From Sufficiency to Reflection: Reinforcement-Guided Thinking Quality in Retrieval-Augmented Reasoning for LLMs

Anonymous ACL submission

Abstract

Reinforcement learning-based retrieval-augmented generation (RAG) methods enhance the reasoning abilities of large language models (LLMs). However, most rely only on final-answer rewards, overlooking intermediate reasoning quality. This paper analyzes existing RAG reasoning models and identifies three main failure patterns: (1) *information insufficiency:* failure to retrieve adequate support; (2) *faulty reasoning:* logical or content-level flaws despite sufficient information; (3) *answer–reasoning inconsistency:* a valid reasoning chain with a mismatched final answer. We propose **TIRESRAG-R1**, a novel framework using a *think–retrieve–reflect* process and a multi-dimensional reward system to improve reasoning and stability. TIRESRAG-R1 introduces: (1) a *sufficiency reward* to encourage thorough retrieval; (2) a *reasoning quality reward* to assess rationality and accuracy of the reasoning chain; (3) a *reflection reward* to detect and revise errors. It also employs a *difficulty-aware reweighting strategy* and *training sample filtering* to boost performance on complex tasks. Experiments on four multi-hop QA datasets show TIRESRAG-R1 outperforms prior RAG methods and generalizes well to single-hop tasks.

1 Introduction

Large language models (LLMs) (Grattafiori et al., 2024; Yang et al., 2025; OpenAI et al., 2024a) have achieved remarkable performance across a wide range of downstream tasks, such as mathematical reasoning (Ahn et al., 2024), code generation (Jiang et al., 2025), open-domain question answering (Kamalloo et al., 2023). A key factor behind this success is *chain-of-thought prompting* (Wei et al., 2022), which enables LLMs to generate intermediate reasoning steps before arriving at a final answer, significantly enhancing performance

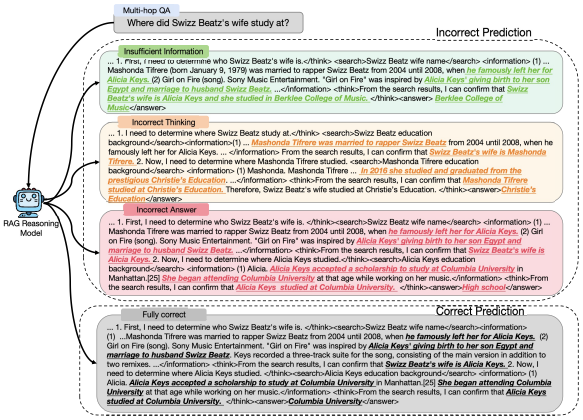

Figure 1: An example showing different reasoning trajectories for answering a multi-hop query. It compares insufficient information, incorrect predictions, and fully correct reasoning.

on reasoning tasks (Wang et al., 2023; Pan et al., 2023; Snell et al., 2025). Despite these advances, the internal knowledge of LLMs is not always reliable. For instance, when faced with time-sensitive queries (Mousavi et al., 2024) or conflicting evidence within their internal representations (Xu et al., 2024), LLMs often produce hallucinations or factual inaccuracies due to outdated or ambiguous information (Li et al., 2024a; Wang et al., 2024b).

The *retrieval-augmented generation (RAG)* paradigm (Gao et al., 2024a) improves factual accuracy and robustness by enabling LLMs to access up-to-date external knowledge. However, standard RAG often treats retrieval and generation as loosely coupled, lacking mechanisms for multi-step reasoning such as query decomposition or identifying knowledge gaps, which limits performance on tasks requiring deeper reasoning.

To overcome this, a growing body of research has turned to *reinforcement learning (RL)* (a paradigm that has demonstrated strong performance in mathematical reasoning and code generation (Shao et al., 2024; Wang et al., 2024a)) to train

LLMs for retrieval-augmented reasoning. In this setting, the model learns when and how to retrieve, as well as how to integrate retrieved information into coherent reasoning chains, guided by outcome-level rewards (e.g., answer correctness) (Jin et al., 2025a; Song et al., 2025; Sun et al., 2025a). Despite recent progress, outcome-based RL methods still face two significant limitations: (1) they often neglect the quality and validity of intermediate reasoning steps, and (2) they lack fine-grained feedback to guide the reasoning process during multi-step tasks. As a result, models may learn incorrect reasoning paths that degrade answer quality, or they may arrive at correct answers through flawed reasoning, thereby compromising interpretability.

In this paper, to address these challenges, we begin by evaluating the performance of RAG-based reasoning models trained with outcome-based rewards. Through human analysis on examples from multi-hop reasoning datasets (e.g., MuSiQue (Trivedi et al., 2022)), as shown in Fig. 1, we identify three primary bottlenecks limiting model accuracy: (1) the model retrieves sufficient information but still fails to produce the correct answer; (2) reasoning is prematurely interrupted, resulting in inadequate retrieval; and (3) the reasoning chain is correct, yet the final answer is incorrect. Based on these insights, we propose **TIRESRAG-R1**, an approach that enhances reasoning chain quality by incorporating rewards focused on sufficiency and reasoning coherence. Additionally, we introduce a *thinking–retrieval–reflection* framework, motivated by the principle that models should first ensure the integrity of their reasoning chains. If the final answer is incorrect despite a valid chain, the model engages in reflection to revise its response.

TIRESRAG-R1 enhances RAG-based reasoning by introducing a thinking–retrieval–reflection framework, where the model can decide post-answer whether to reflect and revise its output through an additional round of reasoning and retrieval. To guide learning, TIRESRAG-R1 assigns multi-dimensional reward signals to each reasoning trajectory, including: an answer reward, a sufficiency reward to encourage retrieval of adequate evidence, a reasoning quality reward evaluating logical coherence, alignment, error awareness, and conciseness, and a reflection reward that promotes correction of wrong answers while discouraging unnecessary changes. To address the varying difficulty of retrieval tasks, we propose a difficulty-aware advantage reweighting strategy that adjusts reward weights based on the ease of retrieving sufficient evidence, and an advantage filtering mechanism that excludes trivial examples (where all responses are correct or incorrect) to stabilize RL training. Together, these components enable more robust and interpretable reasoning, significantly outperforming traditional outcome-based methods across multiple QA benchmarks.

Our contributions can be summarized as follows:

- We are the first to define *overthinking* and *underthinking* in RAG and conduct comprehensive analysis to reveal the main bottlenecks faced by current RL-trained RAG reasoning models.
- We propose **TIRESRAG-R1**, which enhances reasoning by introducing a reflection mechanism and fine-grained multi-dimensional rewards to improve both reasoning chains and answer accuracy. We also propose a difficulty-aware advantage reweighting strategy with no additional computational cost to guide the model toward optimizing on harder questions.
- Our experiments show that TIRESRAG-R1 outperforms multiple state-of-the-art RAG methods on several open-ended QA datasets, with an average accuracy improvement of 5.8%. Further ablation studies and analysis confirm the effectiveness of our fine-grained rewards.

2 Related Work

Retrieval-Augmented Generation. RAG has been widely adopted to mitigate hallucination, domain knowledge gaps, and temporal staleness (Ayala and Bechard, 2024; Siriwardhana et al., 2023; Gade and Jetcheva, 2024). Traditional RAG typically follows a static retrieve-then-generate pipeline (Lewis et al., 2020; Guu et al., 2020), which is effective for open-domain QA but often struggles with multi-hop reasoning, latent constraints, and ambiguous queries (Tang and Yang, 2024; Li et al., 2024b; Chan et al., 2024). Recent work has proposed more cognitively inspired architectures, such as AdaptiveRAG (Jeong et al., 2024) using query classification to choose retrieval strategies. In parallel, modular and hybrid frameworks (Gao et al., 2024b; Zhang et al., 2024; Zhou et al., 2024) integrate evidence aggregation, verification, and query rewriting, reflecting a deeper interaction between retrieval and reasoning.

Reinforcement Learning for LLM Reasoning. Reinforcement learning (RL) has recently been explored as a way to improve multi-step reasoning in LLMs. Models such as GPT-o1 and DeepSeek-R1 (OpenAI et al., 2024b; DeepSeek-AI et al., 2025) demonstrate RL-based training for structured reasoning. Follow-up work (e.g., SimpleRL-Zoo (Zeng et al., 2025), Open-Reasoner-Zero (Hu et al., 2025c), and Light-R1 (Wen et al., 2025)) further leverage curriculum design, reward shaping, and improved optimization algorithms such as Dr GRPO (Liu et al., 2025) to enhance verifiability and coherence.

Reinforcement Learning for RAG Reasoning. Recent studies combine RL with RAG to dynamically guide when and what to retrieve (Song et al., 2025; Jin et al., 2025a; Chen et al., 2025). Early approaches rely primarily on outcome-level rewards, while newer ones introduce process-level rewards (Wang et al., 2025; Sha et al., 2025; Zhang et al., 2025b). For instance, R3-RAG (Li et al., 2025b) computes document relevance at each step to refine search strategies, and others enrich the "search–think–answer" pipeline with additional fields to prompt deeper reflection (Ren et al., 2025; Shi et al., 2025). In contrast, our work directly rewards the model's reasoning process by measuring the sufficiency of retrieved documents, enabling more adaptive retrieval-reasoning coordination.

For more details on related work, please see Appendix A.

3 Prelimary Study

Although R1-like RAG models demonstrate stronger reasoning capabilities than traditional RAG models in QA tasks, questions remain about the faithfulness of their reasoning processes, both in relation to the provided context and in how well their final answers reflect that reasoning. This section focuses on the following central question: *In current reasoning-oriented RAG models tackling multi-hop tasks, what is the correlation between the predicted answers, the retrieved documents, and the generated reasoning text?*

3.1 Preliminary Analysis

We trained a reasoning-capable RAG model using the GRPO algorithm from DeepSeek-R1, following the data setup of R1-Searcher. We used Qwen-2.5-3B-Instruct with a local retriever and evaluated on in-domain 2Wikimultihopqa (2Wiki) and out-of-domain Musique test sets.

Category \ Datasets	2Wiki		MuSiQue	
	Corr.	Incorr.	Corr.	Incorr.
Overthinking	13	25	13	31
Good thinking	232	89	80	96
Underthinking	16	125	0	280

Table 1: Distribution of correct (Corr.) and incorrect (Incorr.) predictions across different reasoning categories on 2Wiki and MuSiQue datasets.

In what follows, we will denote an example as $E = (Q, RD, A)$, where Q is the query, RD the combined reasoning steps and retrieved documents, and A the predicted or gold answer. Regarding reasoning behavior, we categorize the model's reasoning into three cases: **overthinking** (gold answer could be inferred before the final step), **good thinking** (gold answer emerges exactly at the final step), and **underthinking** (model stops without sufficient content). Following prior work (Joren et al., 2025), we use an LLM to assess whether Q and RD are sufficient to derive the gold answer, but unlike (Joren et al., 2025), we also provide the gold answer to reduce misjudgments. Details of fine-tuning setup and experimental analysis settings are all provided in Appendix J.

