, Erhan Zhang, Lingyong Yan, Shuaiqiang
571 Wang, Jizhou Huang, Dawei Yin, and Jiaxin Mao.
572 2025c. [Mao-arag: Multi-agent orchestration for adaptive retrieval-augmented generation](#). *Preprint*, arXiv:2508.01005.

575 DeepSeek-AI, Daya Guo, and Yang. 2025. [DeepSeek-R1: Incentivizing Reasoning Capability in LLMs](#)

577 [via Reinforcement Learning](#). *arXiv e-prints*,
578 arXiv:2501.12948.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia,
579 Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang,
580 and Haofen Wang. 2024. [Retrieval-augmented generation for large language models: A survey](#). *Preprint*,
581 arXiv:2312.10997. 582 583

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin,
584 Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and
585 Jie Zhou. 2025. [Deeprag: Thinking to retrieve step by step for large language models](#). *Preprint*,
586 arXiv:2502.01142. 587 588

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara,
589 and Akiko Aizawa. 2020. [Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics. 590 591 592 593 594 595

Wentao Hu, Wengyu Zhang, Yiyang Jiang, Chen Jason
596 Zhang, Xiaoyong Wei, and Qing Li. 2025. [Removal of hallucination on hallucination: Debate-augmented rag](#). *Preprint*, arXiv:2505.18581. 597 598 599

Ziyang Huang, Xiaowei Yuan, Yiming Ju, Jun Zhao,
600 and Kang Liu. 2025. [Reinforced internal-external knowledge synergistic reasoning for efficient adaptive search agent](#). *Preprint*, arXiv:2505.07596. 601 602 603

Abhinav Java, Srivathsan Koundinyan, Nagarajan
604 Natarajan, and Amit Sharma. 2025. [Frugalrag: Learning to retrieve and reason for multi-hop qa](#). *Preprint*, arXiv:2507.07634. 605 606 607

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon,
608 Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei
609 Han. 2025a. [Search-r1: Training llms to reason and leverage search engines with reinforcement learning](#). *Preprint*, arXiv:2503.09516. 610 611 612

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong,
613 Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao
614 Yang, and Ji-Rong Wen. 2025b. [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. 615 616 617 618 619 620

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke
621 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 (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics. 622 623 624 625 626 627

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Red-
628 field, Michael Collins, Ankur Parikh, Chris Alberti,
629 Danielle Epstein, Illia Polosukhin, Jacob Devlin, Ken-
630 ton Lee, Kristina Toutanova, Llion Jones, Matthew
631 Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob 632

633	Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research . <i>Transactions of the Association for Computational Linguistics</i> , 7:452–466.	
634		
635		
636		
637	Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025a. Search-o1: Agentic search-enhanced large reasoning models . <i>Preprint</i> , arXiv:2501.05366.	
638		
639		
640		
641	Xingxuan Li, Weiwen Xu, Ruochen Zhao, Fangkai Jiao, Shafiq Joty, and Lidong Bing. 2025b. Can we further elicit reasoning in LLMs? critic-guided planning with retrieval-augmentation for solving challenging tasks . In <i>Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)</i> , pages 25589–25604, Vienna, Austria. Association for Computational Linguistics.	
642		
643		
644		
645		
646		
647		
648		
649	Hao Lin, Peitong Xie, Jingxue Chen, Jie Lin, Qingkun Tang, and Qianchun Lu. 2025. Domain-aware rag: Mol-enhanced rl for efficient training and scalable retrieval . <i>Preprint</i> , arXiv:2509.06650.	
650		
651		
652		
653	Yu Liu, Yanbing Liu, Fangfang Yuan, Cong Cao, Youbang Sun, Kun Peng, WeiZhuo Chen, Jianjun Li, and Zhiyuan Ma. 2025. Opera: A reinforcement learning-enhanced orchestrated planner-executor architecture for reasoning-oriented multi-hop retrieval . <i>Preprint</i> , arXiv:2508.16438.	
654		
655		
656		
657		
658		
659	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 <i>Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)</i> , pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.	
660		
661		
662		
663		
664		
665		
666		
667	Jianbiao Mei, Tao Hu, Daocheng Fu, Licheng Wen, Xueming Yang, Rong Wu, Pinlong Cai, Xinyu Cai, Xing Gao, Yu Yang, Chengjun Xie, Botian Shi, Yong Liu, and Yu Qiao. 2025. O²-searcher: A searching-based agent model for open-domain open-ended question answering . <i>Preprint</i> , arXiv:2505.16582.	
668		
669		
670		
671		
672		
673	Thang Nguyen, Peter Chin, and Yu-Wing Tai. 2025. Marag: Multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning . <i>Preprint</i> , arXiv:2505.20096.	
674		
675		
676		
677	Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report . <i>Preprint</i> , arXiv:2412.15115.	
678		
679		
680		
681		
682		
683		
684	Qwen3. 2025. Qwen3: Think deeper, act faster .	
685	Jingyi Ren, Yekun Xu, Xiaolong Wang, Weitao Li, Weizhi Ma, and Yang Liu. 2025. Effective and transparent rag: Adaptive-reward reinforcement learning for decision traceability . <i>Preprint</i> , arXiv:2505.13258.	
686		
687		
688		
689		
	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 . <i>Preprint</i> , arXiv:2305.15294.	690
		691
		692
		693
		694
	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models . <i>Preprint</i> , arXiv:2402.03300.	695
		696
		697
		698
		699
		700
	Tianjun Shen, Yujia Qin, Max Bartolo, Sebastian Ruder, and 1 others. 2023. Bamboogle: A new benchmark for multi-hop question answering . Google AI Blog.	701
		702
		703
	Wentao Shi and Yiqing Shen. 2025. Reinforcement fine-tuning for reasoning towards multi-step multi-source search in large language models . <i>Preprint</i> , arXiv:2506.08352.	704
		705
		706
		707
	Yaorui Shi, Sihang Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. 2025. Search and refine during think: Facilitating knowledge refinement for improved retrieval-augmented reasoning . <i>Preprint</i> , arXiv:2505.11277.	708
		709
		710
		711
		712
	Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag . <i>Preprint</i> , arXiv:2501.09136.	713
		714
		715
		716
	Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025a. R1-searcher: Incentivizing the search capability in llms via reinforcement learning . <i>Preprint</i> , arXiv:2503.05592.	717
		718
		719
		720
		721
	Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025b. R1-searcher++: Incentivizing the dynamic knowledge acquisition of llms via reinforcement learning . <i>Preprint</i> , arXiv:2505.17005.	722
		723
		724
		725
		726
		727
	Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2025a. Zerosearch: Incentivize the search capability of llms without searching . <i>Preprint</i> , arXiv:2505.04588.	728
		729
		730
		731
		732
	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. 2025b. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis . <i>Preprint</i> , arXiv:2505.16834.	733
		734
		735
		736
		737
		738
		739
	Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions . In <i>Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)</i> ,	740
		741
		742
		743
		744
		745

