LiR^{3}AG: A Lightweight Rerank Reasoning Strategy Framework for Retrieval-Augmented Generation

Guo Chen[1], **Junjie Huang**[1*], **Huaijin Xie**[2], **Fei Sun**[3], **Tao Jia**[1 4]

[1]College of Computer and Information Science, Southwest University, Chongqing, China
[2]School of Computer and Cyber Sciences, Communication University of China, Beijing, China
[3]State Key Laboratory of AI Safety, Institute of Computing Technology, CAS, Beijing, China
[4]College of Computer and Information Science, Chongqing Normal University
cg1281838223@email.swu.edu.cn, {junjiehuang, taojia}@swu.edu.cn, 202312063032@cuc.edu.cn, sunfei@ict.ac.cn

Abstract

Retrieval-Augmented Generation (RAG) effectively enhances Large Language Models (LLMs) by incorporating retrieved external knowledge into the generation process. Reasoning models improve LLM performance in multi-hop QA tasks, which require integrating and reasoning over multiple pieces of evidence across different documents to answer a complex question. However, they often introduce substantial computational costs, including increased token consumption and inference latency. To better understand and mitigate this trade-off, we conduct a comprehensive study of reasoning strategies for reasoning models in RAG multi-hop QA tasks. Our findings reveal that reasoning models adopt structured strategies to integrate retrieved and internal knowledge, primarily following two modes: Context-Grounded Reasoning, which relies directly on retrieved content, and Knowledge-Reconciled Reasoning, which resolves conflicts or gaps using internal knowledge. To this end, we propose a novel Lightweight Rerank Reasoning Strategy Framework for RAG (LiR^{3}AG) to enable non-reasoning models to transfer reasoning strategies by restructuring retrieved evidence into coherent reasoning chains. LiR^{3}AG significantly reduce the average 98% output tokens overhead and 58.6% inferencing time while improving 8B non-reasoning model's F1 performance ranging from 6.2% to 22.5% to surpass the performance of 32B reasoning model in RAG, offering a practical and efficient path forward for RAG systems.

1 Introduction

Retrieval-Augmented Generation (RAG) has become a powerful paradigm for enhancing Large Language Models (LLMs) by integrating external knowledge into the generation process. By mitigating hallucinations and enabling access to up-to-date, domain-specific, or proprietary information, RAG significantly improves the accuracy, factuality, and traceability of generated responses (Lewis et al. 2020; Ayala and Bechard 2024; Fan et al. 2024; Huang et al. 2025). Among the many downstream applications of RAG, multi-hop Question Answering (QA) stands out as a particularly challenging yet impactful task (Ho et al. 2020). It requires retrieving and reasoning over multiple, often disjoint, pieces of context (or evidence) scattered across different documents or corpora. Compared with vanilla QA, multi-hop QA requires more logical reasoning rather than direct output. Tackling multi-hop QA thus pushes RAG systems beyond surface-level retrieval, demanding capabilities for logical inference and contextual synthesis—hallmarks of more advanced reasoning (Sun et al. 2023).

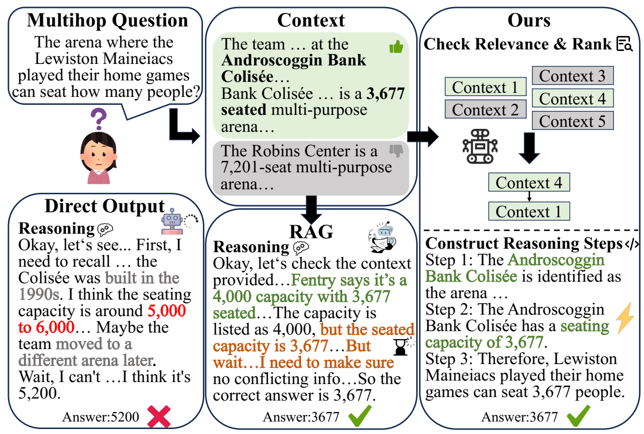

Figure 1: A multi-hop QA example where direct generation hallucinates, RAG answers correctly but with redundant reasoning, and LiR^{3}AG uses relevant evidence to generate concise, accurate reasoning steps and offer the right answers.

Meanwhile, recent advances in reasoning capabilities have led to the emergence of reasoning LLMs (OpenAI 2024). Reasoning models typically make their intermediate reasoning (or think) steps explicit in the generated output. Notable reasoning LLMs include OpenAI's o1 (OpenAI 2024), DeepSeek-R1 (Guo et al. 2025), and Qwen (Yang et al. 2025), among others. Recent works have demonstrated that incorporating reasoning LLMs into the RAG can substantially boost performance on multi-hop QA (Xu et al. 2025). While single-hop QA may benefit sufficiently from retrieval alone, multi-hop QA requires chaining retrieved evidence and resolving inter-document dependencies. Reasoning-enhanced RAG systems can bridge these gaps by producing coherent, logically grounded answers

[*]Junjie Huang is the corresponding author.

based on retrieved content (Islam et al. 2024).