We report sufficiency judgment results separately based on whether the model's prediction is correct. The results are shown in Table 1. We can observe that among correctly predicted examples, overthinking and underthinking account for very small proportions, while good thinking accounts for 86% in Musique and 89% in 2Wiki. This indicates that for correctly predicted examples, the model's reasoning process is generally just right and sufficient. Among incorrectly predicted examples, underthinking accounts for a large proportion. This is unsurprising, as the model did not retrieve relevant documents and thus lacked enough information to answer the question, so answering incorrectly is expected. Surprisingly, overthinking and good thinking together account for 48% in Musique and 31% in 2Wiki. This shows that the reasoning process was sufficient, but the model still failed to produce the correct answer.

3.2 Fine-grained Analysis

Since overthinking accounts for a small proportion among predictions, we further analyzed the incorrectly predicted examples belonging to good thinking and overthinking to determine specific er-

Context Sufficiency	Error type	Case	Ratio
Insufficient	Knowledge retrieved incorrect	**Question:** What is the place of birth of the founder of Leister Technologies? **Model output:** ... <search> founder of Leister Technologies </search> ... <think> Louis Leitz (founder of Esselte Leitz) ... therefore answer Ingersheim </think> <answer> Ingersheim </answer> **Gold Answer:** Wilhelmshaven	30.2%
Insufficient	Thinking incorrect	**Question:** Where was the place of burial of John De Ferrers, 1st Baron Ferrers of Chartley's father? **Model output:** ... <search> John De Ferrers, 1st Baron Ferrers of Chartley father's burial place </search> ... <think> Linked to Sir Humphrey Ferrers of Tamworth Castle and concluded burial at Tamworth Church </think> <answer> Tamworth Church </answer> **Gold Answer:** Staffordshire	22.1%
Sufficient	Thinking process mismatch evidence	**Question:** Who is Gaius Rubellius Blandus's father-in-law? **Model output:** ... <search> Gaius Rubellius Blandus father-in-law </search> ... <think> Married Julia (granddaughter of Tiberius) ⇒ therefore father-in-law is Tiberius </think><answer> Tiberius </answer> **Gold Answer:** Drusus Julius Caesar	16.4%
Sufficient	Thinking correct but incorrect answer	**Question:** Which film has the director died earlier, Girls School Screamers or Lucky Numbers? **Model output:** ... <search> director of Girls School Screamers </search> ... <search> director of Lucky Numbers </search> ... <think> Found John P. Finnegan (died 2007) and Nora Ephron (died 2012) ⇒ therefore the director of Lucky Numbers director died earlier </think> <answer>Girls School Screamers</answer> **Gold Answer:** Lucky Numbers	5.7%
Sufficient	Thinking process match evidence but thinking process is incorrect	**Question:** Which film has the director who was born first, Nasib Si Labu Labi or Solo Per Te Lucia? **Model output:** ... <search> director of Nasib Si Labu Labi birth year </search> ... <search> director of Solo Per Te Lucia birth year </search> ... <think> P. Ramlee (born 1936) vs Franco Rossi (born 1923) ⇒ therefore Nasib Si Labu Labi director was born first </think><answer> Nasib Si Labu Labi </answer> **Gold Answer:** Solo Per Te Lucia	25.6%

Table 2: Fine-grained error analysis with examples.

ror causes. We manually reviewed these examples and classified them into five error types, as shown in Table 2.

We observe the following. When the reasoning process was insufficient to fully answer the question, we found two error types: (i) the model mistakenly believes the retrieved content matches the question entity and therefore stops further retrieval and outputs an answer, while in fact the retrieved content is irrelevant; (ii) the model only obtained partial content relevant to the question and ended the reasoning, which is a reasoning omission. When the reasoning process was sufficient to answer the question, a prominent problem was that the model failed to follow evidence, indicating difficulty in understanding retrieved content and integrating it into reasoning (Shi et al., 2025; Li et al., 2025a). Another issue was that the reasoning process was correct but the final answer was wrong, or the model showed correct evidence but failed to reason correctly from it.

Based on these findings, we categorize the failure cases of RAG reasoning on part of the test sets into three major types: **retrieval failure**, **reasoning failure**, and **answer failure**. Motivated by this observation, we propose **TIRESRAG-R1**, which aims to enhance the model's awareness of retrieval sufficiency while improving both its reasoning chain and final answer quality.

4 Method

In this section, we first introduce our GRPO-based *thinking–search–answer–reflect* pipeline. Then, we elaborate on the reward design, which is the core of our RL algorithm. On top of the standard answer reward, we carefully add three additional reward signals: *thinking reward*, *sufficient reward*, and *reflect reward*. Finally, we observe that during training there exists a class of "extreme" examples that are either answered correctly by the model in all rollouts or answered incorrectly in all rollouts. Such examples contribute very limited training signals and may even introduce noise, affecting training stability. To address this issue, we introduce two optimization mechanisms: a *difficulty-aware advantage reweighting strategy* and a *group filtering mechanism* that filters out those extreme problems. A complete algorithmic description can be found in Appendix K.

4.1 Trajectory Generation with Search and Reflect

Given a question q, we first prompt the LLM π_θ to generate a long chain of thought. During this thinking process, the model is encouraged to trigger search operations in order to use external document information. When the model decides that a search is needed, it terminates the current thinking step. The reasoning text so far is wrapped with `<think> </think>` tags, and the search query is wrapped with `<search> </search>` tags. We extract the search query and feed it into the retriever π_{ret} to obtain k relevant documents $D_k = \{d_1, \ldots, d_k\}$. These documents are wrapped with `<information> </information>` tags and appended back to the trajectory. The model then

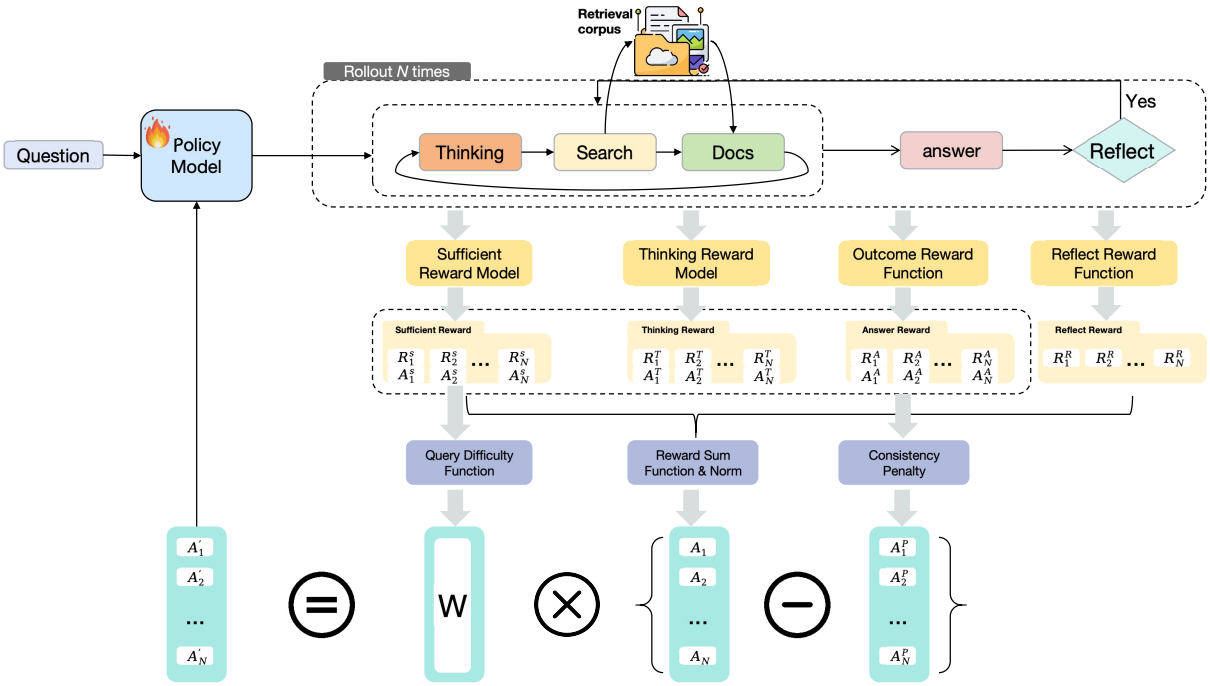

Figure 2: Overall architecture of TIRESRAG-R1, which integrates search, reasoning, and reflection with a dynamic reward design to guide reinforcement learning.

continues reasoning based on the retrieved documents. This process of thinking and searching repeats until the model judges that enough information has been collected to generate the final answer. At that point, the answer is wrapped with `<answer> </answer>` tags. However, according to our prompt design, the model is instructed to reflect on the generated answer, potentially entering another cycle of thinking and searching before producing a second answer wrapped again with `<answer> </answer>`. This reflection mechanism is specifically introduced to address the issue discussed in Section 3, namely that the reasoning process may be correct but the first generated answer is wrong. In the final trajectory, the content inside the last `<answer> </answer>` is taken as the model's predicted answer.

4.2 GRPO for Training

As shown in Figure 2, we adopt the *group relative policy optimization (GRPO)* algorithm (DeepSeek-AI et al., 2025) for RL training. For each query in GRPO, a group of G rollout trajectories, as described in Section 4.1, is generated using the current policy π_θ^{old}. Here π_θ^{old} also serves as a frozen reference model initialized with the same parameters as the policy model. The GRPO algorithm uses the following optimization objective to update the policy:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim Q,\{o_i\}_{i=1}^{G}\sim\pi_{\text{old}}}\left[\frac{1}{G}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\min\left(\frac{\pi_\theta(o_{i,t}\mid q)}{\pi_{\theta_{\text{old}}}(o_{i,t}\mid q)}A_i,\text{clip}\left(\frac{\pi_\theta(o_{i,t}\mid q)}{\pi_{\theta_{\text{old}}}(o_{i,t}\mid q)},1-\epsilon,1+\epsilon\right)A_i\right)\right]$$

where ϵ is the clipping hyper-parameter and β is the KL-divergence penalty coefficient. The advantage A_i for each response is computed as $A_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^{G})}{\text{std}(\{r_j\}_{j=1}^{G})}$, where $\{r_i\}_{i=1}^{G}$ are the rewards from the group. The detailed definition of each reward is introduced in Section 4.3.

