# Integrating Chain-of-Thought into Generative Retrieval: A Preliminary Study

Anonymous ACL submission


## Abstract

While generative retrieval (GR) demonstrates competitive performance on standard retrieval benchmarks, existing approaches directly map queries to document identifiers (docids) without intermediate deliberation, limiting their effectiveness for complex queries that require multi-step reasoning. As a preliminary study on integrating chain-of-thought (CoT) into generative retrieval, we introduce ThinkGR, a unified framework that interleaves CoT with docid generation, enabling iterative thinking and retrieval within a single generative process. To bridge the gap between free-form thought generation and structured retrieval targets, we design (1) a hybrid decoding strategy that dynamically switches between unconstrained thought generation and constrained docid decoding, and (2) a two-phase training approach that first aligns thought-retrieval patterns through supervised fine-tuning, then optimizes thought quality via retrieval-grounded reinforcement learning. Experiments on four multi-hop retrieval benchmarks demonstrate that ThinkGR achieves state-of-the-art performance with an average improvement of +6.86%. Our work opens new avenues for enhancing generative retrieval with explicit deliberation capabilities, with promising implications for retrieval tasks requiring complex reasoning. [1]


## 1 Introduction

Generative Retrieval (GR) has emerged as a promising paradigm that reformulates document retrieval as a sequence generation task (De Cao et al., 2021; Tay et al., 2022; Bevilacqua et al., 2022). Unlike traditional dense retrieval methods that rely on embedding similarity matching, GR directly generates document identifiers (docids) from queries by leveraging generative model architectures. This approach enables end-to-end optimization and has demonstrated competitive performance on standard retrieval benchmarks (Zhou et al., 2022; Zhang et al., 2024; Sun et al., 2023). Recent work by Zhang et al. (2025c) reproduced representative DR and GR under matched backbones, and through controlled and transparent comparisons, showed that GR surpasses standard dual-encoder dense retrievers on Natural Questions (Kwiatkowski et al., 2019).

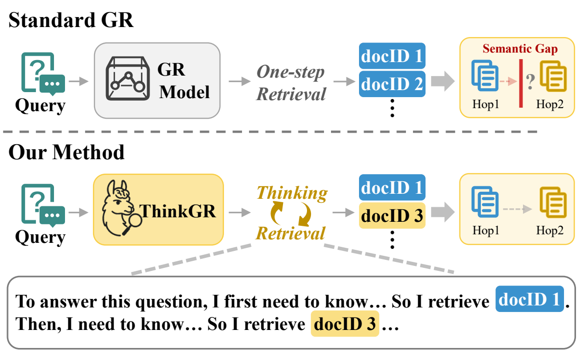


Figure 1: Comparison between standard generative retrieval and our thought-augmented approach. (a) Standard GR: Directly generates docids from the query without intermediate deliberation. (b) ThinkGR: Interleaves chain-of-thought with docid generation, enabling iterative thinking and retrieval within a single forward pass.

Despite these advances, existing GR methods share a fundamental limitation: they directly map queries to docids without intermediate deliberation. This single-step generation paradigm works well for straightforward queries where the target document is semantically close to the query surface form. However, when queries require complex reasoning, such as in multi-hop retrieval scenarios where resolving the query necessitates traversing semantic connections across multiple documents (Yang et al., 2018; Trivedi et al., 2022), the lack of an explicit thinking process severely limits its retrieval effectiveness. For example, given the query "*What is the capital of the country where the*

[1]Our code is available at https://anonymous.4open.science/r/GenRAR-6357

*director of Erta Ale was born?*", answering it requires first retrieving documents about the director of Erta Ale, then retrieving documents about the corresponding country's capital—a scenario where standard single-hop retrieval models struggle to retrieve the final target document, as it lacks direct semantic similarity to the original query.

Inspired by recent advances in chain-of-thought (CoT) prompting (Wei et al., 2022) and deliberative reasoning models (Jaech et al., 2024; Guo et al., 2025), we investigate enabling GR to interleave reasoning and retrieval within a unified generative process. This iterative paradigm can bridge the semantic gap between complex queries and target documents through intermediate reasoning steps (Figure 1). However, effectively merging the thought process into retrieval models remains challenging due to the inherent differences between open-ended reasoning and discrete retrieval actions.

To address this challenge, we propose ThinkGR, a unified framework that integrates CoT into GR, enabling the model to iteratively think and retrieve within a single generative process. To unify the decoding space, we design a hybrid decoding strategy that dynamically switches between unconstrained generation for thought tokens and constrained decoding for docids, using semantic triples as document representations that naturally bridge natural language and structured retrieval targets. To unify the learning process, we employ a two-phase training strategy: first aligning thought-retrieval patterns through supervised fine-tuning, then optimizing thought quality using retrieval accuracy as a grounded reward signal.