746	pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.	2369–2380, Brussels, Belgium. Association for Computational Linguistics.	802
747			803
748	Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. 2025. Chain-of-retrieval augmented generation . <i>Preprint</i> , arXiv:2501.14342.	Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale . <i>Preprint</i> , arXiv:2503.14476.	804
749			805
750			806
751			807
752	Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023. Self-knowledge guided retrieval augmentation for large language models . <i>Preprint</i> , arXiv:2310.05002.		808
753			809
754			810
755	Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. 2025a. Mmsearch-r1: Incentivizing llms to search . <i>Preprint</i> , arXiv:2506.20670.	Chi Zhang, Qiyang Chen, and Mengqi Zhang. 2025a. Mixture-of-rag: Integrating text and tables with large language models . <i>Preprint</i> , arXiv:2504.09554.	812
756			813
757			814
758			
759	Peilin Wu, Mian Zhang, Xinlu Zhang, Xinya Du, and Zhiyu Zoey Chen. 2025b. Search wisely: Mitigating sub-optimal agentic searches by reducing uncertainty . <i>Preprint</i> , arXiv:2505.17281.	Dingchu Zhang, Yida Zhao, Jialong Wu, Baixuan Li, Wenbiao Yin, Liwen Zhang, Yong Jiang, Yufeng Li, Kewei Tu, Pengjun Xie, and Fei Huang. 2025b. Evolvesearch: An iterative self-evolving search agent . <i>Preprint</i> , arXiv:2505.22501.	815
760			816
761			817
762			818
763	Ruofan Wu, Youngwon Lee, Fan Shu, Danmei Xu, Seung won Hwang, Zhewei Yao, Yuxiong He, and Feng Yan. 2025c. Composerag: A modular and composable rag for corpus-grounded multi-hop question answering . <i>Preprint</i> , arXiv:2506.00232.	Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, Yu Fu, Xingtai Lv, Yuchen Zhang, Sihang Zeng, Shang Qu, Haozhan Li, Shijie Wang, Yuru Wang, Xinwei Long, and 20 others. 2025c. A survey of reinforcement learning for large reasoning models . <i>Preprint</i> , arXiv:2509.08827.	820
764			821
765			822
766			823
767			824
768	Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, Qunshu Lin, Junbo Zhao, Zhaoxiang Zhang, Wenhao Huang, Ge Zhang, Chenghua Lin, and J. H. Liu. 2024. A Comparative Study on Reasoning Patterns of OpenAI’s o1 Model . <i>arXiv e-prints</i> , arXiv:2410.13639.	Ningning Zhang, Chi Zhang, Zhizhong Tan, Xingxing Yang, Weiping Deng, and Wenyong Wang. 2025d. Credible plan-driven rag method for multi-hop question answering . <i>Preprint</i> , arXiv:2504.16787.	825
769			826
770			827
771			828
772			829
773			830
774			
775	Weiqi Wu, Xin Guan, Shen Huang, Yong Jiang, Pengjun Xie, Fei Huang, Jiuxin Cao, Hai Zhao, and Jingren Zhou. 2025d. Masksearch: A universal pre-training framework to enhance agentic search capability . <i>Preprint</i> , arXiv:2505.20285.	Qi Zhang, Shouqing Yang, Lirong Gao, Hao Chen, Xiaomeng Hu, Jinglei Chen, Jiexiang Wang, Sheng Guo, Bo Zheng, Haobo Wang, and Junbo Zhao. 2025e. Lets: Learning to think-and-search via process-and-outcome reward hybridization . <i>Preprint</i> , arXiv:2505.17447.	831
776			832
777			833
778			834
779			835
780	Fengli Xu, Qian Yue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. 2025. Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models . <i>arXiv e-prints</i> , arXiv:2501.09686.	Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. 2025f. Process vs. outcome reward: Which is better for agentic rag reinforcement learning . <i>Preprint</i> , arXiv:2505.14069.	837
781			838
782			839
783			840
784			841
785			842
786			
787			843
788			844
789	An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report . <i>Preprint</i> , arXiv:2505.09388.	Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025g. Qwen3 embedding: Advancing text embedding and reranking through foundation models . <i>arXiv preprint arXiv:2506.05176</i> .	845
790			846
791			847
792			848
793			
794			849
795			850
796	Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering . In <i>Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing</i> , pages	Zhejun Zhao, Yuehu Dong, Alley Liu, Lixue Zheng, Pingsheng Liu, Dongdong Shen, Long Xia, Jiashu Zhao, and Dawei Yin. 2025. Tura: Tool-augmented unified retrieval agent for ai search . <i>Preprint</i> , arXiv:2508.04604.	851
797			852
798			853
799			
800			854
801			855
		Jiaru Zou, Dongqi Fu, Sirui Chen, Xinrui He, Zihao Li, Yada Zhu, Jiawei Han, and Jingrui He. 2025. Gtr: Graph-table-rag for cross-table question answering . <i>Preprint</i> , arXiv:2504.01346.	856
			857