4.3 Reward Design

As described in Section 3, we aim to achieve three major goals: (1) encourage the policy model to search sufficiently before generating an answer, (2) improve the quality of the reasoning chain and make it more consistent with the retrieved information, and (3) enhance the model's ability to correct answers through reflection. To achieve these goals, we adopt a mixed reward function with the following components:

Answer reward measures the match between the predicted answer and the gold answer. Following prior work (Jin et al., 2025a), we use the F1 score as the answer reward to avoid reward hacking issues observed with exact match (EM):

$$R^A = F_1(a, a^*). \quad (1)$$

Sufficient reward measures whether the reasoning trajectory RD provides enough information to support the gold answer. Following Section 3, we use a locally deployed LLM to score $(Q, RD, \text{gold answer})$ with 0/1, where 1 means sufficient and 0 means insufficient:

$$R^S = \begin{cases} 1.0, & \text{if } (Q, RD, o_g) \text{ is sufficient,} \\ 0.0, & \text{otherwise.} \end{cases} \quad (2)$$

Thinking reward measures the quality of the thinking part of the trajectory. The locally deployed LLM is prompted to evaluate logical soundness, alignment with retrieved content, error awareness, and concise accuracy. It outputs a score in $[0, 1]$:

$$R^T \in [0, 1]. \quad (3)$$

Reflect reward encourages the model to revise wrong answers into correct ones. We extract each intermediate answer $a_1, a_2, \ldots$ from the trajectory, compare each with the gold answer using an accuracy score, and assign rewards as follows: a positive reward when reflection corrects an incorrect answer, a negative reward when it replaces a correct answer with an incorrect one, and zero otherwise.

$$R^R = \begin{cases} +1.0, & \text{if } CEM(a_1, a^*) = 0 \text{ and } CEM(a_2, a^*) = 1, \\ -1.0, & \text{if } CEM(a_1, a^*) = 1 \text{ and } CEM(a_2, a^*) = 0, \\ 0.0, & \text{otherwise.} \end{cases} \quad (4)$$

Dynamic Weight of Reward. Because of the mixed reward mechanism, the model could overfit to auxiliary signals such as sufficient reward or thinking reward and ignore answer accuracy. We design a dynamic weight schedule $a_t = \frac{1}{1+\exp\left(\frac{t-0.9T}{10}\right)}$, where t denotes the current training step and T denotes the total number of training steps. This schedule gradually shifts focus from auxiliary reasoning rewards to answer accuracy as training proceeds.

Finally, the overall reward is computed as

$$R^{\text{sum}} = R^A + a_t\big(w_t R^T + w_s R^S + w_r R^R\big), \quad (5)$$

4.4 Optimization Strategy

Difficulty-aware Resampling. Despite employing a mixed reward mechanism, we observed that different rollout groups can yield very different raw rewards but almost identical normalized advantages (e.g., $[0.8, 0.85, 0.9, 0.95, 1.0]$ vs. $[0.0, 0.05, 0.1, 0.15, 0.2]$). This phenomenon causes simple problems to receive the same optimization emphasis as hard problems. Following Zhang and Zuo (2025), we introduce a difficulty-aware advantage reweighting strategy. For each sample, we estimate the problem difficulty by computing the average sufficient-reward score across all its rollouts, denoted as R_S^{avg}. The weight function is defined as

$$W(R^S_{\text{avg}}) = A + \frac{B - A}{1 + \exp\big[k(R^S_{\text{avg}} - \rho_0)\big]}, \quad (6)$$

where A, B, ρ_0, k are tunable hyper-parameters controlling the sensitivity of the reweighting. R^s_{avg} is the average sufficient-reward score of all rollouts associated with question q.

Consistency Penalty. Because we use a mixed reward mechanism, it is possible for a trajectory to receive a high reasoning reward but a low answer reward. To encourage consistent trajectories where high reasoning quality aligns with high answer accuracy, we add a small penalty term to discourage such inconsistencies:

$$A_i^P = -\lambda_p A_i^T \cdot A_i^T \cdot A_i^A, \quad \text{if } A_i^S \cdot A_i^T \cdot A_i^A < 0. \quad (7)$$

where A_i^S, A_i^T, and A_i^A denote the group-normalized values of the sufficient reward, the thinking reward, and the answer reward, respectively. The final difficulty-aware advantage is then expressed inline as $A_i' = (A_i - A_i^P) \cdot W(R^s_{\text{avg}})$, where A_i is the original normalized advantage for rollout i.

Group Filtering. As noted in (Jin et al., 2025a), applying GRPO to RAG tasks can lead to late-stage collapse, where training rewards drop to zero. We also observe this phenomenon: in later stages, many queries have groups where all rollouts are either completely correct or completely wrong, leading to zero advantage and noisy updates. To mitigate this, we introduce a simple filtering mechanism that removes these saturated queries from the training set.

5 Experiments

Experimental Setup. Following the evaluation protocol of R1-Searcher, we assess performance on four widely used and challenging multi-hop QA datasets: **HotpotQA** (Yang et al., 2018), **2WikiMultiHopQA** (Ho et al., 2020), **Bamboogle** (Press et al., 2023), and **Musique** (Trivedi et al., 2022). For HotpotQA, 2WikiMultiHopQA, and Musique, we adopt the test splits released by R1-Searcher, each containing 500 examples. For Bamboogle, we use the full set of 125 test examples. For training, we use the dataset provided by R1-Searcher,

Method	Hotpotqa			2wikimultihopqa			Musique			Bamboogle		
	EM	F1	Judge	EM	F1	Judge	EM	F1	Judge	EM	F1	Judge
Direct Generation	18.0	23.5	24.2	19.0	23.6	23.0	3.6	9.0	6.0	20.8	30.7	28.8
COT	18.2	24.4	25.2	21.0	25.7	25.0	4.4	10.9	8.8	21.6	31.3	28.8
Naive RAG	29.4	40.7	43.4	26.0	30.5	29.8	5.2	11.0	8.0	18.4	27.0	22.4
Sure	20.2	27.2	32.0	21.0	24.6	26.2	3.0	8.5	5.2	10.4	17.9	12.8
IRCOT	22.2	32.7	38.6	17.4	23.4	26.0	5.2	10.8	8.4	13.6	23.8	24.0
Self-ask	14.4	24.6	39.0	14.8	19.6	25.0	4.0	10.3	7.4	9.6	22.3	18.4
RAG with Agentic Search	7.0	10.1	10.4	10.4	12.3	12.4	1.4	6.8	3.4	9.6	13.3	12.0
Search-o1	12.4	17.6	17.2	17.0	19.6	18.8	3.4	8.8	6.4	14.4	23.0	18.4
SFT	15.8	20.1	18.2	28.4	31.1	30.0	2.2	9.6	3.8	8.0	15.8	8.8
SimpleDeepSearcher	34.4	45.3	47.6	39.6	46.5	47.8	12.4	20.5	18.8	33.6	44.1	42.4
ReSearch-Base	28.8	38.4	36.2	37.2	40.7	40.0	14.4	24.2	17.4	34.8	45.5	37.2
ReSearch-Instruct	30.8	41.7	43.6	38.0	42.0	41.6	14.2	23.8	18.0	34.8	47.1	42.0
R1-search-Base	30.8	40.7	48.0	37.2	41.1	40.2	13.6	23.9	17.0	33.2	41.3	37.2
R1-search-Instruct	31.2	42.2	43.4	42.6	47.1	46.2	15.4	25.3	18.6	33.2	43.5	39.6
Search-R1-Base	35.4	50.1	51.2	44.0	50.9	51.4	14.8	26.0	22.6	38.4	50.9	48.4
Search-R1-Instruct	37.4	49.3	50.4	47.6	51.7	53.4	16.2	23.7	21.0	40.2	50.3	47.4
LeTS-Instruct*	37.1	-	55.2	41.0	-	47.5	17.5	-	26.9	38.4	-	51.2
TIRESRAG-R1-Base	**41.0**	53.0	**56.4**	52.4	58.4	60.2	16.2	26.9	24.4	40.4	52.6	50.0
TIRESRAG-R1-Instruct	41.0	**54.2**	56.0	**52.8**	**59.6**	**61.4**	**19.4**	**30.0**	**27.4**	**44.0**	**54.7**	**52.8**

Table 3: Main experimental results on HotpotQA, 2WikiMultiHopQA, Musique, and Bamboogle.

which includes 4,561 examples from the HotpotQA training set and 3,587 examples from the 2WikiMultiHopQA training set.

Evaluation Metrics. We evaluate model performance using four metrics: **Exact Match (EM), F1, LLM-as-Judge, and Cover Exact Match (CEM)**. EM checks exact matches, F1 accounts for partial overlaps, LLM-as-Judge (via GPT-4o) assesses semantic correctness, and CEM measures whether the gold answer is covered. For more detailed settings, please refer to App. C.2.

Baselines. We compare **TIRESRAG-R1** with 14 representative baselines spanning four categories: (1) **Naive prompt methods:** Direct, COT, and R1-based; (2) **Retrieval-augmented prompt methods:** Naive RAG, Agentic-R1, Search-o1, SURE, IRCOT, Self-Ask, and RQRAG; (3) **SFT methods:** SFT and SimpleDeepSearcher; (4) **RL methods:** Search-R1, R1-Searcher, Research, and the process-reward method LeTS. For detailed descriptions and configurations of these baselines, please refer to the App. C.3.

5.1 Main Results

Table 3 shows that **TIRESRAG-R1** achieves superior or competitive performance on all four complex multi-hop datasets when trained on either Qwen-2.5-3B-Base or Qwen-2.5-3B-Instruct. Our key findings are as follows:

(1) For the small 3B models, prompt-based methods generally perform poorly. For instance, *Search-o1* even underperforms *naive RAG*, as 3B models struggle to interpret instructions and generate effective retrieval queries without fine-tuning.

(2) Incorporating format rewards may not be optimal. Compared to Search-R1 (answer reward only), Research and R1-Searcher exhibit average EM drops of 5.13% and 4.60%, respectively. We attribute this decline to format-based learning, which reduces the exploration capacity of the 3B models and weakens useful reward signals when answers are correct but deviate slightly in format.

(3) TIRESRAG-R1 delivers substantial performance gains. On in-domain datasets HOTPOTQA and 2WIKIMULTIHOPQA, it outperforms SEARCH-R1 by average EM margins of 4.7% and 7.0%, respectively. For out-of-domain datasets MUSIQUE and BAMBOOGLE, we observe improvements of 5.3% and 4.6%. Compared to LeTS, TIRESRAG-R1 achieves additional gains of 5.8 and 4.2 points in EM and LLM-AS-JUDGE, respectively. These results indicate that our approach effectively learns higher-quality reasoning chains and generalizes well to unseen domains.

5.2 Analysis

We perform a comprehensive analysis to better understand the factors influencing our method's effectiveness. Below, we highlight the most significant findings; for a more detailed discussion and a case study, please refer to Appendix G.