However, integrating reasoning models into RAG comes with substantial computational overhead, as they consume more tokens for intermediate reasoning steps, with longer inference times and redundant generation. The cost is particularly pronounced in multi-hop settings, where both the volume and structure of retrieved context can further amplify reasoning complexity. Balancing reasoning performance with inference efficiency remains an open challenge, particularly for real-world deployment in latency-sensitive scenarios. To address this, we explore whether the reasoning strategies of large models can be replicated using more lightweight alternatives. Currently, while reasoning models are able to reason over retrieved documents, their reasoning behavior remains opaque and poorly understood, especially in complex tasks like multi-hop QA. Furthermore, we conduct a systematic analysis of reasoning models in RAG and uncover two dominant reasoning strategies: **Context-Grounded Reasoning**, where answers are directly derived from retrieved evidence, and **Knowledge-Reconciled Reasoning**, where internal knowledge is used to supplement or verify external content. Our findings highlight the critical role of context-grounded reasoning when relevant information is available, forming the basis for transferring this strategy to non-reasoning models.

Based on this insight, we propose a Lightweight Rerank Reasoning strategy framework for RAG (LIR^{3}AG) to explicitly transfer and enhance the context-grounded reasoning strategy for non-reasoning models. LIR^{3}AG comprises three modules: Retriever, Reranker, and Reasoning Constructor. As shown in Figure 1, given a multi-hop question, direct generation approach leads to hallucination and inaccurate reasoning due to lack of grounded evidence. Reasoning RAG improves accuracy by incorporating external context, but may introduce redundant or verbose reasoning. In contrast, LIR^{3}AG first identifies and reranks the most relevant pieces of evidence in the correct reasoning order, then constructs concise and logically consistent reasoning steps, ultimately yielding more accurate and efficient answers. Empirical results on several multi-hop QA datasets demonstrate that LIR^{3}AG consistently outperforms both vanilla RAG and strong reasoning model baselines, while significantly reducing computational overhead in terms of token usage and inference time. Our work provides a new perspective on reasoning in RAG, showing that structured reasoning strategies can be explicitly modeled and efficiently executed without the computational burden of reasoning LLMs.

Our main contributions are as follows:

- We conduct the first systematic analysis of reasoning strategies in reasoning-augmented RAG models, shedding light on how reasoning models thinking within RAG framework.
- We propose LIR^{3}AG, an innovative and lightweight framework that transfers the reasoning strategy to non-reasoning models. LIR^{3}AG consists of three modules: Retriever, Reranker, and Reasoning Constructor that enable effective reasoning steps generation by non-reasoning models.
- LIR^{3}AG not only achieves state-of-the-art performance among non-reasoning models but also significantly reduces reasoning overhead in terms of token usage and inference latency.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances Large Language Models (LLMs) by integrating external knowledge sources to provide more accurate and contextually relevant responses (Izacard and Grave 2021; Gao et al. 2023). The core strategy of RAG involves using a retriever to fetch highly relevant text snippets based on a query, which are then fed into a generation module to improve output quality (Karpukhin et al. 2020; Fan et al. 2024). This approach has been shown to reduce hallucinations and enhance performance in question-answering tasks (Wang et al. 2022; Ji et al. 2023; Chen et al. 2024). However, these methods have limitations: RAG relies on the reasoning capabilities of LLMs for multi-hop question answering like HotpotQA (Yang et al. 2018), and it struggles to capture interconnections between pieces of information, leading to suboptimal performance in knowledge-intensive tasks (Petroni et al. 2021; Wang et al. 2024) due to incomplete retrieval. To improve the effectiveness of RAG, several knowledge graph-based several have been developed to optimize the retrieval and generation processes (Edge et al. 2024; Guo et al. 2024; Ma et al. 2024) for leveraging global information. In addition to introducing knowledge graphs, a potential research direction is to introduce reasoning models into the RAG system (Jaech et al. 2024).

2.2 Reasoning Model

Reasoning models break down complex problems into intermediate steps, solving them progressively to mimic human logical thinking, often termed Chain-of-Thought (CoT) reasoning (Wei et al. 2022). ReAct (Yao et al. 2022) is a notable reasoning approach that integrates reasoning with action, enabling models to interact with external environments during inference to dynamically gather information and optimize problem-solving. The concept of reasoning model was proposed by OpenAI's o1 (Jaech et al. 2024), which introduced inference-time scaling to allocate more computational resources during reasoning, enabling deliberate, step-by-step problem solving. Subsequent models like Deepseek-R1 (Guo et al. 2025) and QwQ (Qwen Team 2025) have significantly advanced the field. However, challenges persist, particularly in the faithfulness and effectiveness of CoT outputs (Lyu et al. 2023; Feng et al. 2023; Wang, Yue, and Sun 2023). Recent research indicates that reasoning models like these may not always reveal the true reasoning behind their answers, potentially masking their thought processes (Chen et al. 2025; Turpin et al. 2023). Furthermore, issues such as repetitive or fragmented CoT outputs, including looping thoughts or mixed-language responses within a single problem, highlight the need to balance reasoning depth with output quality, while managing the added computational costs during inference (Marjanović et al. 2025).

3 What Reasoning Models Actually Thinking in RAG

In this section, we first introduce the role of reasoning models in RAG, then present an empirical study of their reasoning strategies in multi-hop QA, and finally summarize our findings.