858 A Related work 909

859 A.1 Architecture of Agentic RAG Systems 910

860 Recent advances in knowledge intensive reasoning 911
861 highlight the importance of system level innova- 912
862 tions in RAG. Beyond traditional linear pipelines, 913
863 emerging studies redesign the architecture and or- 914
864 chestration of RAG systems to achieve more adap- 915
865 tive, explainable, and robust agentic search. These 916
866 methods share a common goal: tightly integrating 917
867 reasoning, retrieval, and rereasoning through mod- 918
868 ular design, multi agent cooperation, and planning 919
869 driven workflows. 920

870 Early structural innovations such as IRCoT 921
871 (Trivedi et al., 2023) interleave retrieval with chain- 922
872 of-thought reasoning, allowing each reasoning 923
873 step to guide the next retrieval action. This idea 924
874 is extended in ComposeRAG (Wu et al., 2025c) 925
875 which decomposes the entire RAG pipeline into 926
876 atomic, composable modules question decompo- 927
877 sition, query rewriting, retrieval decisions, verifi- 928
878 cation, and answer synthesis thereby improving 929
879 flexibility and interpretability. 930

880 Multi agent frameworks further enhance robust- 931
881 ness. DRAG (Hu et al., 2025) introduces structured 932
882 debates supporter, opponent, judge across both re- 933
883 trieval and generation stages to reduce hallucina- 934
884 tions. MA-RAG (Nguyen et al., 2025) and MAO- 935
885 ARAG (Chen et al., 2025c) partition the RAG pro- 936
886 cess into specialized agents planner, step definer, 937
887 extractor, QA agents and orchestrate them through 938
888 multi step coordination, enabling adaptive multi 939
889 round refinement. 940

890 Another line of work emphasizes plan driven rea- 941
891 soning. PAR RAG (Zhang et al., 2025d) adopts a 942
892 “Plan–Act–Review” paradigm to prevent error prop- 943
893 agation in multi hop QA. Similarly, OPERA (Liu 944
894 et al., 2025) employs a Planner Executor architec- 945
895 ture where the goal planner decomposes tasks and 946
896 the reasoning execution module handles precise 947
897 retrieval trained via the MAPGRPO multi agent 948
898 optimization framework. 949

899 Systems also begin to model retrieval as an 950
900 explicit decision process. DeepRAG (Guan 951
901 et al., 2025) formulates RAG as an MDP, mak- 952
902 ing “whether, what, and how many times to re- 953
903 trieve” part of the system’s action space. PAIRS 954
904 (Chen et al., 2025b) follows a dual path strategy 955
905 that fuses parametric and retrieved knowledge to 956
906 dynamically determine when external evidence is 957
907 necessary, without additional training. 958

908 To broaden capabilities, several works integrate 959

909 tool usage or handle heterogeneous information 910
910 sources. TURA (Zhao et al., 2025) unifies static 911
911 corpus retrieval with dynamic API calls within a 912
912 single agent. Search-o1 (Li et al., 2025a) embeds 913
913 agentic search directly into long reasoning chains 914
914 and introduces a Reason-in-Documents module 915
915 for structured interpretation of retrieved evidence. 916
916 Meanwhile, MixRAG (Zhang et al., 2025a) and 917
917 GTR (Zou et al., 2025) extend RAG to heteroge- 918
918 neous documents including tables and graph struc- 919
919 tured data via hierarchical representations, hetero- 920
920 geneous graph construction, and layered retrieval. 921

921 As illustrated in the figure 6, we depict the typ- 922
922 ical structural characteristics of existing RAG ap- 923
923 proaches and further point out the key issues these 924
924 methods exhibit in practice. 925

925 A.2 Reinforcement Learning for RAG 926

926 A growing body of research strengthens agentic 927
927 RAG systems through reinforcement learning 928
928 which treats both reasoning and retrieval as learn- 929
929 able policies. These methods share the goal of 930
930 teaching models when to think, when to search, 931
931 how to search, and how to integrate retrieved 932
932 evidence, while differing in reward granularity 933
933 outcome-level vs. process-level, use of GRPO vari- 934
934 ants, memory mechanisms, and the modeling of 935
935 knowledge boundaries. 936