Different RL Methods. To assess the generality of our training strategy, we also apply it to **Reinforce++** (Hu et al., 2025a) that does not use a

Method	Hotpotqa			2wikimultihopqa			Musique			Bamboogle		
	EM	F1	CEM	EM	F1	CEM	EM	F1	CEM	EM	F1	CEM
Ours-Base+GRPO	41.0	53.0	47.8	52.4	58.4	59.0	16.2	26.9	21.4	40.4	52.6	43.6
Ours-Instruct+GRPO	41.0	54.2	46.0	**52.8**	**59.6**	**60.8**	**19.4**	**30.0**	**23.2**	**44.0**	**54.8**	**47.2**
Ours-Base+Reinforce++	**42.2**	**55.9**	49.0	52.0	58.6	61.2	16.6	28.7	22.2	36.8	50.7	44.0
Ours-Instruct+Reinforce++	39.6	53.9	**49.8**	46.2	55.0	58.4	15.4	25.4	20.6	37.6	48.8	42.4

Table 4: Comparison between GRPO and Reinforce++ across datasets.

Datasets	**HotpotQA**	**2WikiMultiHopQA**	**MusiQue**	**Bamboogle**	**Average**
Thinking Length					
Naive grpo	318.7	293.1	467.7	215.2	323.7
Search-R1-Instruct	258.4	331.3	360.8	194.3	286.2
TIRESRAG-R1-Instruct	**252.1**	**312.1**	**303.6**	**229.2**	**274.3**
Search Steps					
Naive grpo	2.7	2.9	3.9	2.2	2.93
Search-R1-Instruct	2.3	2.3	2.8	2.2	2.40
TIRESRAG-R1-Instruct	**2.1**	**2.7**	**2.6**	**2.0**	**2.35**

Table 5: Comparison of average thinking length and search steps across datasets.

Method	**NQ**	**PopQA**	**TriviaQA**
COT	10.5	9.7	7.8
Sure	25.5	30.4	13.4
SimpleDeepSearcher	33.4	38.9	59.6
ReSearch-Instruct	35.8	41.8	58.4
TIRESRAG-R1-Instruct	**38.0**	**43.0**	**60.0**

Table 6: Generalization results on single-hop benchmarks.

critic model. The key difference between Reinforce++ and GRPO is that it normalizes advantages across the entire batch rather than within group rollouts. We train both Qwen2.5-3B-Base and Instruct models under identical settings. As shown in Table 4, compared with GRPO's group-normalization strategy, Reinforce++ performs better on in-domain datasets (HotpotQA, 2WikiMultiHopQA) but worse on out-of-domain datasets (Musique, Bamboogle), which is consistent with the findings in R1-Searcher.

Efficiency Analysis. We compare search counts and thought lengths with naive GRPO and Search-R1. As shown in Table 5, despite encouraging reflection and sufficient information gathering, our method does not hurt efficiency. Compared to naive GRPO, our method reduces average search count by 0.58 points and token generation by 49.4 points. Compared to Search-R1, search count decreases by 0.05 points and token generation by 12 points. This indicates that RL-trained Agent-RAG models can achieve improved reasoning quality without sacrificing efficiency.

5.3 Generalization on Single-Hop Benchmarks

Since our primary training data consist of multi-hop QA examples, we also evaluate on single-hop tasks to assess generalization. We use three widely adopted single-hop QA benchmarks: NQ, PopQA, and TriviaQA and present comparisons with the best baseline in each category. More comprehensive results can be found in the App. H. Table 6 shows that our method consistently outperforms all baselines across all metrics. On NQ, it achieves EM score of 38.0, exceeding the strongest baseline (ReSearch-Instruct) by 2.2 points. On PopQA, we obtain 43.0, surpassing baselines by 1.5 points. For TriviaQA, we achieve 60.0, with an improvement of 2.1 points. These results demonstrate that, despite being trained primarily on multi-hop data, our method generalizes effectively to single-hop tasks, highlighting its robustness across reasoning types and task distribution.

6 Conclusion

In this work, we reveal the limitations of current outcome-supervised RL-trained RAG reasoning models. To address these issues, we propose **TIRESRAG-R1**, which uses a novel *think–search–reflect* paradigm, explicitly encouraging models to reflect on uncertain answers. We design multiple reward functions to improve document retrieval sufficiency and reasoning quality, and introduce a difficulty-aware advantage strategy to provide stronger learning signals on hard problems. Comprehensive evaluations show that TIRESRAG-R1 outperforms existing methods on four multi-hop QA benchmarks. Further analysis highlights that our method is compatible with different RL algorithms and exhibits strong generalization.

Limitations

Model scaling. Our experiments are conducted only on Qwen2.5-3B due to computational constraints. Although TIRESRAG-R1 shows promising results on smaller models, its effectiveness on larger architectures (e.g., Qwen2.5-7B) remains unexplored.

Reward modeling. We use Qwen3-8B for sufficient and thinking scoring. Although it provides accurate signals, using a stronger model such as GPT-4o, or a specialized reward model fine-tuned on domain-specific data, may further improve performance. Exploring more accurate reward models (e.g., as suggested in recent thinking-reward literature) is a promising direction.

Reflection signal sparsity. While our reflection mechanism corrects some wrong answers, the number of training examples requiring reflection is limited, reducing useful learning signals. Future work could synthesize reflection-rich data for SFT before applying TIRESRAG-R1 to further improve the model's ability to reflect effectively.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian's, Malta. Association for Computational Linguistics.

Orlando Ayala and Patrice Bechard. 2024. Reducing hallucination in structured outputs via retrieval-augmented generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 228–238, Mexico City, Mexico. Association for Computational Linguistics.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. RQ-RAG: Learning to refine queries for retrieval augmented generation. In *First Conference on Language Modeling*.

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. *Preprint*, arXiv:2503.19470.

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, and . Aixin Liu etc. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Anoushka Gade and Jorjeta Jetcheva. 2024. It's about time: Incorporating temporality in retrieval augmented language models. *Preprint*, arXiv:2401.13222.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024a. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint*. ArXiv:2312.10997.

Yunfan Gao, Yun Xiong, Meng Wang, and Haofen Wang. 2024b. Modular rag: Transforming rag systems into lego-like reconfigurable frameworks. *Preprint*, arXiv:2407.21059.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, and . Binh Tang etc. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

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

Jian Hu, Jason Klein Liu, and Wei Shen. 2025a. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models. *Preprint*, arXiv:2501.03262.

Jian Hu, Xibin Wu, Wei Shen, Jason Klein Liu, Zilin Zhu, Weixun Wang, Songlin Jiang, Haoran Wang, Hao Chen, Bin Chen, Weikai Fang, Xianyu, Yu Cao, Haotian Xu, and Yiming Liu. 2025b. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *Preprint*, arXiv:2405.11143.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025c. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *Preprint*, arXiv:2503.24290.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2025. A survey on large language models for code generation. *ACM Trans. Softw. Eng. Methodol.* Just Accepted.

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

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao 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 '25, page 737–740, New York, NY, USA. Association for Computing Machinery.

Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ankur Taly, and Cyrus Rashtchian. 2025. Sufficient context: A new lens on retrieval augmented generation systems. In *The Thirteenth International Conference on Learning Representations*.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, Toronto, Canada. Association for Computational Linguistics.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain QA of LLMs. In *The Twelfth International Conference on Learning Representations*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. 2024. PlanRAG: A plan-then-retrieval augmented generation for generative large language models as decision makers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6537–6555, Mexico City, Mexico. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Jiarui Li, Ye Yuan, and Zehua Zhang. 2024a. Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases. *Preprint*, arXiv:2403.10446.

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. *Preprint*, arXiv:2501.05366.

Xingxuan Li, Weiwen Xu, Ruochen Zhao, Fangkai Jiao, Shafiq Joty, and Lidong Bing. 2024b. Can we further elicit reasoning in llms? critic-guided planning with retrieval-augmentation for solving challenging tasks. *Preprint*, arXiv:2410.01428.

Yuan Li, Qi Luo, Xiaonan Li, Bufan Li, Qinyuan Cheng, Bo Wang, Yining Zheng, Yuxin Wang, Zhangyue Yin, and Xipeng Qiu. 2025b. R3-rag: Learning step-by-step reasoning and retrieval for llms via reinforcement learning. *Preprint*, arXiv:2505.23794.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

Chunyang Meng, Shijie Song, Haogang Tong, Maolin Pan, and Yang Yu. 2024. Deepscaler: Holistic autoscaling for microservices based on spatiotemporal gnn with adaptive graph learning. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*, ASE '23, page 53–65. IEEE Press.

Seyed Mahed Mousavi, Simone Alghisi, and Giuseppe Riccardi. 2024. Dyknow: Dynamically verifying time-sensitive factual knowledge in llms. *Preprint*, arXiv:2404.08700.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, and . AJ Ostrow etc. 2024a. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, and . Alex Karpenko etc. 2024b. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

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

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. *Preprint*, arXiv:2505.13258.

Zeyang Sha, Shiwen Cui, and Weiqiang Wang. 2025. Sem: Reinforcement learning for search-efficient large language models. *Preprint*, arXiv:2505.07903.

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*, pages 9248–9274, Singapore. Association for Computational Linguistics.

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

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. *Preprint*, arXiv:2505.11277.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

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. *Preprint*, arXiv:2503.05592.

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. *Preprint*, arXiv:2505.04588.

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. *Preprint*, arXiv:2505.16834.

Yixuan Tang and Yi Yang. 2024. Multihop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In *First Conference on Language Modeling*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Yanlin Wang, Yanli Wang, Daya Guo, Jiachi Chen, Ruikai Zhang, Yuchi Ma, and Zibin Zheng. 2024a. Rlcoder: Reinforcement learning for repository-level code completion. *Preprint*, arXiv:2407.19487.

Yike Wang, Shangbin Feng, Heng Wang, Weijia Shi, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2024b. Resolving knowledge conflicts in large language models. In *First Conference on Language Modeling*.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. 2025.

Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *Preprint*, arXiv:2505.15107.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. 2025. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *Preprint*, arXiv:2503.10460.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024. Knowledge conflicts for LLMs: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8541–8565, Miami, Florida, USA. Association for Computational Linguistics.

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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *Preprint*, arXiv:2503.18892.

Jixiao Zhang and Chunsheng Zuo. 2025. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. *Preprint*, arXiv:2504.09696.

Qi Zhang, Shouqing Yang, Lirong Gao, Hao Chen, Xiaomeng Hu, Jinglei Chen, Jiexiang Wang, Sheng Guo, Bo Zheng, Haobo Wang, and Junbo Zhao. 2025a. Lets: Learning to think-and-search via process-and-outcome reward hybridization. *Preprint*, arXiv:2505.17447.

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. 2025b. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *Preprint*, arXiv:2505.14069.

Xiaoming Zhang, Ming Wang, Xiaocui Yang, Daling Wang, Shi Feng, and Yifei Zhang. 2024. Hierarchical retrieval-augmented generation model with rethink for multi-hop question answering. *Preprint*, arXiv:2408.11875.