3.1 Reasoning Models in RAG

Retrieval-Augmented Generation (RAG) is widely adopted that enhances language models with access to external knowledge. A typical RAG pipeline consists of two main stages. First, given an input query, a Retriever Module $\mathcal{M}_{\text{Retriever}}$ selects a set of top-k relevant contexts $\mathcal{C}_k = \{C_1, C_2, \ldots, C_k\}$ from corpus:

$$\mathcal{C}_k = \mathcal{M}_{\text{Retriever}}(\text{query}, \text{corpus}). \tag{1}$$

Then, a LLM generator LLM_G conditions on both the input query and the retrieved contexts to produce the final output answer, formulated as:

$$\text{Answer} = \text{LLM}_G(\text{query}, \mathcal{C}_k; \Theta), \tag{2}$$

where Θ denotes the model's parametric knowledge learned during pretraining. Notably, there may exist conflicts between the retrieved contexts $\mathcal{C}_k$ and the model's parametric knowledge Θ. To address these conflicts, reasoning models have been introduced with the aim of reconciling conflicts.

As shown in Table 1, reasoning models as generators markedly enhances inference performance. For example, there is a maximal improvement of 16.0% on the MuSiQue dataset. However, the cost problem of reasoning models remains unsolved. To investigate this, we designed controlled experiments on multi-hop QA tasks, aiming to characterize the underlying reasoning strategies employed in RAG.

3.2 Experiments Validation

To examine these reasoning mechanisms, we used the Qwen3 model on the HotpotQA dataset using a standard Vanilla RAG setup for 500 queries. Notably, reasoning behavior in Qwen3 can be explicitly controlled by injecting a designated label into the prompt[1]. We analyzed the model's intermediate reasoning content to understand how it utilized retrieved content and internal knowledge.

Based on preliminary observations, we propose two hypotheses regarding the reasoning strategy in the RAG framework (For specific examples, please refer to Section 3.3):

1. **Context-Grounded Reasoning**: The model treats the retrieved context as reliable evidence and performs reasoning directly based on it, without substantially invoking its internal knowledge.
2. **Knowledge-Reconciled Reasoning**: The model critically examines the retrieved information, compares it against its own internal knowledge, and resolves inconsistencies or gaps through reasoning.

[1]Qwen3 provides a soft switch mechanism that allows users to dynamically control the model's behavior by adding `/no_think` to user prompts.

To systematically evaluate these hypotheses, we employed GPT-4o-mini to annotate all model outputs based on their reasoning contents. This analysis provides a foundation for identifying the predominant reasoning strategies adopted by the model during inference, and to better understand the reasoning behaviors in RAG. The prompt for annotation can refer to the Appendix.

3.3 Findings

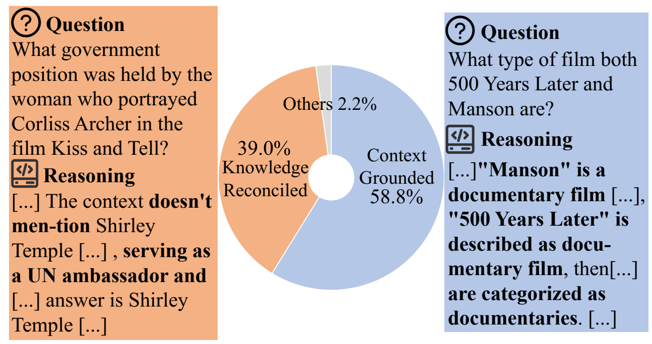

Figure 2: Distribution of annotated reasoning strategies based on model outputs. The majority of responses follow the Context-Grounded Reasoning strategy (58.8%). Two examples are shown to illustrate the feature of each strategy.

As shown in Figure 2, 58.8% of model responses followed the Context-Grounded strategy, while 39.0% aligned with Knowledge-Reconciled reasoning. The remaining 2.2% were ambiguous and excluded from further analysis. Given their marginal proportion, these cases are excluded from further discussion.

The high prevalence of **Context-Grounded Reasoning** indicates that, when the retrieval module provides relevant and sufficient information, the model predominantly relies on this external context as the principal evidence for inference. In such instances, reasoning is conducted with minimal engagement of the model's internal knowledge. As shown in the example in Figure 2, the model directly leverages evidence like the fact that `Manson and 500 Years Later` are both documentaries film to quickly reach the correct answer.

In contrast, **Knowledge-Reconciled Reasoning** usually occurs when the retrieved contexts are irrelevant. Under these conditions, the model draws upon internal knowledge to resolve knowledge conflicts, but this may lead to redundant reasoning. The example in Figure 2 shows that when the provided context is irrelevant, the model first checks the irrelevance, signaled by expressions such as `doesn't mention`. Then introduces its own knowledge, resulting in a complex reasoning process, which may cause lengthy reasoning.

Therefore, we use these findings as the basis for transferring reasoning strategies. We mainly transfer the method of Context-Grounded Reasoning and solve the redundant thinking problem caused by irrelevant context in Knowledge-Reconciled Reasoning, thereby effectively solving the cost problem of reasoning models.