We conduct extensive experiments on four multi-hop retrieval benchmarks: HotpotQA, 2WikiMultihopQA, MuSiQue, and MoreHopQA. Results demonstrate that ThinkGR achieves state-of-the-art performance, significantly outperforming existing methods with an average improvement of +6.86%.

Our contributions can be summarized as follows: (1) This work represents one of the earliest explorations of integrating chain-of-thought into generative retrieval, demonstrating the feasibility and potential of this paradigm. (2) We identify key challenges in enabling iterative thinking and retrieval, and propose solutions including a hybrid decoding strategy and a two-phase training approach. (3) ThinkGR achieves state-of-the-art results on four benchmarks, demonstrating its effectiveness in multi-hop retrieval scenarios that require complex thinking.

## 2 Related Work

### 2.1 Generative Retrieval

Generative retrieval formulates document retrieval as a sequence generation task, directly producing docids through autoregressive decoding (De Cao et al., 2021; Bevilacqua et al., 2022; Sun et al., 2023). Previous research has explored various types of docids, which can be categorized into lexical ids and numeric ids. Lexical ids include title (De Cao et al., 2021), substring (Bevilacqua et al., 2022), URL (Zhou et al., 2022), termsets (Zhang et al., 2024), etc., while numeric ids are typically obtained through clustering of document representations (Tay et al., 2022; Zhou et al., 2022; Sun et al., 2023). This paradigm enables end-to-end optimization and has shown competitive performance on standard benchmarks (Zhang et al., 2025c,b).

While GR has achieved promising results, existing approaches lack intermediate deliberation mechanisms, limiting their effectiveness on complex queries. Although Lee et al. (2022) applied GR to multi-hop settings using document fragments, their approach lacks explicit thought mechanisms and faces scalability challenges for large corpora. Our work addresses this gap by integrating chain-of-thought into the generative retrieval framework.

### 2.2 Thought-Augmented Retrieval

Recent advances in chain-of-thought prompting (Wei et al., 2022) and deliberative reasoning (Jaech et al., 2024; Guo et al., 2025) have inspired efforts to incorporate thought processes into retrieval systems.

**LLM-Driven Multi-Step Retrieval.** One line of work leverages LLMs' intrinsic reasoning capabilities to decompose complex queries and guide iterative retrieval. ReAct (Yao et al., 2023) alternates between reasoning steps and retrieval actions; Self-ask (Press et al., 2023) decomposes questions into sub-questions with follow-up Q&A; IRCoT (Trivedi et al., 2023) interleaves retrieval with stepwise reasoning to refine subsequent searches. Recent methods employ reinforcement learning: Auto-RAG (Yu et al., 2024) trains LLMs to decide retrieval timing; DeepRAG (Guan et al., 2025) models the process as a Markov Decision Process; R3-RAG (Li et al., 2025) uses outcome and process rewards for optimization. However, these approaches require separate LLM and retriever modules with iterative handoffs, preventing end-to-end optimization.

**Reasoning-Augmented Dense Retrieval.** Another line integrates thought processes into dense retrieval models. Early methods incorporate thought implicitly without explicit tokens: MDR (Xiong et al., 2021) iteratively encodes queries with retrieved context to guide subsequent retrieval; GRITHopper (Erker et al., 2025) unifies language modeling and contrastive retrieval in a single LLM; O1 Embedder (Yan et al., 2025) leverages LLM-generated "thoughts" as intermediate representations before aggregating into embeddings. Recent work explores explicit thought tokens before embedding generation: GEM (Zhang et al., 2025a) inserts bottleneck tokens with specialized attention masks to compress semantic information; LREM (Tang et al., 2025) establishes a "think-then-embed" paradigm where models generate keyword-form CoT before final embeddings. While these methods demonstrate the value of explicit thought for retrieval, they operate within dense retrieval and typically perform thought generation only once before embedding. In contrast, our work integrates thought into generative retrieval, enabling *iterative* thought-retrieval interleaving within a single decoding pass, particularly beneficial for multi-hop scenarios requiring progressive reasoning.

# 3 Method

To realize the idea of integrating chain-of-thought into generative retrieval, we develop ThinkGR as a preliminary instantiation. ThinkGR formulates retrieval as a unified sequence generation problem, generating an interleaved sequence of thought tokens and docids through a single forward pass, where each thought segment contextualizes and guides the subsequent retrieval decision. We implement ThinkGR by training Llama-3.1-8B-Instruct with a two-phase training strategy. Figure 2 shows an overview of our method.