Qingfei Zhao, Ruobing Wang, Dingling Xu, Daren Zha, and Limin Liu. 2025. R-search: Empowering llm reasoning with search via multi-reward reinforcement learning. *Preprint*, arXiv:2506.04185.

Yujia Zhou, Zheng Liu, Jiajie Jin, Jian-Yun Nie, and Zhicheng Dou. 2024. Metacognitive retrieval-augmented large language models. In *The Web Conference 2024*.

A Related Work

Retrieval-Augmented Generation. RAG has emerged as a prominent framework to augment LLMs with external knowledge, aiming to mitigate issues such as hallucination (Ayala and Bechard, 2024), domain incompleteness (Siriwardhana et al., 2023), and temporal staleness (Gade and Jetcheva, 2024). Conventional RAG systems adopt a static retrieve-then-generate paradigm, where a retriever first fetches top-ranked documents given an input query, and a generator conditions its output on the retrieved context (Lewis et al., 2020; Guu et al., 2020). While this structure has proven effective for factual QA and open-domain generation, it often falls short when confronted with complex reasoning tasks involving multi-hop dependencies (Tang and Yang, 2024), latent constraints (Li et al., 2024b), or ambiguous query intents (Chan et al., 2024). To overcome these limitations, recent works have proposed more cognitively informed RAG architectures. For example, AdaptiveRAG (Jeong et al., 2024) uses query classification to trigger different retrieval strategies, PlanRAG (Lee et al., 2024) decomposes tasks into executable plans for targeted retrieval, and ITER-RETGEN (Shao et al., 2023) incorporates intermediate generation to iteratively reformulate queries. In parallel, modular and hybrid RAG frameworks (Gao et al., 2024b; Zhang et al., 2024; Zhou et al., 2024) have introduced componentized systems that integrate query rewriting, evidence aggregation, and verification in sequential or recursive pipelines. These advances suggest that effective RAG increasingly requires not just better retrieval quality, but dynamic, context-aware reasoning to inform when, what, and how to retrieve. This growing interdependence between retrieval and reasoning sets the stage for more adaptive mechanisms (particularly those that go beyond predefined rules) highlighting the need for learning-based control in complex RAG workflows.

Reinforcement Learning for LLM Reasoning. Driven by the growing need for LLMs to perform complex and reliable reasoning across diverse tasks, recent research has turned to reinforcement learning as a promising paradigm to enhance their reasoning capabilities. The release of GPT-o1 (OpenAI et al., 2024b) and DeepSeek-R1 (DeepSeek-AI et al., 2025) marked a shift toward training LLMs that exhibit structured, multi-step reasoning through RL-based objectives. Early efforts such as SimpleRL-Zoo (Zeng et al., 2025) and Open-Reasoner-Zero (Hu et al., 2025c) explored direct RL fine-tuning from base models, eliminating the reliance on extensive supervised instruction tuning. Building on this foundation, approaches like DeepScaler (Meng et al., 2024) and Light-R1 (Wen et al., 2025) introduced cold-start datasets and reward schemes explicitly designed to promote step-by-step thinking and verifiable inferences. In parallel, improvements to RL algorithms, such as Dr GRPO (Liu et al., 2025), refined policy optimization to better align with the cognitive demands of long-form reasoning.

Reinforcement Learning for RAG Reasoning. There is a growing body of work focused on bringing RL-based reasoning into the retrieval-augmented generation framework (Song et al., 2025; Jin et al., 2025a; Chen et al., 2025). Inspired by DeepSeek-R1, these approaches use regularized rewards to encourage the model to think and generate retrieval queries to search external corpora. However, these methods only consider the final outcome reward signal to train the model, which is relatively simple, and do not deeply optimize according to the characteristics of the RAG task itself. In view of this, many works incorporate the model's thinking process into the reward calculation (Wang et al., 2025; Sha et al., 2025; Zhang et al., 2025b). For example, Zhao et al. (2025) uses different models to generate answers based on the same evidence in order to calculate evidence quality, while also alleviating answer bias caused by the preference of the policy model itself. R3-RAG (Li et al., 2025b) calculates the relevance between each retrieved document and the question at every step, attempting to improve the model's search strategy through fine-grained process rewards. Different from these works, our work directly measures the sufficiency between all retrieved documents and the question, in order to enhance the model's awareness of searching less or more. In addition, some works attempt to add new information fields to the "search–thinking–answer" pipeline proposed by search-r1-type methods to prompt the model to think more about the documents (Ren et al., 2025). For example, Shi et al. (2025) lets the model refine the retrieved documents during the reasoning process before proceeding to thinking. Our work, unlike in these methods, directly rewards the model's thinking process, allowing the model to optimize its reasoning process, rather than manually adding thinking rules.

B Experimental Setup for Preliminary Studies

We trained a reasoning-capable RAG model using the GRPO algorithm from Deepseek-R1. Following R1-Searcher, we selected part of the training data from HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020), with a total of 8,148 examples. We used the Qwen-2.5-3B-Instruct model for training. For the retriever, we deployed a system locally based on the BGE-large-en-v1.5 retrieval, with the retrieval corpus from English Wikipedia provided by KILT (Petroni et al., 2021) in 2019. The training prompt can be found in Appendix J. We trained the model for one epoch (detailed experimental settings are the same as those in Section 5.) and evaluated the model on the in-domain 2Wiki test set and the out-of-domain Musique test set. After obtaining predictions on 500 test examples from each dataset, we used GPT-4o to evaluate the correlation between the model's predictions and the retrieved content plus the reasoning process.

We introduce notation for a generic open-domain question–reasoning setting, assuming sufficient contextual information. Consider an example represented as $E = (Q, RD, A)$, where Q is the query, RD is the combined reasoning process and retrieved documents, and A is either the model's predicted answer or the gold answer. We define $RD = \{R_1, D_1, \ldots, R_i, D_i, R_{i+1}\}_{i=1}^{n}$, where n is the number of sub-questions generated by the model during reasoning, R_i denotes the intermediate reasoning step prior to document retrieval, and D_i represents the set of documents retrieved in response to the sub-question formulated in R_i.

We define three cases: (1) **Overthinking**: the model produces too many reasoning steps, meaning that the gold answer could already be inferred at some step R_i with $i < n$. (2) **Good thinking**: the model obtains sufficient content exactly before the final reasoning step, i.e., the gold answer can only be inferred at step R_{n+1}. (3) **Underthinking**: the model stops reasoning without obtaining sufficient content in RD, and still outputs an answer. Since Joren et al. (2025) has shown enabling LLMs to judge whether the provided context is adequate for answering a question is effective, we directly input Q, RD, and the gold answer into an LLM to assess whether Q and RD together are sufficient to derive the correct answer (see prompt in App.J). Importantly, we include the gold answer in the input, unlike what (Joren et al., 2025) does. which uses only Q and RD. This is because we found that omitting the answer leads to more errors: when RD is actually insufficient, the model is more likely to incorrectly judge it as sufficient.

C Implementation Details and Baselines

C.1 Implementation Details.

For all baselines and **TIRESRAG-R1**, we use Qwen-2.5-3B (base and instruct variants) as the backbone. Following R1-Searcher, we use the 2019 English Wikipedia as the external knowledge source, and BGE-large-en-v1.5 as the retrieval engine. Our training framework is built upon OpenRLHF (Hu et al., 2025b), and we use FlashRAG (Jin et al., 2025b) for evaluation. The total batch size is 108, with a learning rate of 2×10^{-6}. We generate 5 rollout samples per input for reward estimation and set temperature = 1 during rollout to encourage exploration. The KL coefficient β is set to 0, the number of iterations per batch μ = 2, and the clipping parameter $\epsilon = 0.2$. Top-k during retrieval is set to 5. In the difficulty-aware resampling strategy, the hyperparameters are set to $A = 0.4$, $B = 1.5$, $\rho_0 = 0.75$, and $k = 10.0$. For the consistency penalty, we set $\lambda_p = 0.1$. In the overall reward computation, the weights are configured as $w_t = 0.6$, $w_s = 0.3$, and $w_r = 0.3$. All models are trained on 3 NVIDIA H200 GPUs: 2 GPUs are allocated for policy optimization, and 1 GPU is dedicated to rollout inference via vLLM (Kwon et al., 2023).

C.2 Evaluation Metrics

For open-ended evaluation, we report three widely used metrics: **Exact Match (EM)**, **F1 score** (aligned with the RL training answer reward), and **LLM-as-Judge**. EM measures whether the ground-truth answer exactly matches the model prediction. F1 is computed between predicted and gold answers to handle partial overlap. For LLM-as-Judge, we use GPT-4o to evaluate the semantic correctness of the prediction given the question and supporting evidence. The prompts used for LLM-as-Judge are listed in App. J. Due to the high cost of using LLM-as-a-Judge, we only adopt this metric in the main experiments. For the other experiments, we use **Cover Exact Match (CEM)**, which assesses whether the ground-truth answer is included in the predicted answer.

C.3 Baselines.

To evaluate the effectiveness of our proposed **TIRESRAG-R1** method, we implement and compare against 14 baselines, grouped into four categories:

(1) Naive prompt methods: **Direct** answers questions directly using its own parametric knowledge. **COT** is instructed to produce a chain of thought before the final answer. **R1-based** uses a distilled DeepSeek-Qwen-3B reasoning model to first reason then answer. We implement all three naive prompt methods ourselves.

(2) Retrieval-augmented prompt methods: **Naive RAG** extends **Direct** by retrieving documents for the query and appending them as additional input before direct answering. **Agentic-R1** (Li et al., 2025a) enables the model to autonomously retrieve external knowledge when needed while avoiding interference from irrelevant content. **Search-o1** (Li et al., 2025a) introduces a Reason-in-Documents module that condenses retrieved content into coherent reasoning steps, iteratively guiding the model to the final answer. **SURE** (Kim et al., 2024) generates and evaluates summaries of retrieved passages for multiple answer candidates. **IRCOT** (Trivedi et al., 2023) interleaves retrieval with Chain-of-Thought reasoning. **Self-Ask** (Press et al., 2023) improves multi-hop reasoning by decomposing the original question into intermediate sub-questions that are answered before final prediction. **RQRAG** (Chan et al., 2024) learns to explicitly refine queries through rewriting, decomposition, and disambiguation. For these retrieval-augmented methods, we use the implementations provided in `flashrag`.

(3) SFT methods: **SFT** fine-tunes the model directly on training pairs of questions and gold answers. During training, we only input the question and instruct the model to output a short answer without any intermediate reasoning or evidence selection. **SimpleDeepSearcher** (Sun et al., 2025b) constructs a high-quality dataset containing intermediate reasoning and retrieval steps, then fine-tunes the model with question-to-(reasoning,answer) pairs. Its input is also only the question, but its output format follows R1-Searcher with four components:

- <thinking> ... </thinking>,
- <|begin_search_query|> ... </|end_search_query|>,
- <|begin_search_result|> ... </|end_search_result|>,
- \box{answer}.