936 R-Search (Shi and Shen, 2025) employs GRPO 937
937 based reinforcement fine tuning to generate struc- 938
938 tured outputs (<think>, <search>, <result>, <an- 939
939 swer>) using multi component rewards for answer 940
940 correctness, DAG validity, and format compliance. 941
941 CR-Planner (Li et al., 2025b) uses dual critics to 942
942 guide subgoal planning and execution, while Deep- 943
943 RAG formulates retrieval decisions as an MDP, en- 944
944 abling adaptive policies for “whether and when to 945
945 retrieve.” MaskSearch (Wu et al., 2025d) strength- 946
946 ens general purpose search ability through a DAPO 947
947 based RL stage, and ZeroSearch (Sun et al., 2025a) 948
948 reduces API cost by replacing real retrieval with 949
949 simulated noisy documents under a curriculum- 950
950 based rollout strategy. EvolveSearch (Zhang et al., 951
951 2025b) further alternates SFT and RL in a self 952
952 improving loop that requires no human labeled rea- 953
953 soning traces. 954

954 The R1-Searcher (Song et al., 2025a) and Search- 955
955 R1 (Jin et al., 2025a) series, which explicitly train 956
956 LLMs to autonomously generate search queries 957
957 during reasoning. R1-Searcher adopts a two stage 958
958 pipeline SFT for format acquisition followed by 959
959 outcome based RL to encourage effective external

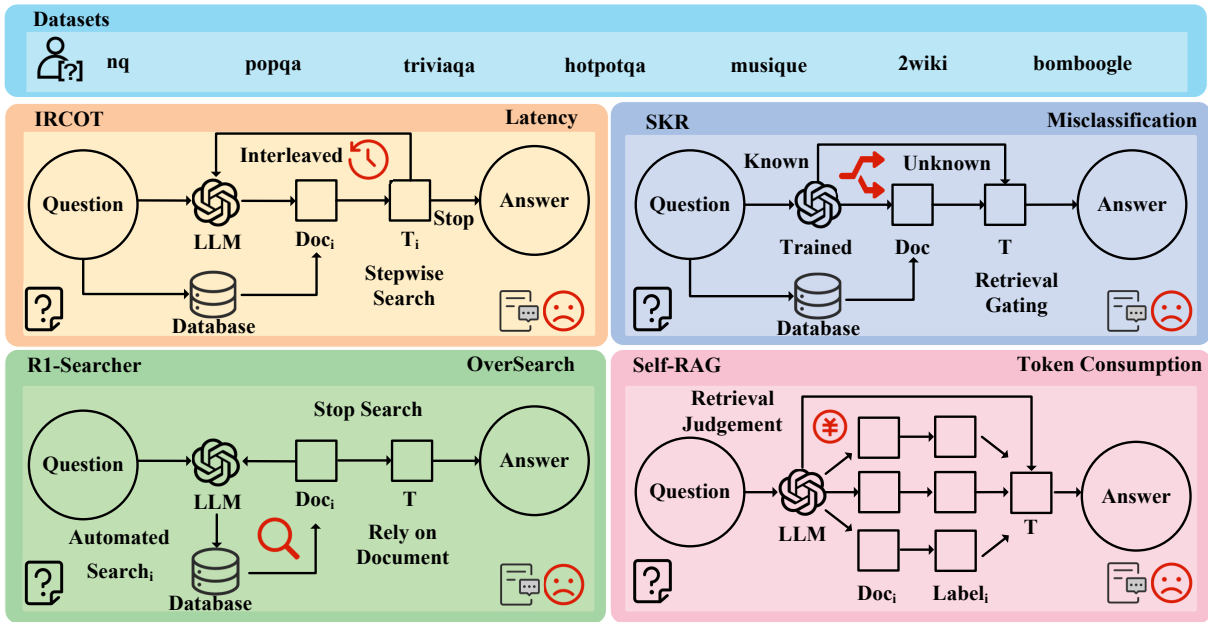


Figure 6: The overall framework and limitations of other methods.

search usage. R1-Searcher++ (Song et al., 2025b) extends this with a memory mechanism that accumulates and internalizes retrieved information over time. Search-R1 simplifies training with retrieved-token masking to stabilize RL, substantially outperforming strong RAG baselines across multiple QA datasets. Taken together, these works demonstrate the feasibility of training LLMs to actively plan and execute external search through lightweight reward signals.

FrugalRAG (Java et al., 2025) directly optimizes retrieval cost and shows that competitive RAG performance can be achieved with far fewer retrieval calls through lightweight SFT RL.IKEA which focuses on knowledge sufficiency and retrieval necessity (Huang et al., 2025) introduces a knowledge-boundary-aware reward and dataset that jointly encourage. This explicitly reframes retrieval as a conditional rather than default action. Similar ideas appear in FrugalRAG.

ReasonRAG which develops process level rewards (Zhang et al., 2025f) supervises intermediate steps query generation, evidence extraction, answer synthesis using process level rewards. LeTS (Zhang et al., 2025e) combines process level and outcome level signals to better supervise multi-step reasoning. ARENA (Ren et al., 2025) employs adaptive rewards and structured generation to train models to identify key evidence and produce interpretable decision traces.

RL has also been applied in domain-specific

and multimodal contexts. Domain-Aware (Lin et al., 2025) RAG uses GRPO to optimize domain adapted query and passage generation with efficient fusion strategies. MMSearch-R1 (Wu et al., 2025a) extends RL incentives to multimodal retrieval. Thai Legal GRPO demonstrates that GRPO can significantly improve citation accuracy and legal reasoning. ReSearch (Chen et al., 2025a) trains tool augmented reasoning agents in executable multi-step environments.