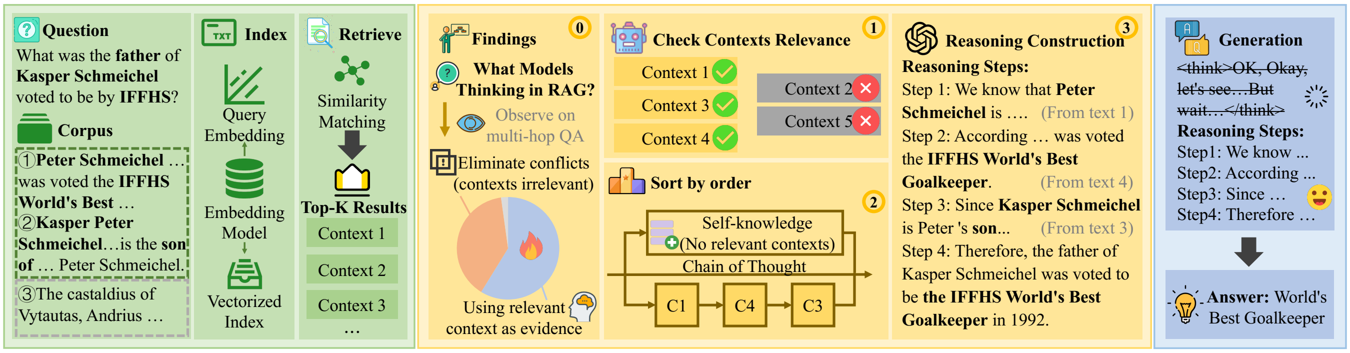

Figure 3: Overall pipeline of our LIR^{3}AG framework. The Retriever first retrieves potentially relevant contexts. Reranker examines their relevance to the question, filters out irrelevant ones, and orders the remaining contexts according to the expected reasoning sequence. Reasoning Constructor assembles these contexts into structured reasoning steps, which are subsequently passed to the generator to produce the final answer.

4 Methodology

In this section, we propose our LIR^{3}AG framework as illustrated in Figure 3, which consists of three main modules: Retriever, Reranker, and Reasoning Constructor. After that, the newly constructed context-enhanced text is subsequently used for downstream answer generation tasks.

4.1 Retriever

For the Retriever Module $\mathcal{M}_{\text{Retriever}}$, common retrieval approaches include sparse retrieval, dense retrieval, and hybrid retrieval. Sparse retrieval typically relies on algorithms such as BM25 to perform lexical matching over the corpus (Robertson, Zaragoza et al. 2009). In contrast, dense retrieval encodes text into vector representations using embedding models and performs semantic matching based on similarity functions (e.g., cosine similarity) to identify relevant context chunks (Karpukhin et al. 2020). Hybrid retrieval combines both sparse and dense retrieval methods, and often incorporates additional post-retrieval modules such as rerankers to further refine the results (Gao et al. 2023; Zhang et al. 2025).

The Retriever Module $\mathcal{M}_{\text{Retriever}}$ returns an ordered list of context passages, each associated with a relevance score, as follows:

$$\mathcal{M}_{\text{Retriever}}(\text{query, corpus}) \rightarrow \langle (C_i, s_i) \rangle_{i=1}^{n}, \tag{3}$$

where $\langle (C_i, s_i) \rangle_{i=1}^{n} = \langle (C_1, s_1), (C_2, s_2), \ldots, (C_n, s_n) \rangle$ denotes a relevance-ranked list of context-score pairs, with C_i representing a retrieved context and s_i its corresponding relevance score. In RAG, the Retriever Module is responsible for selecting the top-k most relevant passages from the corpus given the input query.

4.2 Reranker

For the Reranker Module $\mathcal{M}_{\text{Reranker}}$, we go beyond the naive context reordering commonly employed in vanilla RAG. Instead, inspired by our findings in Section 3, we treat the retrieved contexts as evidence and leverage them to support downstream reasoning. This module comprises two key components: (1) verifying the relevance of each context and retaining only those that are pertinent to the question, and (2) arranging the selected contexts in a reasoning-consistent order. The latter is particularly critical for enabling effective multi-hop question answering.

Formally, the Reranker Module $\mathcal{M}_{\text{Reranker}}$ filters and reorders the retrieved contexts based on their semantic alignment with the query:

$$\mathcal{M}_{\text{Reranker}}\left(\langle (C_i, s_i) \rangle_{i=1}^{n}\right) \xrightarrow{\text{LLM}_R} \langle (\hat{C}_j, \hat{s}_j) \rangle_{j=1}^{m}, \tag{4}$$

where $m \leq n$, and $\hat{C}_j$ and $\hat{s}_j$ are the updated context and responsed relevance score assigned by $\mathcal{M}_{\text{Reranker}}$, typically computed with the assistance of an non-reasoning LLM_R. The output of the Reranker Module serves as the foundation for the subsequent Reasoning Constructor.

4.3 Reasoning Constructor

For the Reasoning Constructor, we leverage a non-reasoning LLM to assemble the filtered contexts into structured reasoning steps, thereby simulating the "thinking" process typically exhibited by reasoning models.