A total of 871 examples are constructed to reproduce this method, and during inference we extract the text inside the \box{} as the model's predicted answer.

(4) RL methods: **Search-R1** (Jin et al., 2025a) uses only F1 as reward. **R1-Searcher** (Song et al., 2025) uses answer reward plus a format reward. **Research** (Chen et al., 2025) also incorporates format reward. To ensure fairness, we re-trained these RL-based baselines on our training set using the authors' released code and hyperparameter settings, without using their checkpoints. **LeTS** (Zhang et al., 2025a) combines step-level rewards with answer rewards and introduces rollout-level redundancy penalties and group-level knowledge-matching rewards. Since LeTS has no public code, we report the results from the original paper.

All baselines are evaluated under a unified protocol: each method first produces its predicted answers, which are then scored using the same standardized evaluation script.

D Detailed Reinforce++

Reinforce++ (Hu et al., 2025a) is an efficient RLHF algorithm without a critic network, designed to address overfitting in advantage estimation and reward hacking in REINFORCE-based methods. Its core idea is to use the global batch mean reward as the baseline, rather than constructing a separate baseline for each prompt as in RLOO or GRPO. This avoids prompt-specific bias and improves stability and generalization. We adopt the same reward computation strategy as in our main experiments to ensure consistency across training and evaluation.

E Visualization of Annealing

To better illustrate the dynamic weight schedule defined in the main content, Figure 4 plots the four annealing strategies used in Section 7.2. Our strategy keeps the weight stable in the early and mid training phases, and then drops sharply in the late phase, aligning with our goal of letting the model focus on answer accuracy in the later stages.

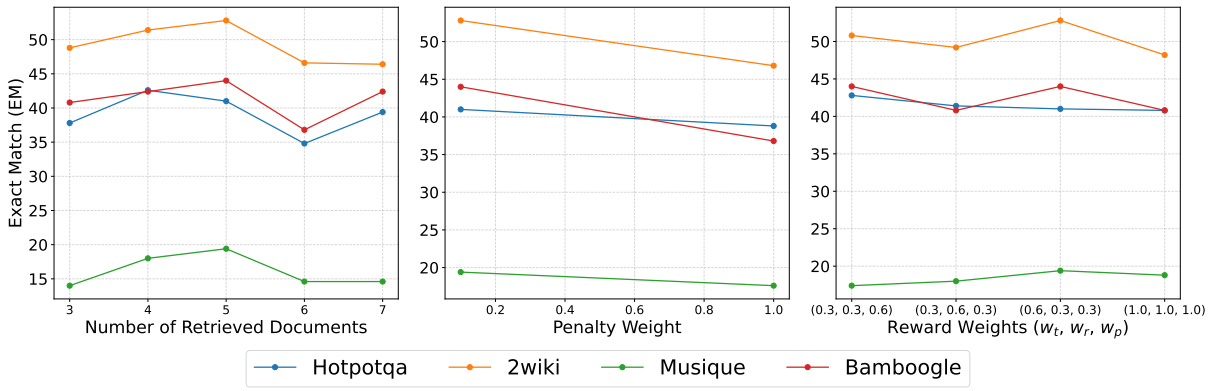

Figure 3: Effect of different hyperparameters.

Method	Hotpotqa			2wikimultihopqa			Musique			Bamboogle		
	EM	F1	CEM	EM	F1	CEM	EM	F1	ACC	CEM	F1	CEM
Naive GRPO	36.4	48.5	41.6	46.4	53.1	54.0	16.2	26.0	19.6	36.0	49.2	40.8
TIRESRAG-R1-Instruct	**41.0**	**54.2**	**46.0**	**52.8**	**59.6**	**60.8**	**19.4**	**30.0**	**23.2**	**44.0**	**54.7**	**47.2**
w/o Filter	18.8	24.6	26.8	21.4	26.5	28.8	6.0	11.4	9.4	19.2	28.8	25.6
w/o Difficulty	38.2	50.4	43.6	49.2	54.0	55.4	17.0	27.0	21.4	35.2	49.3	38.4
w/o Penalty	38.0	49.9	44.2	44.2	50.9	52.0	15.6	24.9	19.0	39.2	50.6	43.2
w/o Reflect	37.8	47.9	41.2	44.6	52.0	53.8	10.8	21.4	15.2	32.8	45.3	37.6
w/o Sufficient	37.4	48.5	44.6	41.4	46.3	46.4	14.0	22.3	16.4	32.6	43.4	35.8
w/o Thinking	39.8	51.8	47.6	44.8	51.3	53.8	14.8	23.9	19.2	37.6	46.4	39.2

Table 7: Ablation Study.

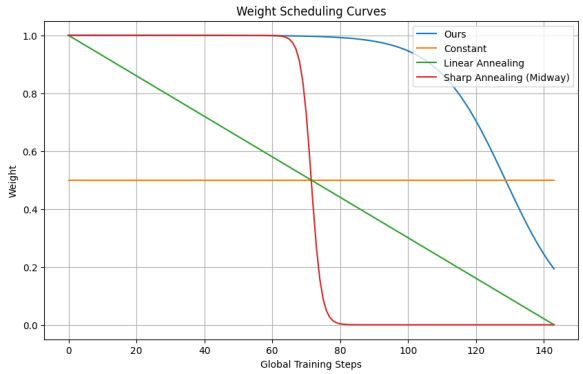

Figure 4: Visualization of different reward weight scheduling strategies curves.

F Ablation Study

To verify the effectiveness of each module in **TIRESRAG-R1**, we conduct systematic ablation experiments on the Qwen2.5-3B-Instruct model. Table 7 shows the obtained results. We observe that: **(1)** The **filter module** is crucial for model stability. As shown in Table 7, removing it significantly degrades performance: compared with standard GRPO, the average F1 score drops by 21.2%. This is because, in the later training stages, the model collapses. We show the training curves in Section 7.3. **(2)** The **difficulty-aware weighting** and **penalty** mechanisms are crucial for effectively integrating our diverse reward signals. While ablations without them still outperform naive GRPO, the gains are limited, showing average performance drops of 4.48 and 5.58 points compared to our full method. **(3)** Each reward component plays a critical role in overall performance. Removing any single reward leads to noticeable degradation—and in all cases, performance drops below that of naive GRPO. Notably, removing the **sufficient reward** results in the largest decline, indicating that the model may engage in reward hacking and neglect crucial external documents. In contrast, removing the **thinking reward** has the smallest impact, with an average drop of 6.3% compared to our full method. This is likely because the **answer**, **sufficient**, and **reflect** rewards already provide partial supervision for generating high-quality reasoning chain.

G Analysis

G.1 Impact of Hyperparameters.

We explore three key hyperparameters: the number of retrieved documents, the penalty weight, and the reward mixture weights. Figure 3 present the obtained results.

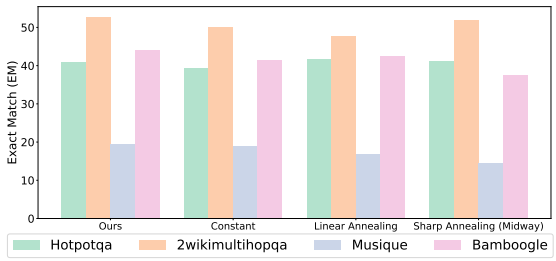

Figure 5: Comparison of different reward weight scheduling strategies.

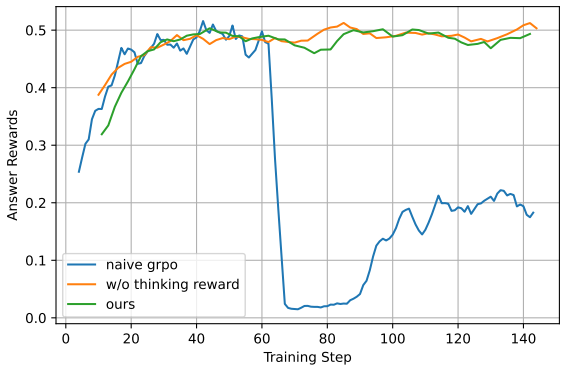

Figure 6: Training dynamics of answer rewards over steps.

Figure 7: Training dynamics of thinking rewards over steps.

- *Impact of number of retrieved documents.* As the number of retrieved documents increases from 3 to 5 (ours), the model performance improves, with EM increasing by 3.95 points on average. When increasing from 5 (ours) to 7, the average EM drops by 3.6 points, suggesting that retrieving too many documents introduces noise, hurting reasoning quality. There is thus a trade-off between providing more information and avoiding noise.

- *Impact of penalty weight.* In Eq. 7, λ_p controls the magnitude of the consistency penalty in the advantage calculation. Setting $\lambda_p = 0.5$ or $\lambda_p = 1$, we observe that as λ_p increases, performance decreases, with the largest drop when $\lambda_p = 1$. This suggests over-emphasizing consistency can suppress beneficial reward signals and hurt final performance.

- *Impact of reward mixture weights.* In Eq. 5, thinking, sufficient, and reflect rewards are combined with different weights. We try settings of (0.3,0.3,0.6), (0.3,0.6,0.3), (0.3,0.3,0.3), and our (0.6,0.3,0.3). Results show that giving the highest weight to thinking reward (ours) yields the best performance, likely because thinking directly measures the quality of the reasoning chain, which more strongly impacts final answers.

G.2 Impact of Annealing Strategy.

As described in Section 4.3, we adopt a decaying schedule for mixed reward weights over training steps. To analyze its effectiveness, we compare several variants (fixed weights, linear decay, fast decay). Weight curves are shown in Fig. 4. Results in Fig. 5 show that linear annealing performs worst, with an average EM drop of 2.35 points across datasets, suggesting auxiliary signals decay too early. In contrast, our proposed schedule achieves the best or second-best results on all datasets, especially on 2WikiMultiHopQA and Bamboogle. Fixed scheduling is relatively stable but suboptimal. Pre-low annealing performs slightly better on HotpotQA but worse on others.

G.3 Learning Curve Analysis.

Figures 6 and 7 show training dynamics. For naive GRPO, because it does not filter out "all-correct" or "all-wrong" queries, its answer reward is similar to ours early on but then drops sharply, showing clear collapse, before slowly recovering but remaining below initial levels. Our method maintains stability throughout training, with answer reward staying high. For the baseline without thinking reward, answer reward remains similar to ours, but thinking reward fluctuates and declines, indicating the model fails to learn a stable reasoning process. This further shows that process-level rewards help both reasoning quality and training stability.

G.4 Influence of Reward Backbone Model.