## 3.1 Semantic Triple Representation

A critical design choice in generative retrieval is how to represent documents as generation targets. Traditional docids fail to capture the relational semantics essential for complex queries. We observe that multi-hop queries inherently require traversing semantic relationships between entities, a structure naturally expressed as knowledge triples.

We thus represent documents as structured knowledge triples (head entity, relation, tail entity), serving as docids. This design serves two purposes: (1) triples encode fine-grained semantic relationships, enabling the model to perform semantic traversal through explicit relation following, and (2) natural-language triples allow the model to leverage the pre-trained LLM's semantic understanding to generalize to unseen entities and relations. We designed a prompt to instruct LLM to generate these triples. To construct SFT data, we further prompt the LLM to generate thought-retrieval chains based on the correct document triples and questions. Detailed prompts are provided in Appendix A. To ensure high-quality training data and minimize cascading errors, we implemented a rigorous filtering process that removes samples with formatting errors, incorrect triples, and factual inaccuracies. This process resulted in a high-quality curated set of 228K SFT data from HotpotQA, 2WikiMultihopQA, and Musique datasets.

## 3.2 Thought-Retrieval Alignment

The first training phase establishes the model's ability to generate interleaved thought-retrieval sequences. By focusing on this structural alignment, we enable the model to utilize its pre-trained knowledge for robust generalization to unseen entities and relations, even if they were not explicitly seen during fine-tuning. In this process, the generated thought steps naturally guide the retrieval operations, forming a coherent and effective workflow. Formally, given input query $x$, the model learns to generate an interleaved sequence $y = (r_1, d_1, r_2, d_2, \ldots)$ where $r_i$ denotes thought segments and $d_i$ denotes docids. The training objective minimizes the negative log-likelihood:

$$\mathcal{L}_{\text{SFT}} = -\sum_t \log P(y_t | y_{<t}, x), \qquad (1)$$

where $y_t$ denotes each token in the output sequence. After this phase, the model acquires the basic thought-retrieval generation capability, preparing it for further optimization.

## 3.3 Retrieval-Grounded Thought Optimization

While the alignment phase teaches the model the structural pattern of thought-retrieval interleaving, it is limited by the quality of demonstration data and cannot discover better thought strategies beyond imitation. To address this, we introduce a retrieval-grounded reinforcement learning phase

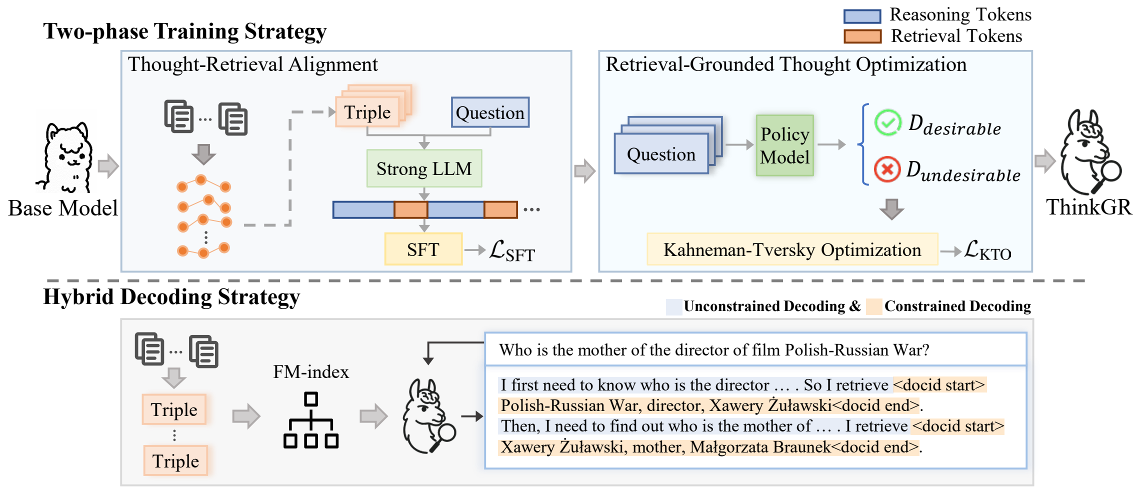


Figure 2: Overview of the ThinkGR framework. Top: Two-phase training strategy (Thought-Retrieval Alignment followed by Retrieval-Grounded Thought Optimization). Bottom: Hybrid Decoding Strategy combining constrained and unconstrained decoding.

that uses retrieval accuracy as a proxy reward for thought quality. The key insight is that in our unified framework, the quality of thought directly manifests in retrieval outcomes. This creates a natural feedback loop: we can use retrieval accuracy as an automatic and grounded reward signal to optimize the thought process, without requiring expensive human annotations of thought quality.