Unlike traditional reasoning models, where the reasoning process is often trained via reinforcement learning and not explicitly optimized for RAG, our method introduces a more targeted construction of reasoning chains $\mathcal{RC}$ tailored for the RAG setting:

$$\mathcal{RC} = f_{\text{template}}\left(\langle (\hat{C}_1, \hat{s}_1), \ldots, (\hat{C}_m, \hat{s}_m) \rangle\right), \tag{5}$$

where f_{template} is a prompt templating function, such as `Reasoning Steps: Step 1` $\hat{C}_1$`. Step 2:` $\hat{C}_2$`.` $\ldots$, which explicitly organizes the retrieved evidence into a multi-step reasoning structure.

Finally, the constructed reasoning chain $\mathcal{RC}$ is passed into a downstream generation model LLM_G to produce the final answer:

$$\text{Answer} = \text{LLM}_G(\mathcal{RC}). \tag{6}$$

In our framework, both LLM_R (used for reranking) and LLM_G (used for answer generation) are non-reasoning models and may differ in scale. Additional configuration details can be found in Section 5.

5 Experiment

In this section, we conduct comprehensive experiments to answer the following research questions.

- **RQ1 (Effectiveness)**: How LIR^{3}AG improves the performance of non-reasoning model in multi-hop QA tasks?
- **RQ2 (Cost Analysis)**: How does the cost of LIR^{3}AG compare to reasoning models?
- **RQ3 (Ablation Study)**: How do individual modules of LIR^{3}AG influence its performance?
- **RQ4 (Parameter Analysis)**: How do different parameter settings of LIR^{3}AG affect its performance?

5.1 Experiments Setup

Datasets We utilize four widely used multi-hop benchmark datasets: HotpotQA (Yang et al. 2018), 2WikiMultihopQA (Ho et al. 2020), MultiHop-RAG (Tang and Yang 2024) and MuSiQue (Trivedi et al. 2022). These multi-hop QA tasks demand models to identify and integrate information from multiple sources in a logical and coherent manner. The core objective is to simulate more realistic and complex human reasoning, where intermediate inference steps are necessary to reach the correct answer. These steps require methods to not only retrieve relevant context fragments, but also to reasonably reason about the relationships between them.

Baselines To evaluate the effectiveness of our method, we set up two baselines: **Direct Output** directly feeds the input question into the original model without using any external context, and the model generates answers based on its internal knowledge. This method reflects the performance of pure language models in the absence of document support. **Vanilla RAG** adopts the standard RAG framework, in which we retrieve the most relevant document paragraphs from the embedded corpus, and then concatenate them with the question and input them into the generation model. This setting represents the basic form of the current mainstream RAG method.

These baselines provide direct and representative comparison objects for our subsequent methods. Among them, **Direct Output** is used as a reference for the performance of the model itself, while **Vanilla RAG** is used as the baseline of the current mainstream methods. Although we have not compared with more complex models, these two can already preliminarily reflect the performance of the reasoning model in the RAG task and the effectiveness of our method.

Evaluation Metrics To evaluate the performance, we adopt two metrics: Exact Match (EM) and F1 Score. EM measures whether the output answer exactly matches the correct answer, while F1 evaluates the word-level overlap between the output result and the answer, thus capturing precision and recall. With these two metrics, we can evaluate the accuracy of the generated answers and thus observe the performance of the model.

Implementation Details. For evaluation, we selected three Qwen3 models of varying sizes(include 8B, 14B and 32B). Qwen3 allows control over its reasoning mode by injecting specific tags into the prompt, enabling or disabling reasoning as needed. We explored two modes of Qwen3 models: with and without reasoning capabilities (referred to as think and no-think modes respectively), to intuitively assess the impact of reasoning on overall performance. For each model, we observe its direct output results without context and its performance as a backbone model in Vanilla RAG. In our method, we selected Qwen3-8B as the generator with the smallest parameters, and GPT-4o-mini as the intermediate component backbone.More implementation details can be found in Appendix.

5.2 Performance Analysis (RQ1)

Table 1 presents the performance comparison of LIR^{3}AG against baseline methods on four multi-hop QA datasets. We evaluated the methods across multiple backbone models and propose the following conclusions:

- **LIR^{3}AG improves the performance of multi-hop question answering to the state-of-the-art and effectively learns the reasoning strategy** . Across all datasets, our method achieves the best performance, demonstrating its effectiveness in enhancing multi-hop reasoning. Notably, even with the smallest backbone Qwen3-8B, LIR^{3}AG exceeds the best Vanilla RAG performance obtained with significantly larger models. This demonstrates that LIR^{3}AG can effectively learn the reasoning strategy, leading to significant gains.

- **Reasoning generally improves QA performance but may introduce redundancy.** Across all baseline settings, reasoning models almost consistently outperform their non-reasoning counterparts. This suggests that step-by-step helps models perform more structured reasoning and better handle complex multi-hop questions. Interestingly, we observe an exception on MultiHop-RAG. Within the Vanilla RAG setting, reasoning models exhibit slightly lower performance compared to non-reasoning models. This deviation may be because the dataset is designed around retrieval paragraphs that have been structured for multi-hop reasoning, so that additional reasoning instructions could introduce unnecessary verbosity or overthinking, potentially leading the model away from the most direct answer path.