The reward model plays a critical role. Fig. 8 compares using Qwen2.5-3B and Qwen3-3B for suffi-

Method	Nq			Popqa			Triviaqa		
	EM	F1	CEM	EM	F1	CEM	EM	F1	CEM
Direct Generation	6.8	10.8	9.5	8.6	11.8	9.4	7.5	19.0	9.5
COT	10.5	17.5	15.0	9.7	13.8	10.7	7.8	20.6	10.1
Naive RAG	23.4	32.9	31.9	29.4	37.0	34.5	12.9	29.9	16.5
Sure	25.5	34.2	27.9	30.4	35.7	31.1	13.4	29.7	16.0
IRCOT	15.4	25.1	33.2	25.1	31.5	35.8	10.1	25.0	17.6
Self-ask	17.2	27.1	41.4	24.8	31.7	41.5	9.5	24.1	20.4
SFT	6.3	12.9	10.1	7.6	11.8	8.4	5.2	14.9	7.0
SimpleDeepSearcher	33.4	44.0	44.1	38.9	44.3	44.1	59.6	67.2	66.5
ReSearch-Instruct	35.8	46.2	44.5	41.8	47.4	46.3	58.4	66.1	64.1
Search-R1-Instruct	34.2	44.0	44.2	37.9	43.5	44.1	54.5	62.2	62.3
R1-search-Instruct	35.2	46.4	44.8	40.3	46.1	45.6	57.3	65.6	64.0
TIRESRAG-R1-Instruct	**38.0**	**49.1**	**47.9**	**43.0**	**48.8**	**47.9**	**60.0**	**68.2**	**66.9**

Table 8: Generalization results on single-hop benchmarks (NQ, PopQA, TriviaQA).

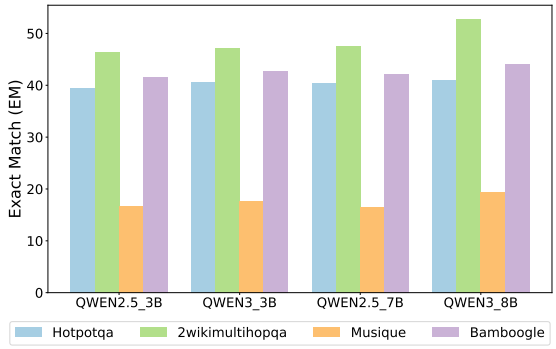

Figure 8: Comparison of different reward backbone models.

cient and thinking scoring. We observe consistent improvements with Qwen3-3B: on Musique, EM improves from 16.6 to 17.6; on Bamboogle, EM improves from 41.6 to 42.8. A similar trend is seen comparing Qwen2.5-3B and Qwen2.5-7B: on HotpotQA, EM improves from 39.4 to 40.4 and F1 from 52.7 to 54.4. These results show that stronger reward models provide better supervision and thus improve final performance.

H Generalization on Single-Hop Benchmarks

We present in Table 8 the complete experimental results with more baselines and additional metrics. As shown, our method outperforms all baselines across all metrics.

I Case Study

Figures 14, 15, and 16 present three examples illustrating the effectiveness of TIRESRAG-R1. In the first example, the TIRESRAG-R1-trained Qwen2.5-3B follows a correct reasoning process and gives the correct answer, while the naive GRPO model, despite having sufficient information, produces an incorrect reasoning chain and thus a wrong answer. In the second example, the TIRESRAG-R1-trained model successfully retrieves enough information before answering, whereas the naive GRPO model guesses prematurely without sufficient evidence and produces a wrong answer. In the third example, the naive GRPO model has sufficient information and correct reasoning but still outputs the wrong answer, while our model successfully produces the correct answer. These examples show that TIRESRAG-R1 guides models toward better reasoning chains and ensures stronger consistency between reasoning and answers.

J Prompt Templates

Figures 9 to 13 present all prompt templates mentioned in this paper.

- **Figure 9:** Prompt used for training instruction-based models, setting the system's reasoning–retrieval–reflection strategy and output format.
- **Figure 10:** Prompt used for training base models, converting the above instruction-based prompt into a conversation format.
- **Figure 11:** Prompt for scoring the sufficiency of reasoning trajectories. We follow the sufficient-context method: the model is prompted to list sub-questions that solve the main question, and only if all sub-questions can be answered from the given references is the trajectory judged sufficient. A demonstration example is provided.

- **Figure 12:** Prompt for scoring reasoning quality. The four criteria described in the main text (logical soundness, alignment, error awareness, conciseness) are included. The model is prompted to output only a number between 0 and 1 to avoid format errors. In experiments, the reward model followed the prompt strictly and output valid scores.

- **Figure 13:** Prompt used for LLM-as-Judge evaluation. We embed both the predicted answer and the gold answer into the prompt and feed it to GPT-4o. If GPT-4o judges that the predicted and gold answers are semantically equivalent, it returns "yes," otherwise "no."

K Algorithm

The pseudo code for TIRESRAG-R1 is shown in Algorithm 1.

Prompt Template for RAG-reasoning Model Generation for Instruction-based Model

```
    You are a helpful assistant.  Answer the given question.
    You must reason **clearly and completely** inside <think> and </think>
before providing any final answer.  Always identify and verify all key
entities (e.g., person names, locations, dates, awards) mentioned in the
question.
    If you are uncertain about an entity or fact, or if the question
requires external knowledge, you may use <search>your query</search>,
and the top search results will be returned between <information> and
</information>.  Carefully read and reflect on each newly retrieved piece
of information.  You can search as many times as you want.
    When reasoning, you must ensure your reasoning path aligns strictly
with the evidence.
    After reasoning, before providing your final answer, rethink it to make
sure the answer is exactly correct for the original question.  Use the
most accurate span from the evidence when possible.
    Only after satisfying all the above, give the final answer inside
<answer> and </answer>.  For example, <answer>Beijing</answer>.
    After outputting the final answer in <answer></answer>, you have
one chance to reflect on the answer.  If you choose not to reflect,
nothing further needs to be done.  Otherwise, you can then re-examine
your thinking process, the information obtained, and even search for
more information to verify your previous answer or correct the previous
answer.  Remember, after reflection ends, you should output the answer in
<answer></answer>.
```

Figure 9: Prompt template for instruction model training.

Prompt Template for RAG-reasoning Model Generation for Base Model

```
    Answer the given question.
    You must reason **clearly and completely** inside <think> and </think>
before providing any final answer.
    Always identify and verify all key entities (e.g., person names,
locations, dates, awards) mentioned in the question.
    If you are uncertain about an entity or fact, or if the question
requires external knowledge, you may use <search>your query</search>,
and the top search results will be returned between <information> and
</information>.  Carefully read and reflect on each newly retrieved piece
of information.
    You can search as many times as you want.
    When reasoning, you must ensure your reasoning path aligns strictly
with the evidence.
    After reasoning, before providing your final answer, rethink it to make
sure the answer is exactly correct for the original question.
    Use the most accurate span from the evidence when possible.
    Only after satisfying all the above, give the final answer inside
<answer> </answer>.  For example, <answer> Beijing </answer>.
    After outputting the final answer in <answer> </answer>, you have one
chance to reflect on the answer.  If you choose not to reflect, nothing
further needs to be done.
    Otherwise, you can then re-examine your thinking process, the
information obtained, and even search for more information to verify
your previous answer or correct the previous answer.  Remember, after
reflection ends, you should output the answer in <answer> </answer>.
```

Figure 10: Prompt template for base model training.

Algorithm 1 TIRESRAG-R1 Training with GRPO and Multi-Dimensional Rewards

Require: Policy model π_θ, reference model $\pi_{\theta_{\text{old}}}$, dataset $\mathcal{D}$, retrieval model π_{ret}, hyperparameter weights $w_{\text{think}}, w_{\text{suff}}, w_{\text{reflect}}$, KL penalty β, iterations T, rollouts per query G, buffer B, dynamic weight d_w

Ensure: Updated policy model π_θ

for $t = 1, \ldots, T$ **do**

  Compute dynamic weight:

$$a_t \leftarrow \frac{1}{1 + \exp\left(\frac{T - 0.9t}{10}\right)}$$

  Sample $Q \subset \mathcal{D}$

  for $q \in Q$ **do**

    Generate rollouts $\{y_i\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid q)$

    for $i = 1, \ldots, G$ **do**

      Extract reasoning trajectory RD_i and prediction a_i from y_i

      Compute answer reward:

$$R_i^A \leftarrow \mathrm{F1}(a_i, a^*)$$

      where a^* is the gold answer for q

      Compute sufficient reward:

$$\mathrm{R_i^S}(q, RD_i, a^*) = \begin{cases} 1, & RD_i \text{ contains sufficient info to derive } a^*, \\ 0, & \text{otherwise} \end{cases} \quad i$$

      Compute think reward:

$$R_i^T \leftarrow \mathrm{Think}(RD_i), \quad \mathrm{Think}(RD_i) \in [0, 1]$$

      scored on logic, alignment, error-awareness, and conciseness

      Compute reflection reward:

$$\mathrm{R_i^T}(a_i) = \begin{cases} +1, & \text{if } CEM(a_1, a^*) = 0 \text{ and } CEM(a_2, a^*) = 1, \\ -1, & \text{if } CEM(a_1, a^*) = 1 \text{ and } CEM(a_2, a^*) = 0, \\ 0, & \text{otherwise} \end{cases}$$

      Combine rewards:

$$R_i^{sum} = a_t \cdot \left(w_{\text{think}} R_i^T + w_{\text{suff}} R_i^S + w_{\text{reflect}} R_i^R\right) + R_i^A$$

    end for

    if $0.1 < \forall\, r_i^{(a)} < 0.9$ for $i \in \{1, \ldots, G\}$ **then**

      Compute advantage by normalizing batch reward:

$$A_i \leftarrow \frac{R_i^{\text{sum}} - \frac{1}{G}\sum_{j=1}^{G} R_j^{\text{sum}}}{\sqrt{\frac{1}{G}\sum_{j=1}^{G}\left(R_j^{\text{sum}} - \frac{1}{G}\sum_{k=1}^{G} R_k^{\text{sum}}\right)^2}}$$

      Compute consistency penalty A_i^P by Eq. 7

      Apply difficulty-aware weighting:

$$A_i' \leftarrow (A_i - A_i^P) \cdot W(R_{\text{avg}}^S), \quad \text{where } W(R_{\text{avg}}^S) \text{ is calculated by Eq. 6}$$