B Experiments Details

B.1 Datasets

We evaluate our approach on seven established open-domain question answering (ODQA) and multi-hop reasoning benchmarks:

Natural Questions (Kwiatkowski et al., 2019) is a large-scale dataset derived from real Google search queries, paired with Wikipedia articles and human-annotated long and short answers. It emphasizes answer localization within full documents under realistic retrieval conditions. TriviaQA (Joshi et al., 2017) consists of over 650k question-answer pairs authored by trivia enthusiasts, accompanied by evidence documents collected from the web or Wikipedia. It features high linguistic diversity and complex compositional questions, making it a strong testbed for generalization. PopQA (Mallen et al., 2023) is a dynamic, large-scale dataset constructed from Wikidata facts about popular entities (e.g., celebrities, movies), designed to assess mod-

Parameter	Value	Description
n	32	Number of sampled rollouts per prompt.
total_epochs	10	Maximum number of training epochs.
total_training_steps	500	Total number of RL training steps.
batch_size	32	Training batch size.
ppo_mini_batch_size	32	Mini-batch size used for policy updates.
ppo_micro_batch_size_per_gpu	1	Micro-batch size per GPU to control peak memory usage.
max_prompt_length	2048	Maximum length of the input prompt.
max_obs_length	4096	Maximum length of tool observations.
max_action_length	8192	Maximum length of tool actions.
max_response_length	16384	Maximum length of the generated response.
temperature	1.0	Sampling temperature for rollout generation.
top_p	1.0	Nucleus sampling threshold for rollouts.
max_turns	5	Maximum number of interaction turns per episode.
lr	1×10^{-6}	Learning rate for the actor (policy) optimizer.
kl_loss_coef	1×10^{-4}	Coefficient of the KL regularization term in the actor loss.
tensor_model_parallel_size	4	Tensor model parallelism degree for rollout inference.
gpu_memory_utilization	0.5	vLLM rollouts

Table 5: Experimental parameter design and training configuration used in our framework.

els’ internalized knowledge of time-sensitive and culturally relevant facts without external retrieval.

HotpotQA (Yang et al., 2018) requires systems to perform multi-hop reasoning across multiple Wikipedia paragraphs to answer questions and provide supporting facts. It includes both “distractor” and “fullwiki” settings to evaluate reasoning and retrieval jointly. 2WikiMultihopQA (Ho et al., 2020) is a synthetically generated multi-hop QA dataset built from structured paths in Wikipedia and Wikidata. It ensures logically sound two-hop questions with minimal noise, enabling controlled evaluation of reasoning capabilities. MuSiQue (Trivedi et al., 2023) constructs challenging multi-step questions by systematically composing single-hop questions, ensuring that answers cannot be inferred from any individual hop alone. It mitigates shortcut learning and enforces sequential reasoning. Bamboogle (Shen et al., 2023) is an adversarially designed multi-hop QA dataset where questions are crafted to prevent surface-level heuristics or keyword matching.

B.2 Models

We conduct experiments on the Qwen2.5 (Qwen et al., 2025) and Qwen3 (Yang et al., 2025) model families, and adopt Qwen3-235B-A22B-Instruct as the LLM-as-a-Judge to provide automatic evaluation signals. For the training targets, we fine-tune Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct as non-reasoning models, and Qwen3-4B and Qwen3-8B as reasoning-oriented models. All experiments are performed on eight H800 GPUs.

The choice of model scale is guided by a trade-off between computational feasibility and the qual-

ity of learning signals. Larger models incur prohibitive memory and computational costs, leading to severe GPU memory pressure, whereas smaller models suffer from insufficient internal reasoning and representation capacity. As a result, low-capacity models often fail to answer questions relying on their internal knowledge, which in turn yields sparse or ineffective reward signals during training. Therefore, we select medium-scale models to ensure stable and informative reward feedback under a practical computational budget.

B.3 Experimental Setup

During the training phase, we specify the detailed hyperparameter configuration in the Table 5. A validation check and checkpoint saving are performed every 30 optimization steps. The training data is sourced from the 2WikiMultihopQA (Ho et al., 2020) dataset, which we partition into two subsets: 80% is used for training within the reinforcement learning framework, and the remaining 20% serves as a validation set to monitor training dynamics. All experiments are conducted on eight NVIDIA H800 GPUs. For the reasoning model, each training iteration involves the generation of a full CoT, resulting in a total training duration of approximately 24 hours. In contrast, the non-reasoning model does not produce genuine CoT sequences; instead, it simulates highly abbreviated reasoning paths. Consequently, its training time is significantly reduced to about 6 hours, during which the model is already able to effectively explore and converge toward the optimal reward region. As shown in the figure 7, it demonstrates the changes in different metrics as the training steps progress.

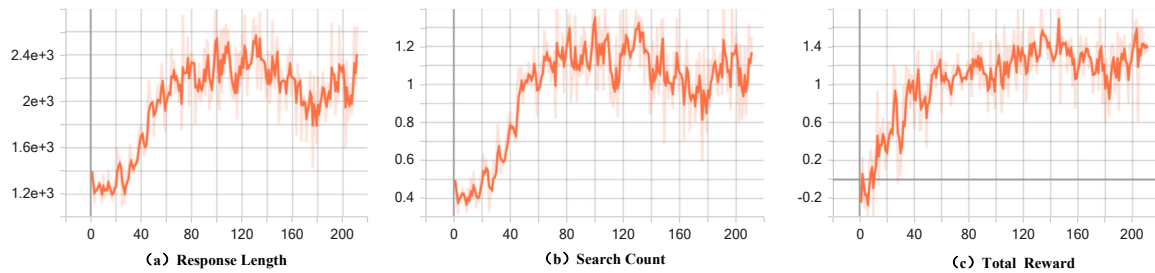


Figure 7: Trends of Evaluation Metrics Across Training Steps for Qwen3-4B.