5.3 Cost Analysis (RQ2)

In this experiment, we conducted a comparative analysis to evaluate the cost efficiency of LIR^{3}AG using Qwen3-8B. For each method, we measured tokens (both input and output) and average inference time. For token costs, input tokens include the query and any retrieved context provided to the downstream generator, output tokens cover all generated tokens, including reasoning steps. For inference time, it contains the entire process from input to final output (i.e., full end-to-end reporting). These two analyses demonstrate that LIR^{3}AG provides substantial efficiency gains over reasoning models, both in terms of token usage and inference time.

Method	Backbone LLM	HotpotQA EM	HotpotQA F1	2WikiMultihopQA EM	2WikiMultihopQA F1	MultiHop-RAG EM	MultiHop-RAG F1	MuSiQue EM	MuSiQue F1
Direct Output	8B-no-think	0.172	0.257	0.370	0.382	0.492	0.506	0.038	0.109
	8B-think	0.185	0.285	0.388	0.408	0.529	0.541	0.053	0.143
	14B-no-think	0.150	0.236	0.270	0.276	0.552	0.554	0.024	0.071
	14B-think	0.228	0.316	0.390	0.404	0.546	0.561	0.084	0.174
	32B-no-think	0.176	0.246	0.370	0.398	0.580	0.597	0.032	0.135
	32B-think	0.254	0.358	0.394	0.408	0.608	0.621	0.086	0.190
Vanilla RAG	8B-no-think	0.310	0.403	0.462	0.489	0.602	0.611	0.134	0.239
	8B-think	0.328	0.445	0.562	0.604	0.596	0.613	0.239	0.379
	14B-no-think	0.342	0.441	0.502	0.537	0.628	0.637	0.160	0.277
	14B-think	0.356	0.457	0.562	0.603	0.598	0.615	0.232	0.350
	32B-no-think	0.346	0.452	0.468	0.503	**0.658**	0.668	0.182	0.316
	32B-think	0.380	0.501	0.579	0.634	0.642	0.654	0.271	0.410
Ours	8B-no-think	**0.402**	**0.521**	**0.586**	**0.653**	**0.658**	**0.673**	**0.326**	**0.464**

Table 1: The performance of LIR^{3}AG and baseline methods using EM and F1 on four multi-hop QA datasets. **Bold** represents the best results, while underlined denotes the second-best. LIR^{3}AG achieves the SOTA performance even when using the smallest backbone (Qwen3-8B).

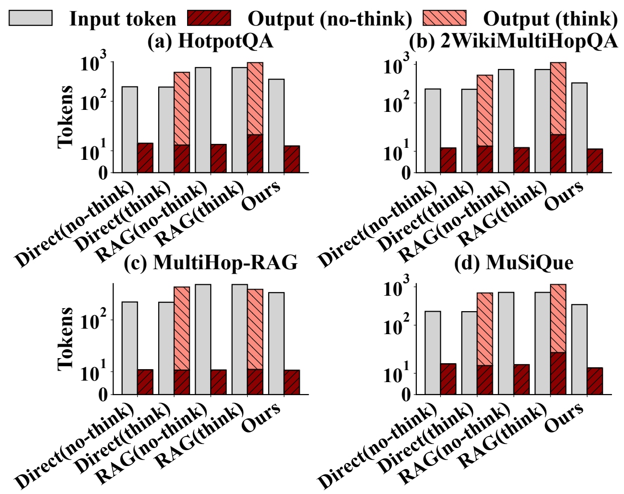

Figure 4: Token cost of methods on multi-hop QA datasets.

Token Cost Analysis As shown in Figure 4, LIR^{3}AG consistently reduces the total number of tokens used during inference compared to reasoning models. The optimization of tokens mainly comes from the fact that LIR^{3}AG directly eliminates conflicts in the context retrieved by retriever and generates clear reasoning steps. This avoids the knowledge conflicts and repeated redundant thinking that may occur in reasoning models.

Inference Time Analysis Figure 5 presents the comparison of inference latency between LIR^{3}AG and reasoning-based models. The results indicate that LIR^{3}AG not only saves token budget but also achieves noticeable improvements in inference speed. With the efficient generation by non-reasoning models, we improved our thinking efficiency by migrating and simplifying the way reasoning models are thought of in the RAG workflow.

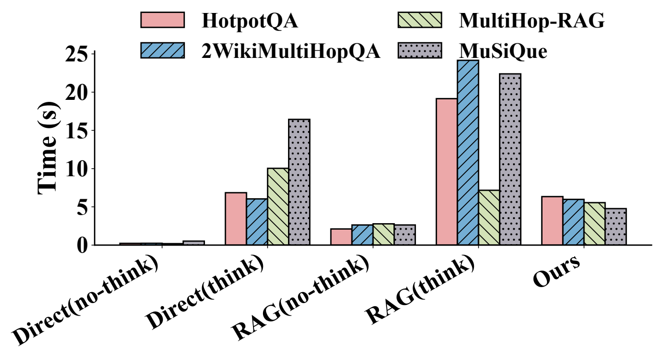

Figure 5: Time cost of methods on multi-hop QA datasets.

5.4 Ablation Study (RQ3)

To verify the contribution of each module in LIR^{3}AG, we conduct ablation studies by removing the Retriever, Reranker, and Reasoning Constructor modules individually. As shown in Table 2, removing any of the three modules leads to substantial performance degradation on both MultiHopRAG and MuSiQue.