We implement this using Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024). Unlike PPO (Schulman et al., 2017) or DPO (Rafailov et al., 2023) which require preference pairs, KTO operates effectively with binary feedback, allowing us to directly leverage retrieval correctness as supervision. Specifically, if the generated docids match the ground-truth exactly, the response is labeled as *desirable*; if the accuracy of generated docids is less than $\tau$, it is labeled as *undesirable*. This creates a grounded optimization objective that directly ties reasoning quality to task performance. Formally, we partition model-generated responses based on retrieval accuracy:

$$D_{\text{desirable}} = \{(x, y) | \text{Acc}_{\text{r}}(y) = 1\}, \tag{2}$$

$$D_{\text{undesirable}} = \{(x, y) | \text{Acc}_{\text{r}}(y) < \tau\}, \tag{3}$$

$$\text{Acc}_{\text{r}}(y) = \frac{|\text{docids}(y) \cap \text{docids}_{\text{gr}}|}{|\text{docids}_{\text{gr}}|}, \tag{4}$$

where $\text{docids}(y)$ denotes the docids generated by the model, and $\text{docids}_{\text{gr}}$ denotes the ground-truth docids. After filtering, we obtained 60K desirable responses and 27K undesirable responses. KTO can implicitly handle the data imbalance through adaptive weighting, so we did not perform further balanced sampling. We optimize the model weights based on the following loss function:

$$\begin{aligned} \mathcal{L}_{\text{KTO}} = &\mathbb{E}_{(x,y)\sim D_{\text{desirable}}}[\lambda_d - v(x, y)] \\ &+ \mathbb{E}_{(x,y)\sim D_{\text{undesirable}}}[\lambda_u - v(x, y)], \end{aligned} \tag{5}$$

where

$$r_\theta(x, y) = \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

$$z_0 = KL(\pi_\theta(y'|x) || \pi_{\text{ref}}(y'|x))$$

$$v(x, y) = \begin{cases} \lambda_d(\beta(r_\theta(x, y) - z_0)) & \text{if } y \in D_{\text{desirable}} \\ \lambda_u(\beta(z_0 - r_\theta(x, y))) & \text{if } y \in D_{\text{undesirable}}, \end{cases}$$

where $v(x, y)$ represents the prospect-theoretic utility function that converts the model's implicit reward into a human-perceived value relative to a reference point. $\pi_\theta$ refers to the policy model being trained, where $\pi_\theta(y|x)$ denotes the probability of generating output sequence $y$ given input $x$ under parameters $\theta$. $\pi_{\text{ref}}$ represents the reference model, which is the supervised fine-tuned model before reinforcement learning. $r_\theta(x, y)$ is the implied reward, $z_0$ is the reference point dynamically estimated per mini-batch, $\lambda_d$ and $\lambda_u$ are automatically adjusted parameters for data imbalance, and $\beta$ is a risk aversion hyperparameter controlling optimization sensitivity.

### 3.4 Hybrid Decoding Strategy

The unified sequence formulation introduces a unique inference challenge: the model must generate tokens from two different spaces—an open vocabulary for thought and a constrained set of valid docids for retrieval—within a single autoregressive process. Naive unconstrained decoding risks generating invalid docids (hallucinations), while fully constrained decoding would prevent flexible thought generation.

We address this through a hybrid decoding strategy that dynamically switches between unconstrained and constrained generation modes. For constrained docid generation, we employ FM-index (Ferragina and Manzini, 2000) following Bevilacqua et al. (2022), which returns valid next tokens in constant time. We preprocess corpus triples into the format "<docid_start>head entity, relation, tail entity<docid_end>" and store them in the FM-index. During inference, the decoding process operates as follows: (1) ThinkGR begins with unconstrained decoding to generate thought tokens, enabling "slow thinking" where the model elaborates on the query context and determines what information is needed; (2) When the model decides a retrieval is necessary, it generates "<docid_start>", which triggers a switch to constrained decoding mode; (3) In constrained mode, the FM-index restricts the output vocabulary to valid next tokens based on the current prefix, ensuring the generated docid exists in the corpus; (4) Upon generating "<docid_end>", the mode switches back to unconstrained for continued thought.

This hybrid strategy realizes the core idea of thought-augmented retrieval: the model autonomously decides when to retrieve and what to retrieve through explicit deliberation, all within a single autoregressive pass. This design demonstrates the feasibility of unifying free-form thought with constrained retrieval generation, eliminating the latency overhead of iterative LLM-retriever invocations while maintaining retrieval validity.