B.4 Prompt Settings

Instruction for Judge LLM

You are an evaluation assistant. Your task is to determine whether the given prediction semantically matches any of the reference answers. This is a flexible match: even if the wording is different, if the meaning is the same, consider it a match. Prediction: normalized prediction Reference Answers: normalized golden Please answer with a single word: - YES → if the prediction expresses the same meaning as any of the reference answers. - NO → otherwise. Only respond with YES or NO.

Instruction for Trained RLM

Please answer the given question by following the structured reasoning protocol below:

1. Always begin with `<think>...</think>` to provide a brief internal reasoning sketch. - This reasoning must primarily rely on your internal knowledge and reasoning abilities. - If your internal knowledge is sufficient to answer the question, do not assume the need for external tools.
2. If your internal knowledge is sufficient to answer the question: - Provide a concise summary of your reasoning inside `<reflect>...</reflect>`. - Then give the final answer inside `<answer>...</answer>`. - Under no circumstances should you generate a `<search>` block if your internal knowledge is adequate.
3. Only if your internal knowledge is genuinely insufficient to answer the question: - Generate a brief keyword-based search query inside `<search>...</search>`. The query should be as concise as possible but must include sufficient factual clues, such as names, dates, locations, key concepts, etc. - The query should be a brief description with multiple keywords instead of a natural language question.
4. When the system returns information inside `<information>...</information>`: - If the information is reliable and sufficient, integrate it with your reasoning inside `<reflect>...</reflect>`. - Then provide the final concise answer inside `<answer>...</answer>`.
5. If the retrieved information is insufficient or contradicts your internal knowledge: - Explain the issue within `<reflect>...</reflect>`. - If necessary, generate a revised `<search>` query.
6. Final rule: - The final answer must be concise and clear, placed inside `<answer></answer>`. - Internal reasoning (in `<think>`) must always be the primary method; tool usage should be a last resort, not the default.

Table 6: An example from naive methods on 2wikimultihopqa dataset, with special symbols used in the thinking process. The thought words are highlighted in blue box and the correct answer is highlighted in green box respectively. The wrong answer is highlighted in red box.

Example #1 from 2wikimultihopqa Dataset

Question:

When Lavinia Norcross Dickinson's father died ?

Labeled Answer:

June 16, 1874

Direct Reasoning - Overthinking caused by high uncertainty

<think>Okay, I need to figure out when Lavinia Norcross Dickinson's father died. First, I remember that Lavinia Dickinson was one of the nineteen Dickinson sisters, right? The Dickinson family was pretty well-known in the 19th century, especially for their poet, Emily Dickinson.

But I'm not exactly sure about the dates of Lavinia's father or when he died. So, the Dickinson family was from Amherst, Massachusetts. Emily Dickinson was born in 1830, and her father was Edward Dickinson. Wait, but was Edward Dickinson the father of all the Dickinson sisters? I think he was the father of Emily, but maybe Lavinia was a different sister? Wait, no, the Dickinson family had several children, including Emily, and others like Lavinia.

...

But I need to check which of them were born when. For example, Emily was born in 1830, then her older sisters might have been born in the 1820s. Let me see if I can get the birth years of the Dickinson daughters. I think the children of Edward Dickinson were:

...

but I'm not entirely sure Alternatively, maybe Lavinia was born in 1825. If that's the case, then her father, Edward Dickinson, would have died when? Edward Dickinson died in 1887. Wait, but that's conflicting because Emily was born in 1830, and Edward died in 1887. So, if Lavinia Norcross Dickinson is one of the daughters of Edward Dickinson, then Lavinia's father died in 1887.

But I need to confirm this Alternatively, maybe Lavinia Norcross Dickinson is a different person. Wait, she might be the daughter of Henry Norcross, a different family. But that seems unlikely. Wait, but the name "Norcross" might be a surname of the Dickinson family. Wait, maybe Lavinia Norcross Dickinson was the daughter of Edward Dickinson, and her father's name is Edward Dickinson. But then, when did he die? Edward Dickinson was the father of Emily Dickinson, and he died in 1887. Therefore, if Lavinia was one of the daughters, then her father died in 1887.

...

Alternatively maybe Lavinia Norcross is a different person. Wait, but I think that in some sources, Lavinia Norcross Dickinson was the daughter of Edward Dickinson and Elizabeth Norcross Dickinson. Wait, but I need to get this right. Let me try to recall. The name "Norcross" is not commonly associated with the main Dickinson family. Wait, maybe I'm mixing up the names. Alternatively, perhaps Lavinia Norcross Dickinson is a different person, not a Dickinson daughter. Wait, but I think Lavinia Dickinson was one of the daughters, and I'm not sure about her full name. Maybe there's confusion with the fact that some of the Dickinson sisters were divorced, and their spouses might have taken surnames. For example, Lavinia Dickinson married a man named Sheldon, so her full name could be Lavinia Norcross Dickinson if her husband's surname was Norcross. Wait, that might be the case. So, Lavinia Norcross Dickinson was the daughter of Edward Dickinson, who died in 1887. Then her father died in 1887. Therefore, the answer is that Lavinia Norcross Dickinson's father died in 1887, when Edward Dickinson died.

But I should confirm this Let me check. Edward Dickinson was born in 1794 and died in 1887. Yes, that's correct. So, if Lavinia was his daughter, then her father died in 1887. Therefore, the answer is 1887.</think>

Answer: Lavinia Norcross Dickinson's father, Edward Dickinson, died on July 4, 1887

Standard RAG - Rely on the documents and Lack of internal reasoning

<think>Okay, let's tackle this question: When did Lavinia Norcross Dickinson's father die?