      Add sample to buffer:

$$B \leftarrow B \cup \{(q, y_i, A_i')\}_{i=1}^{G}$$

    else

      continue

    end if

    Update π_θ on buffer B

  end for

end for

return π_θ

Prompt for Sufficient Reward Evaluation

```
You are an expert LLM evaluator that excels at evaluating a QUESTION, ANSWER
and REFERENCES. Consider the following criteria:
Sufficient Context To The Given Answer:  1 IF the CONTEXT is sufficient to
infer the ANSWER to the question and 0 IF the CONTEXT cannot be used to infer
the ANSWER to the question.  Make the sufficiency judgment based solely on the
context, without relying on your memory to determine whether the question can
be answered from the context.

First, output a list of step-by-step questions that would be used to
arrive at a label for the criteria.  Make sure to include questions about
assumptions implicit in the QUESTION. Include questions about any mathematical
calculations or arithmetic that would be required.

Next, answer each of the questions.  Please note that you may answer these
questions only on the basis of the given context; do not use your own outside
knowledge.  Make sure to work step by step through any required mathematical
calculations or arithmetic.  Finally, use these answers to evaluate the
criteria.

EVALUATION (JSON)
EXAMPLE:
### QUESTION
In which year did the publisher of Roald Dahl's Guide to Railway Safety cease
to exist?
### ANSWER
2001
### References
Roald Dahl's Guide to Railway Safety was published in 1991 by the British
Railways Board.  The British Railways Board had asked Roald Dahl to write
the text of the booklet, and Quentin Blake to illustrate it, to help young
people enjoy using the railways safely.  The British Railways Board (BRB)
was a nationalised industry in the United Kingdom that operated from 1963
to 2001.  Until 1997 it was responsible for most railway services in Great
Britain, trading under the brand name British Railways and, from 1965, British
Rail.  It did not operate railways in Northern Ireland, where railways were
the responsibility of the Government of Northern Ireland.
### EXPLANATION
The context mentions that Roald Dahl's Guide to Railway Safety was published
by the British Railways Board.  It also states that the British Railways Board
operated from 1963 to 2001, meaning the year it ceased to exist was 2001.
Therefore, the context does provide a precise answer to the question.
### JSON
{{"Sufficient Context To The Given Answer":  1}}

Remember the instructions:  You are an expert LLM evaluator that excels at
evaluating a QUESTION, ANSWER and REFERENCES. Consider the following criteria:
Sufficient Context:  1 IF the CONTEXT is sufficient to infer the ANSWER to
the question and 0 IF the CONTEXT cannot be used to infer the ANSWER to the
question.  Make the sufficiency judgment based solely on the context, without
relying on your memory to determine whether the question can be answered from
the context.
First, output a list of step-by-step questions that would be used to
arrive at a label for the criteria.  Make sure to include questions about
assumptions implicit in the QUESTION. Include questions about any mathematical
calculations or arithmetic that would be required.
Next, answer each of the questions.  Please note that you may answer these
questions only on the basis of the given context; do not use your own outside
knowledge.  Make sure to work step by step through any required mathematical
calculations or arithmetic.
Finally, use these answers to evaluate the criteria.  Output the ###
EXPLANATION (Text).  Then, use the EXPLANATION to output the ### EVALUATION
(JSON)
```

Figure 11: Prompt used to evaluate context sufficiency.

Prompt for Thinking Reward Evaluation

```
You are an expert reasoning evaluator for Retrieval-Augmented Generation
(RAG) tasks.
Your goal is to judge the reasoning quality of the model's thinking
process based on the retrieved context and question.
You will assign a reward score between 0 and 1.  This score reflects only
the quality of the reasoning process, not whether the final answer is
correct.

Evaluation Criteria:
1.  Logical Soundness - Is the reasoning coherent and structured?
2.  Contextual Alignment - Does it use retrieved evidence correctly?
3.  Error Awareness - Does it avoid unsupported assumptions?
4.  Clarity and Precision - Is it concise, relevant, and non-redundant?

Scoring:
0.0:  Completely flawed reasoning
1.0:  Perfect reasoning
Intermediate (e.g., 0.3, 0.7) are allowed.

Important:
- Judge only the thinking process, not the answer.
- Reward accurate, grounded, and structured reasoning.

Your Output:
A single float-type score from {{0.0, 0.1, 0.2, ..., 1.0}}.
No explanation.  Only the score.
```

Figure 12: Prompt used to evaluate thinking quality

Prompt for LLM-As-Judge

```
Given a Question and its Golden Answer, verify whether the Predicted
Answer is correct.
The prediction is correct if it fully aligns with the meaning and key
information of the Golden Answer.
Respond with True if the prediction is correct and False otherwise.


Question:  {question}
Golden Answer:  {reference}
Predicted Answer:  {prediction}
```

Figure 13: Prompt used for LLM-as-Judge evaluation.

Question: Who is Gaius Rubellius Blandus's father-in-law?
Golden Answer: Drusus Julius Caesar

Naive GRPO: <think> To find out who Gaius Rubellius Blandus's father-in-law is, I need to first determine who his wife was. I'll search for information about Gaius Rubellius Blandus to find details about his family, particularly his wife. </think>
<search> Gaius Rubellius Blandus family </search>
<information> ... (information retrieved about Julia Livia being granddaughter of Tiberius, but not focusing on her father) ... </information>
<think> From the information retrieved, we know that Gaius Rubellius Blandus married Julia in AD 33. Now, I need to find out who Julia's father was to determine her father-in-law. </think>
<search> Julia Livia Gaius Rubellius Blandus wife </search>
<information> ... (mentions Julia Livia is granddaughter of Tiberius) ... </information>
<think> From the information retrieved, we know that Julia Livia was the granddaughter of Tiberius. Therefore, Gaius Rubellius Blandus's father-in-law is Tiberius. </think>
<answer> Tiberius </answer>

TIRESRAG-R1: <think> First, I need to identify who Gaius Rubellius Blandus's wife was. Then I need to find out who her father was. </think>
<search> who was Gaius Rubellius Blandus's wife </search>
<information> Gaius Rubellius Blandus married Julia Livia, granddaughter of Emperor Tiberius, in AD 33. ... </information>
<think> Now that I know his wife is Julia Livia, I need to find out who Julia Livia's father was. </think>
<search> who was Julia Livia's father </search>
<information> Julia Livia was the daughter of Drusus Julius Caesar and Livilla. ... </information>
<think> Therefore, Gaius Rubellius Blandus's father-in-law is Drusus Julius Caesar. </think>
<answer> Drusus Julius Caesar </answer>

Figure 14: Case study where the naive GRPO model, despite having sufficient information, mistakenly infers the grandfather Tiberius as the father-in-law, while TIRESRAG-R1 correctly reasons through the steps and outputs Drusus Julius Caesar.

Question: Where was the place of burial of John De Ferrers, 1st Baron Ferrers of Chartley's father?
Golden Answer: Staffordshire

Naive GRPO: <think> To find out where John de Ferrers, 1st Baron Ferrers of Chartley's father was buried, I need to first determine who his father was. I'll search for information about John de Ferrers, 1st Baron Ferrers of Chartley to find his father's name. </think>
<search> John de Ferrers, 1st Baron Ferrers of Chartley father </search>
<information> ... (information retrieved about Roger Ferrers of Fiddington Gloucestershire, unrelated to the correct father) ... </information>
<think> From the information retrieved, we know that John de Ferrers, 1st Baron Ferrers of Chartley's father was Roger Ferrers of Fiddington Gloucestershire. Now, I need to find out where Roger Ferrers of Fiddington Gloucestershire was buried. </think>
<search> Roger Ferrers of Fiddington Gloucestershire burial place </search>
<information> ... (mentions Fiddington and other unrelated burial info) ... </information>
<think> Based on the information retrieved, Roger Ferrers of Fiddington Gloucestershire was buried at Fiddington. Therefore, the place of burial of John de Ferrers, 1st Baron Ferrers of Chartley's father was Fiddington, Gloucestershire. </think>
<answer> Fiddington, Gloucestershire </answer>

TERRISRAG-R1: <think> First, we need to determine who John de Ferrers, 1st Baron Ferrers of Chartley's father was. Then we need to find out where his father was buried. </think>
<search> John de Ferrers, 1st Baron Ferrers of Chartley father burial place </search>
<information> John de Ferrers, 1st Baron Ferrers of Chartley (1271–1324) was the son of Robert de Ferrers, 6th Earl of Derby and Alianore de Bohun. ... </information>
<think> From the search results, I can confirm that John de Ferrers, 1st Baron Ferrers of Chartley's father was Robert de Ferrers, 6th Earl of Derby. Now I need to find where he was buried. </think>
<search> Robert de Ferrers, 6th Earl of Derby burial place </search>
<information> Robert de Ferrers, 6th Earl of Derby, was buried at St Thomas's Priory at Stafford, in Staffordshire. ...</information>
<think> Therefore, the place of burial of John de Ferrers, 1st Baron Ferrers of Chartley's father is Stafford, in Staffordshire. </think>
<answer> Stafford, in Staffordshire </answer>

Figure 15: Case study where the naive GRPO model, despite retrieving sufficient information, follows an incorrect reasoning chain by focusing on an unrelated person (Roger Ferrers of Fiddington) and outputs the wrong burial place. In contrast, TIRESRAG-R1 reasons through the correct lineage and finds the true location in Staffordshire.

Question: Who was born first, Dennis E. Nolan or Humberto Anguiano?
Golden Answer: Dennis E. Nolan

Naive GRPO: <think> To determine who was born first between Dennis E. Nolan and Humberto Anguiano, I need to find out their birth dates. I'll first search for Dennis E. Nolan's birth date. </think>
<search> Dennis E. Nolan birth date </search>
<information> ... Dennis E. Nolan (1872–1956), United States Army general. ... </information>
<think> From the search results, I found that Dennis E. Nolan was a United States Army general. Now, I'll search for Humberto Anguiano's birth date. </think>
<search> Humberto Anguiano birth date </search>
<information> ... Humberto Anguiano (born 2 November 1910) was a Mexican modern pentathlete. ...</information>
<think> Based on the information retrieved, Dennis E. Nolan was born in 1872 while Humberto Anguiano was born in 1910. Therefore, Dennis E. Nolan was born first. </think>
<answer> Humberto Anguiano </answer>

TERRISRAG-R1: <think> Let's determine who was born first, Dennis E. Nolan or Humberto Anguiano, by finding their respective birth years. </think>
<search> Dennis E. Nolan birth date </search>
<information> ... Dennis E. Nolan (1872–1956), United States Army general. ... </information>
<think> From the search results, I can confirm that Dennis E. Nolan is a United States Army general born in 1872. </think>
<search> Humberto Anguiano birth date </search>
<information> ... Humberto Anguiano (born 2 November 1910) was a Mexican modern pentathlete. ... </information>
<think> Since Dennis E. Nolan was born in 1872 and Humberto Anguiano was born in 1910, Dennis E. Nolan was born first. </think>
<answer> Dennis E. Nolan </answer>

Figure 16: Case study where the naive GRPO model, despite retrieving the correct birth years (1872 for Dennis E. Nolan and 1910 for Humberto Anguiano), produces an inconsistent reasoning chain and wrongly outputs Humberto Anguiano as born first. In contrast, TIRESRAG-R1 follows a correct reasoning process and gives the right answer, Dennis E. Nolan.