# 4 Experiments

## 4.1 Datasets and Metrics

Since our method is specifically designed for retrieval scenarios requiring complex reasoning, we evaluate its effectiveness on four commonly used multi-hop QA datasets: **HotpotQA** (Yang et al., 2018), **2WikiMultiHopQA** (Ho et al., 2020), **MuSiQue** (Trivedi et al., 2022), and **MoreHopQA** (Schnitzler et al., 2024). Detailed descriptions and statistics are provided in Appendix B.

We use *Recall* as our primary evaluation metric, defined as the ratio of correct documents retrieved to the total number of ground-truth documents. This metric is a common choice in multi-hop retrieval benchmarks (Trivedi et al., 2023; Erker et al., 2025), allowing for direct and fair comparison with prior work.

## 4.2 Baselines

We compare our method with standard one-hop retrievers, including *BM25* (Robertson et al., 2009), *Contriever* (Izacard et al., 2021), *BGE* (Xiao et al., 2023), and generative retrieval method *SEAL* (Bevilacqua et al., 2022). To comprehensively assess multi-hop retrieval capability, we primarily compare with two categories of methods specifically designed for this task: LLM-driven multi-step retrieval and reasoning-augmented dense retrieval. LLM-driven multi-step retrieval methods include: *Self-Ask* (Press et al., 2023), prompts the LLM to generate follow-up questions based on the current context; *IR-CoT* (Trivedi et al., 2023), uses CoT generated by LLM to guide retrieval; *ITER-RETGEN* (Shao et al., 2023), iteratively performs reasoning and retrieval; *Auto-RAG* (Yu et al., 2024), employs reinforcement learning to enable autonomous decision-making on retrieval through multi-turn LLM-retriever interactions; and *R3-RAG* (Li et al., 2025), optimizes iterative retrieval via reinforcement learning with outcome and process rewards for document relevance verification. Reasoning-augmented dense retrieval methods include: *MDR* (Xiong et al., 2021) iteratively encodes the concatenated question and previously retrieved documents into a single vector for next-hop retrieval; *GRITHopper* (Erker et al., 2025) integrates causal language modeling with contrastive dense retrieval through ReAct-style instruction tuning, leveraging post-retrieval language modeling to contextualize multi-hop reasoning. Implementation details are provided in Appendix C.

# 5 Experimental Results

We evaluate ThinkGR on four multi-hop QA benchmarks to demonstrate the effectiveness of integrating chain-of-thought into generative retrieval.

## 5.1 Main Results

The main results on the four datasets are presented in Table 1. We derive the following observations from the results:

(1) Our method demonstrates state-of-the-art performance, achieving an average performance

| Method | Model Parameters | HotpotQA | 2Wiki. | Musique | Morehopqa | Average |
|---|---|---|---|---|---|---|
| *Standard Retriever* | | | | | | |
| BM25 | - | 46.39 | 49.80 | 31.57 | 43.74 | 42.88 |
| Contriever | - | 50.35 | 51.25 | 34.02 | 45.04 | 45.17 |
| BGE-large | 326M | 60.48 | 58.43 | 33.39 | 47.58 | 49.97 |
| SEAL | 406M | 56.15 | 48.32 | 35.60 | 47.27 | 46.84 |
| *LLM-Driven Multi-Step Retrieval* | | | | | | |
| Selfask | 70B | 44.40 | 47.07 | 34.28 | 57.60 | 45.84 |
| IRCoT | 70B | 55.79 | 65.12 | 49.96 | 66.82 | 59.42 |
| ITER-RETGEN | 70B | 61.94 | 59.96 | 39.37 | 60.73 | 55.50 |
| Auto-RAG | 7B | 54.52 | 66.05 | 40.63 | 59.48 | 55.17 |
| R3-RAG | 8B | 58.56 | <u>82.82</u> | 51.70 | 70.44 | 65.88 |
| *Reasoning-Augmented Dense Retrieval* | | | | | | |
| MDR | 110M | <u>87.57</u> | 53.91 | 27.84 | 49.60 | 54.73 |
| GritHopper | 7B | **91.03** | 59.97 | <u>60.48</u> | <u>74.82</u> | <u>71.58</u> |
| *Our Method* | | | | | | |
| ThinkGR | 8B | 76.09 | **93.19** | **63.98** | **80.50** | **78.44** |
| *w/o* SFT | 8B | 35.76 | 57.70 | 35.98 | 55.14 | 46.15 |
| *w/o* RL | 8B | 67.66 | 92.03 | 53.40 | 73.84 | 71.73 |
| *w/o* Thought | 8B | 69.86 | 93.01 | 53.23 | 78.00 | 73.53 |

Table 1: Performance comparison of ThinkGR with baselines on four datasets. The results are reported in terms of recall. The best results for each dataset are highlighted in **bold**, and the second-best results are <u>underlined</u>. The model parameters indicate the number of parameters in millions (M) or billions (B).