First, I need to figure out who Lavinia Norcross Dickinson's father is. From the documents provided, I see that Lavinia is the daughter of Edward Dickinson and Emily Norcross. Specifically, Doc 2 mentions that Lavinia was born to Edward Dickinson and his wife Emily Norcross. So, her father is Edward Dickinson.

Now, I need to find out when Edward Dickinson died. Looking through the documents, Doc 4 talks about Edward Dickinson. It states that he died on June 16, 1874. The document also mentions that he died after a speech in the House on his connection with the Hoosac Tunnel. Additionally, Doc 1 mentions that Emily Norcross Dickinson (Lavinia's mother) suffered a stroke in 1874 and died in 1882. However, the death date of Edward Dickinson is clearly stated in Doc 4 as June 16, 1874.

So, putting it all together, Lavinia Norcross Dickinson's father, Edward Dickinson, died on June 16, 1874. The answer should be that date.</think>

Lavinia Norcross Dickinson's father was Edward Dickinson, who died on June 16, 1874 This is explicitly stated in Doc 4, which notes his death date and provides context about his life and career. The information is corroborated across documents, including references to his role as treasurer of Amherst College and his eventual demise from apoplexy during a speech in the House.

Table 7: An example from Our framework on 2wikimultihopqa dataset, with special symbols used in the thinking process. the correct answer is highlighted in green box respectively.

Example #1 from 2wikimultihopqa Dataset
<p>Question: When Lavinia Norcross Dickinson's father died ?</p> <p>Labeled Answer: June 16, 1874</p>
Ours - Integration of internal reasoning and external information
<p><think> Okay, let's see. The question is when Lavinia Norcross Dickinson's father died. First, I need to find out who Lavinia Norcross Dickinson was. From what I remember, she was an American author and writer. She's known for her letters and diaries. Now, I need to find out who her father was. Lavinia Norcross Dickinson was married to Edmond Dickinson. Wait, but that's her husband. Her father's name... I think her father was Samuel Norcross. Wait, but I need to confirm that. Wait, Lavinia Norcross Dickinson was born in 1830. Her father's death date would be somewhere in her early life. Wait, I recall that Samuel Norcross died in 1834. Because Lavinia was born in 1830, so if her father died when she was four years old, that would put the date around 1834. But I need to be sure. Let me make a search.</p> <p></think></p> <p><search> when did Lavinia Norcross Dickinson's father die</p> <p></search></p> <p><information> Doc 1 (Title: ""Emily Norcross Dickinson"") Emily Norcross Dickinson (née Norcross; July 3, 1804 – November 14, 1882) was a member of the Dickinson family of Amherst, Massachusetts, and the mother of American poet Emily Dickinson... Doc 2 (Title: ""Lavinia Norcross Dickinson"") Lavinia "Vinnie" Norcross Dickinson (February 28, 1833 – August 31, 1899) was the younger sister of American poet Emily Dickinson. Vinnie was the youngest of the Dickinson siblings born to Edward Dickinson and his wife Emily Norcross in Amherst... Doc 3 (Title: ""Fred Norcross"") They had two children, Fred Stevenson Norcross III (1926–1990) and Robert Jackman Norcross (1929–2009)... Doc 4 (Title: ""Edward Dickinson"") Edward Dickinson (January 1, 1803 – June 16, 1874) was an American politician from Massachusetts. He is also known as the father of the poet Emily Dickinson; their family home in Amherst, the Emily Dickinson Museum, is a museum dedicated to her... Doc 5 (Title: ""Emily Dickinson"") Two hundred years earlier, her patrilineal ancestors had arrived in the New World—in the Puritan Great Migration—where they prospered. Emily Dickinson's paternal grandfather, Samuel Dickinson, was one of the founders of Amherst College...</p> <p></information></p> <p><reflect> The question is about when Lavinia Norcross Dickinson's father died. From Document 4, we learn that Edward Dickinson (father of Emily Dickinson) died on June 16, 1874. Since Lavinia Norcross Dickinson was born on February 28, 1833 (Document 2), her father Edward Dickinson was alive when she was born. Therefore, his death date of June 16, 1874, is the answer.</p> <p></reflect></p> <p><answer> June 16, 1874</p> <p></answer></p>

Table 8: An example from naive methods on 2wikimultihopqa dataset, with special symbols used in the thinking process. The thought words are highlighted in blue box and the position of correct answers are highlighted in green box respectively. The first wrong thought is highlighted in red box.