The Retriever has the most pronounced impact on MuSiQue, where its removal causes a drastic drop in performance. This suggests that our framework does not overly rely on the Reasoning Constructor—despite being backed by a strong language model, the Reasoning Constructor alone cannot compensate for the absence of relevant context. The Reranker and Reasoning Constructor also contribute significantly—excluding the Ranker yields the lowest score on MuSiQue (F1: 0.233), while removing the Reasoning Constructor results in a comparable drop (F1: 0.238). These results indicate that both selecting high-quality evidence and constructing coherent reasoning chains are also crucial for accurate answer generation.

These findings demonstrate that LIR^{3}AG benefits from a synergistic design, where the Retriever identifies relevant contexts, the Reranker checks and sorts it, and the Reasoning Constructor integrates it into a structured reasoning path.

Model	Retriever	Reranker	Reasoning Constructor	MultiHopRAG		MuSiQue	
				EM	F1	EM	F1
LIR^{3}AG	✓	✓	✓	**0.658**	**0.673**	**0.326**	**0.464**
w/o Retriever	✗	✓	✓	0.606	0.620	0.178	0.293
w/o Reranker	✓	✗	✓	0.610	0.619	0.136	0.233
w/o Reasoning Constructor	✓	✓	✗	0.618	0.625	0.138	0.238

Table 2: Ablation study of three modules, where ✓ and ✗ indicate presence and removal of a module respectively. The table shows that three modules have positive effect on LIR^{3}AG.

Top-k	Reasoning Constructor Input	MultiHopRAG			MuSiQue		
		EM	F1	Generator Input	EM	F1	Generator Input
k=1	353	0.586	0.597	437.23	0.240	0.356	339.05
k=5	770	0.658	0.673	458.82	0.326	0.464	344.20
k=10	**1,284**	**0.696**	**0.707**	**481.96**	**0.352**	**0.497**	**356.45**

Table 3: Effect of varying Top-k on model performance and input length. Larger values of k improve EM and F1 scores on both datasets, while also increasing the number of tokens passed to both Reasoning Constructor and downstream generator.

5.5 Parameter Analysis (RQ4)

In our experiments, we conducted a detailed analysis of two important factors that influence the performance of LIR^{3}AG: the number of retrieved contexts (i.e., the Top-k value) and the model size of the Reranker and Reasoning Constructor module. These two aspects reflect the balance between performance and cost.

The value of k To evaluate the impact of retrieval depth, we set the value of k to 1, 5, 10 and evaluate its performance and cost on the MultiHopRAG and MuSiQue. As shown in Table 3, increasing k consistently improves EM and F1 scores. For example, on MultiHopRAG, F1 improves from 0.597 at k=1 to 0.707 at k=10, suggesting that retrieving more evidence enhances multi-hop reasoning by providing richer context.

However, larger k also increases input length for both the Reasoning Constructor and Generator, leading to higher inference latency and token cost. This reveals a trade-off: higher k enhances context and accuracy, but also raises computational overhead and may introduce irrelevant content.

The model size of Reranker and Reasoning Constructor We investigated the impact of model size on the performance of LIR^{3}AG by varying the backbone model used in the middleware. Model size is a proxy for model capability—larger models generally possess stronger reasoning and representation abilities. As shown in Table 4, stronger models consistently yield better performance across both MultiHopRAG and MuSiQue. These results confirm that model capability (related to model size) plays a pivotal role in improving retrieval quality and final answer accuracy.

Stronger models deliver greater capabilities at the expense of increased computational cost and latency. This highlights a fundamental trade-off in system design: larger models offer better performance, but also demand more computational resources. Therefore, practical deployments of LIR^{3}AG must consider the balance between performance needs and cost constraints when selecting the appropriate model size.

Model	MultiHopRAG		MuSiQue	
	EM	F1	EM	F1
Qwen3-8B	0.616	0.634	0.234	0.344
Qwen3-32B	0.624	0.635	0.248	0.371
GPT-4o-mini	**0.658**	**0.673**	**0.326**	**0.464**

Table 4: Effect of model size on EM and F1 performance across two datasets. Larger models like GPT-4o-mini lead to better performance.

6 Conclusion

In this paper, we introduce LIR^{3}AG, a framework designed to enhance non-reasoning models to acquire reasoning capabilities by strategy transferring within RAG systems. We investigated and defined the reasoning strategy of the reasoning model in RAG workflow, and transferred it. This makes LIR^{3}AG equips non-reasoning models with the ability to generate structured and interpretable reasoning steps. Our approach achieves notable gains in performance while reducing the computational costs commonly associated with explicit reasoning, including tokens and inference time. These results highlight the efficiency and reliability of our method for complex multi-hop tasks and suggest new directions for reasoning-centric RAG. As future work, we plan to integrate LIR^{3}AG with more advanced RAG baselines.

Acknowledgments

This work is supported by the Natural Science Foundation of China (No.62402398, No.72374173), the Natural Science Foundation of Chongqing (No. CSTB2025NSCQGPX1082), the University Innovation Research Group of Chongqing (No.CXQT21005), the Technological Innovation and Application Development Project of Chongqing (No.CSTB2025TIAD-KPX0027), the Fundamental Research Funds for the Central Universities (No.SWU-KR24025, No.SWU-XDJH202303), Open Funding Programs of State Key Laboratory of AI Safety, Lab of High Confidence Embedded Software Engineering Technology and the High Performance Computing clusters at Southwest University.