gain of 6.86% over the strongest baseline. This establishes the effectiveness of interleaving iterative thinking with retrieval in a unified generative process. Note that while ThinkGR does not achieve the best result on HotpotQA, this is primarily because HotpotQA suffers from the over-specification issue. This allows models to find relevant documents through simple character matching, making it unfair to evaluate the thought-retrieval capability of models. To illustrate this point, we conducted a statistical analysis of the n-gram overlap between questions and ground-truth documents across three datasets. The results indicate that HotpotQA indeed exhibits a significantly high overlap rate. Detailed statistical results are presented in Appendix F. Despite this, ThinkGR still significantly outperforms all LLM-Driven Multi-Step Retrieval methods on HotpotQA.

(2) ThinkGR exhibits superior generalization ability on out-of-domain evaluation. Compared to GritHopper, ThinkGR achieves a 5.68% improvement on the out-of-domain dataset Morehopqa. This indicates that ThinkGR is more robust and can generalize better to unseen data and schema not encountered during training, which is crucial for real-world applications.

(3) ThinkGR shows better stability across different datasets. It is evident that Reasoning-Augmented Dense Retrieval methods generally outperform LLM-Driven Multi-Step Retrieval methods, but they perform poorly on the 2WikiMultihopQA dataset. ThinkGR achieves state-of-the-art results on both 2WikiMultihopQA (simpler queries) and Musique (more complex queries), demonstrating strong generalization across different difficulty levels. We conducted a statistical analysis and found that GritHopper retrieves an average of 1.79 documents on 2WikiMultihopQA, while the ground-truth documents average 2.44. This indicates that GritHopper retrieves insufficient documents on 2WikiMultihopQA, leading to its suboptimal performance. In contrast, our method ThinkGR retrieves an average of 2.85 documents on 2WikiMultihopQA, further demonstrating the better stability of our method across different datasets.

## 5.2 Ablation Studies

**Effectiveness of Two-Phase Training.** We investigate the effectiveness of the two-phase training strategy in ThinkGR. The results of models trained only with thought-retrieval alignment or retrieval-

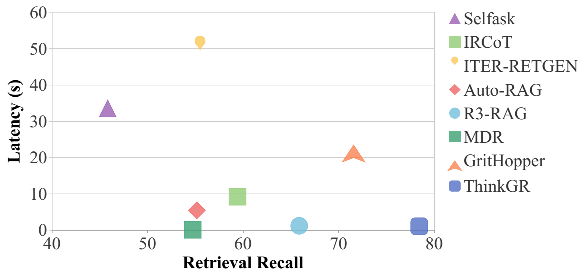


Figure 3: Comparison of effectiveness and efficiency. The x-axis denotes retrieval recall and the y-axis represents average latency per query.

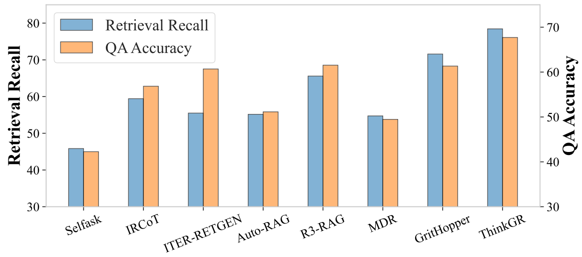


Figure 4: Comprehensive evaluation of retrieval and QA performance. The blue bars show average retrieval recall, while the orange bars represent QA accuracy.

grounded thought optimization are presented in Table 1. From the results, we draw two observations. First, the model trained without retrieval-grounded thought optimization exhibits a significant drop in retrieval performance. This indicates that this optimization phase substantially enhances the model's thought capabilities, enabling it to surpass the performance ceiling achieved by SFT. We illustrate this point further with a specific case study in Appendix H. Second, relying solely on retrieval-grounded thought optimization is insufficient for the model to learn the thought-retrieval workflow. This is evidenced by the overall inferior performance of models trained without prior thought-retrieval alignment. This suggests that supervised training is necessary to adapt the model to the retrieval task.

**Effectiveness of Hybrid Decoding.** We next ablate the hybrid decoding strategy—the interleaved generation of natural language thought tokens and docids within a single output sequence. To evaluate the contribution of this iterative thinking mechanism, we train a variant that generates only the docids sequence, omitting the thought tokens entirely. Essentially, this transforms the explicit thought into implicit thought within the model parameters. The results in Table 1 show that this docids-only variant (w/o Thought) reduces retrieval accuracy, especially on the more challenging dataset Musique. This variant essentially represents a standard constrained decoding strategy where the model directly generates retrieval targets without intermediate thought. This indicates that the interleaved thought tokens are not merely interpretative, but also crucial for bridging semantic gaps across retrieval steps in complex queries. Thus, this ablation study demonstrates that unifying iterative thinking and retrieval into a single generation is superior to methods that rely on implicit reasoning like GritHopper or our docids-only variant. This unified hybrid decoding strategy is essential to ThinkGR, as it enables complex, iterative thought and retrieval in a single end-to-end pass.