Example #2 from 2wikimultihopqa Dataset
<p>Question: What is the date of birth of Frederick of Liège’s father?</p> <p>Labeled Answer: 1027</p>
Direct Reasoning - Overthinking caused by high uncertainty
<p><think>Okay, I need to find the date of birth of Frederick of Liège’s father. First, I should figure out who Frederick of Liège is. From what I remember, Frederick of Liège was a medieval figure, maybe a duke or a noble. Let me start by confirming his identity</p> <p>...</p> <p>Alternatively, I might have mixed up the names. Maybe the father of Frederick of Liège was Henry IV of Liège. But I’m not certain. Let me think again. The House of Liège had several dukes. The first duke was William the Silent (1025-1057), then his son William II (1057-1071), then Henry I (1071-1110), Henry II (1110-1131), Henry III (1131-1163), Henry IV (1163-1186), Henry V (1186-1216), and so on</p> <p>...</p> <p>Alternatively, there might be a Frederick of Liège who was a 16th-century duke. That’s possible. Wait, the name Frederick of Liège might be associated with the 15th or 16th century. For example, Frederick the Lion, but he was a king. Wait, perhaps Frederick of Liège was a lesser noble, but the dukes of Liège are from the 11th century onwards.</p> <p>...</p> <p></think></p> <p>Answer: The date of birth of Frederick of Liège’s father, Henry IV of Liège, is 10 June 1087.</p>
Standard RAG - Overthinking caused by insufficient information
<p><think>Okay, let’s tackle this question: "What is the date of birth of Frederick of Liège’s father?"</p> <p>First, I need to figure out who Frederick of Liège is. The user mentioned "Frederick of Liège," but looking at the external documents, there’s a document titled "Floribert of Liège" which says he was the son of Hubert of Liège. Wait, the question is about Frederick of Liège’s father. But the documents don’t mention a Frederick of Liège directly. Let me check again. Looking at the documents, Doc 1 talks about Floribert of Liège, who was the son of Hubert of Liège. The question is about Frederick of Liège’s father. Maybe there’s a confusion here. The user might be mixing up names. Alternatively, maybe there’s a Frederick mentioned in another document.</p> <p>...</p> <p>Alternatively, maybe the user is referring to Frederick of Denmark (Doc 4), but again, that’s not Liège. Since the documents don’t mention a Frederick of Liège, and the only related person is Floribert, whose father is Hubert, but no birth date for Hubert. Therefore, the answer would be that there’s no information in the provided documents. However, the user might have made a mistake, but since the assistant is supposed to use the documents, maybe the answer is that there’s no such person in the documents.</p> <p>Alternatively, perhaps the user intended to ask about Floribert of Liège’s father, Hubert, but the question says Frederick. Given that, perhaps the assistant should note that there’s no Frederick of Liège in the documents, but if it’s a mistake, then Hubert of Liège’s birth date is unknown. However, the assistant must rely on the documents.</p> <p>...</p> <p><think/></p> <p>Final Answer: The provided documents do not contain the date of birth of Frederick of Liège’s father.</p>

Table 9: An example from Our framework on 2wikimultihopqa dataset, with special symbols used in the thinking process. The position of correct answers are highlighted in green box respectively.

Example #2 from 2wikimultihopqa Dataset
<p>Question: What is the date of birth of Frederick of Liège's father?</p> <p>Labeled Answer: 1027</p>
Ours - Learn to perform multiple retrievals
<p><think> Okay, let's see. The question is asking for the date of birth of Frederick of Liège's father. First, I need to figure out who Frederick of Liège was. From what I remember, Frederick of Liège was a historical figure, probably a noble or a ruler. He was the Duke of Liège, I think. Liège is a city in Belgium, so maybe he was a duke there. Now, I need to find out who his father was. Typically, dukes have parents, so his father would be the previous duke or another prominent figure. Let me recall. Frederick of Liège was born in the 13th century, maybe? Wait, I think he was born around 1224. But wait, that's his birth date, not his father's. Wait, no. The question is about his father's date of birth. So first, I need to confirm who Frederick of Liège was. Let me think. Frederick of Liège was the Duke of Liège from 1248 until his death in 1296. He was the son of Duke Henry I of Liège. Wait, yes, Henry I was the Duke before him. So if Frederick was the son of Henry I, then Henry I would be his father. So now I need to find the date of birth of Henry I of Liège. From what I remember, Henry I was born around 1203. But I need to be sure. Wait, Henry I of Liège was born on December 16, 1203. He was the father of Frederick. Therefore, Frederick's father was Henry I, born December 16, 1203. But I need to confirm this. Wait, but maybe I'm mixing up different Henrys. Let me check my memory. Henry I of Liège was indeed the father of Frederick. He was born in 1203. So the date of birth of Frederick's father would be December 16, 1203. Therefore, the answer should be December 16, 1203. </think> <search> Frederick of Liège father's date of birth </search> <information> Doc 1 (Title: ""Floribert of Liège"") Doc 2 (Title: ""Frederick III of Austria"") Doc 3 (Title: ""Frederick II, Duke of Upper Lorraine"") Doc 4 (Title: ""Frederick III, Duke of Upper Lorraine"") Doc 5 (Title: ""Rainald III of Toul"")... </information> <reflect> The question asks for the date of birth of Frederick of Liège's father. The provided documents do not mention Frederick of Liège. Document 1 refers to Frederick of Verdun, document 2 to Frederick of Antioch, document 3 to Floribert of Liège, document 4 to Frederick III of Upper Lorraine, and document 5 to Frederick II of Upper Lorraine. None of these documents provide information about Frederick of Liège or his father's birth date. Therefore, the information required is not available in the provided documents. </reflect> <search> Albert III, Count of Namur birth date </search> <information> Doc 1 (Title: ""Albert II, Count of Namur"") Doc 2 (Title: ""Albert III, Count of Namur"") Doc 3 (Title: ""Robert II, Count of Namur"") Doc 4 (Title: ""Albert III, Prince of Anhalt-Zerbst"") Doc 5 (Title: ""Albert IV, Count of Habsburg"")... </information> <reflect> The question asks for the date of birth of Frederick of Liège's father. From Document 2, Frederick (Bishop of Liege) was the son of Albert III, Count of Namur. Document 2 also states that Albert III was born in 1027 and died on 22 June 1102. Therefore, the date of birth of Frederick of Liège's father is 1027. </reflect> <answer> 1027 </answer> </p>