References

Ayala, O.; and Bechard, P. 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)*, 228–238.

Chen, J.; Lin, H.; Han, X.; and Sun, L. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17754–17762.

Chen, Y.; Benton, J.; Radhakrishnan, A.; Uesato, J.; Denison, C.; Schulman, J.; Somani, A.; Hase, P.; Wagner, M.; Roger, F.; et al. 2025. Reasoning Models Don't Always Say What They Think. *arXiv preprint arXiv:2505.05410*.

Edge, D.; Trinh, H.; Cheng, N.; Bradley, J.; Chao, A.; Mody, A.; Truitt, S.; Metropolitansky, D.; Ness, R. O.; and Larson, J. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Fan, W.; Ding, Y.; Ning, L.; Wang, S.; Li, H.; Yin, D.; Chua, T.-S.; and Li, Q. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 6491–6501.

Feng, G.; Zhang, B.; Gu, Y.; Ye, H.; He, D.; and Wang, L. 2023. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36: 70757–70798.

Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, H.; and Wang, H. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Guo, Z.; Xia, L.; Yu, Y.; Ao, T.; and Huang, C. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

Ho, X.; Nguyen, A.-K. D.; Sugawara, S.; and Aizawa, A. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2): 1–55.

Islam, S. B.; Rahman, M. A.; Hossain, K.; Hoque, E.; Joty, S.; and Parvez, M. R. 2024. Open-rag: Enhanced retrieval-augmented reasoning with open-source large language models. *arXiv preprint arXiv:2410.01782*.

Izacard, G.; and Grave, E. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, 874–880.

Jaech, A.; Kalai, A.; Lerer, A.; Richardson, A.; El-Kishky, A.; Low, A.; Helyar, A.; Madry, A.; Beutel, A.; Carney, A.; et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y. J.; Madotto, A.; and Fung, P. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12): 1–38.

Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P. S.; Wu, L.; Edunov, S.; Chen, D.; and Yih, W.-t. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6769–6781.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474.

Lyu, Q.; Havaldar, S.; Stein, A.; Zhang, L.; Rao, D.; Wong, E.; Apidianaki, M.; and Callison-Burch, C. 2023. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*.

Ma, S.; Xu, C.; Jiang, X.; Li, M.; Qu, H.; and Guo, J. 2024. Think-on-graph 2.0: Deep and interpretable large language model reasoning with knowledge graph-guided retrieval. *arXiv e-prints*, arXiv–2407.

Marjanović, S. V.; Patel, A.; Adlakha, V.; Aghajohari, M.; BehnamGhader, P.; Bhatia, M.; Khandelwal, A.; Kraft, A.; Krojer, B.; Lù, X. H.; et al. 2025. DeepSeek-R1 Thoughtology: Let's think about LLM Reasoning. *arXiv preprint arXiv:2504.07128*.

OpenAI. 2024. Learning to reason with LLMs. Accessed: 2024-09-12.

Petroni, F.; Piktus, A.; Fan, A.; Lewis, P.; Yazdani, M.; De Cao, N.; Thorne, J.; Jernite, Y.; Karpukhin, V.; Maillard, J.; et al. 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*, 2523–2544.

Qwen Team. 2025. QwQ-32B: Embracing the Power of Reinforcement Learning. https://qwenlm.github.io/blog/qwq-32b/. Accessed: 2025-07-13.

Robertson, S.; Zaragoza, H.; et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4): 333–389.

Sun, J.; Zheng, C.; Xie, E.; Liu, Z.; Chu, R.; Qiu, J.; Xu, J.; Ding, M.; Li, H.; Geng, M.; et al. 2023. A survey of reasoning with foundation models. *arXiv preprint arXiv:2312.11562*.

Tang, Y.; and Yang, Y. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10: 539–554.

Turpin, M.; Michael, J.; Perez, E.; and Bowman, S. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36: 74952–74965.

Wang, B.; Yue, X.; and Sun, H. 2023. Can ChatGPT defend its belief in truth? evaluating LLM reasoning via debate. *arXiv preprint arXiv:2305.13160*.

Wang, L.; Yang, N.; Huang, X.; Jiao, B.; Yang, L.; Jiang, D.; Majumder, R.; and Wei, F. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Wang, M.; Chen, L.; Cheng, F.; Liao, S.; Zhang, X.; Wu, B.; Yu, H.; Xu, N.; Zhang, L.; Luo, R.; et al. 2024. Leave no document behind: Benchmarking long-context llms with extended multi-doc qa. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 5627–5646.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xu, F.; Hao, Q.; Zong, Z.; Wang, J.; Zhang, Y.; Wang, J.; Lan, X.; Gong, J.; Ouyang, T.; Meng, F.; et al. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2369–2380.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K. R.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Zhang, L.; Wang, B.; Qiu, X.; Reddy, S.; and Agrawal, A. 2025. REARANK: Reasoning Re-ranking Agent via Reinforcement Learning. *arXiv preprint arXiv:2505.20046*.