### 5.3 Efficiency Comparison

We compare inference efficiency by measuring average latency per query, with results visualized in Figure 3. ThinkGR exhibits higher efficiency than LLM-Driven Multi-Step Retrieval methods, as its end-to-end generation design eliminates the latency incurred by sequential LLM-retriever invocations. While implicit methods like MDR achieve lower latency by avoiding explicit token generation, ThinkGR strikes a compelling balance: generating explicit thought tokens incurs only moderate latency cost while substantially improving retrieval performance on complex queries. Additionally, the FM-index used in ThinkGR is highly space-efficient, requiring significantly less storage than dense retrieval index (detailed in Appendix K).

### 5.4 Impact on Downstream QA

To comprehensively evaluate the effectiveness of our method, we further assess ThinkGR's performance in the complete question answering task. Following established practice in RAG-based QA evaluation, we employ Llama-3.3-70B-Instruct to generate answers based on the retrieved documents and the original question. We use accuracy (Acc) as QA metric, which is determined by further prompting the LLM for judgment. As visually summarized in Figure 4, ThinkGR achieves the highest average QA accuracy across four benchmarks. Its average accuracy exceeds the strongest baseline by 6.37%, indicating that the quality of retrieved documents directly determines the accuracy of downstream answers. Notably, the QA accuracy does not strictly correlate with the retrieval recall. For example, IRCoT achieves a higher retrieval recall than

| Model | HotpotQA | 2WikiMultihopQA | Musique | Morehopqa | Average |
|---|---|---|---|---|---|
| *Llama3* | | | | | |
| Llama-3.2-1B-Instruct | 26.27 | 37.29 | 17.99 | 25.31 | 26.72 |
| Llama-3.2-3B-Instruct | 39.26 | 46.86 | 25.53 | 50.98 | 40.66 |
| Llama-3.1-8B-Instruct | 56.73 | 77.90 | 42.83 | 56.89 | 58.59 |
| Llama-3.3-70B-Instruct | 68.71 | 86.56 | 56.09 | 74.96 | 71.58 |
| *Qwen3* | | | | | |
| Qwen3-0.6B | 10.45 | 1.29 | 2.42 | 0.27 | 3.61 |
| Qwen3-1.7B | 40.50 | 59.31 | 34.10 | 51.57 | 46.37 |
| Qwen3-4B | 54.59 | 79.82 | 42.17 | 62.34 | 59.73 |
| Qwen3-8B | 57.44 | 82.82 | 43.21 | 66.55 | 62.51 |
| Qwen3-14B | 57.82 | 82.67 | 44.85 | 66.73 | 63.02 |
| Qwen3-32B | 60.24 | 83.15 | 47.65 | 69.59 | 65.16 |

Table 2: Retrieval recall performance of off-the-shelf LLMs employing our designed hybrid decoding strategy via few-shot prompting in a training-free setting.

ITER-RETGEN, but the latter outperforms in QA accuracy. This discrepancy arises from the varying number of documents retrieved by different methods. Retrieving more documents can increase recall, but it may also introduce irrelevant documents that negatively impact QA accuracy. Overall, the experimental results show that ThinkGR effectively enhances retrieval quality without compromising downstream QA performance. This further demonstrates ThinkGR's effectiveness as an end-to-end solution for complex information retrieval tasks. The complete results are provided in Table 6.

### 5.5 Performance of Off-the-Shelf LLMs with Few-Shot Prompting

To investigate the effectiveness of our proposed thought-augmented retrieval paradigm in a training-free setting, we examine whether off-the-shelf LLMs can perform hybrid decoding through prompt engineering. We conduct experiments using carefully designed prompts and few-shot demonstrations. Specifically, we test 10 LLM from the Llama3 (1B - 70B) and Qwen3 (0.6B - 32B) families, providing the same instructions and context examples to guide them toward the desired output structure. Since these off-the-shelf models have not been fine-tuned to generate our specific special tokens (e.g., `<docid_start>`), we use square brackets `[` and `]` as substitutes for the start and end tokens of document identifiers. The results in Table 2 reveal three observations: (1) Off-the-shelf LLMs exhibit reasonable performance (e.g., Llama3.3-70B achieves 71.58%), validating the inherent feasibility of our thought-augmented retrieval paradigm. This demonstrates the effectiveness of our designed paradigm even in a training-free setting. (2) Model scale determines feasibility: while larger LLM show progressively better performance, smaller LLM like Qwen3-0.6B fail catastrophically due to insufficient instruction-following ability. (3) Despite promising results, all these models underperform our trained ThinkGR (78.44%), with even the strongest Llama3.3-70B trailing by 6.86%. This demonstrates the effectiveness of our designed two-phase training strategy.

## 6 Conclusion

This paper presents a preliminary exploration of integrating chain-of-thought into generative retrieval. While existing approaches directly map queries to document identifiers, we demonstrate that interleaving explicit thought processes with retrieval actions can substantially improve performance on complex queries. To realize this idea, we develop ThinkGR as a preliminary instantiation, incorporating a hybrid decoding strategy to unify free-form thought with constrained identifier generation, and a two-phase training approach that leverages retrieval-grounded reinforcement learning to optimize thought quality. Empirical results across four multi-hop benchmarks validate our approach, with ThinkGR achieving an average improvement of +6.86% over state-of-the-art baselines. We hope this preliminary study demonstrates the feasibility and potential of thought-augmented generative retrieval, inspiring further investigation into deliberative generation for information retrieval systems.

## Limitations

As an early exploration into integrating chain-of-thought with generative retrieval, our study has several limitations that suggest directions for future research: (1) Our current implementation relies on established training algorithms (SFT and KTO) to align thought and retrieval. While effective for this preliminary exploration, these methods may not fully unlock the potential of thought-augmented retrieval. Future work could investigate more specialized optimization techniques, such as process reward models that provide fine-grained feedback on thought quality, or curriculum learning strategies that progressively increase thought complexity. (2) Although our method achieves state-of-the-art results on multi-hop retrieval, it is specifically tailored for these scenarios through its triple-based docid design. This specialization may lead to suboptimal performance on standard one-hop benchmarks (e.g., Natural Questions) where intermediate reasoning can introduce unnecessary complexity. Moreover, the current triple representation primarily captures entity-relation structures, potentially overlooking non-factual content. Future work should explore developing more generalizable docid representations (e.g., entity-fact tuples) that can effectively capture both relational semantics for multi-hop traversal and broader content for one-hop matching. (3) To rigorously evaluate our method's effectiveness in complex reasoning scenarios, our experiments primarily compare against approaches specifically designed for multi-hop retrieval, which represent stronger baselines in this setting. While this comparison strategy validates ThinkGR's advantages in its target domain, a comprehensive evaluation of thought-augmented generative retrieval as a general paradigm requires broader analysis. In future work, we plan to conduct more detailed comparisons with existing generative retrieval methods across diverse retrieval tasks to better characterize the benefits and trade-offs of integrating chain-of-thought into the generative retrieval framework.

## Ethics Statement

This work explores the integration of Chain-of-Thought reasoning into generative retrieval models to enhance their retrieval capabilities, particularly for handling complex queries. All experiments derived in this study rely on publicly available datasets and do not involve any collection of personal data or interaction with human subjects.

## References


Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. *Advances in Neural Information Processing Systems*, 35:31668–31683.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Justus-Jonas Erker, Nils Reimers, and Iryna Gurevych. 2025. Grithopper: Decomposition-free multi-hop dense retrieval. *arXiv preprint arXiv:2503.07519*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Paolo Ferragina and Giovanni Manzini. 2000. Opportunistic data structures with applications. In *Proceedings 41st annual symposium on foundations of computer science*, pages 390–398. IEEE.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. Deeprag: Thinking to retrieval step by step for large language models. *arXiv preprint arXiv:2502.01142*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *CoRR*, abs/2405.13576.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Hyunji Lee, Sohee Yang, Hanseok Oh, and Minjoon Seo. 2022. Generative multi-hop retrieval. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1417–1436, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Julian Schnitzler, Xanh Ho, Jiahao Huang, Florian Boudin, Saku Sugawara, and Akiko Aizawa. 2024. Morehopqa: More than multi-hop reasoning. *arXiv preprint arXiv:2406.13397*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

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

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten Rijke, and Zhaochun Ren. 2023. Learning to tokenize for generative retrieval. In *Advances in Neural Information Processing Systems*, volume 36, pages 46345–46361. Curran Associates, Inc.

Jianting Tang, Dongshuai Li, Tao Wen, Fuyu Lv, Dan Ou, and Linli Xu. 2025. Large reasoning embedding models: Towards next-generation dense retrieval paradigm. *arXiv preprint arXiv:2510.14321*.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843.

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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Wenhan Xiong, Xiang Lorraine Li, Srinivasan Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. *International Conference on Learning Representations*.

Ruiran Yan, Zheng Liu, and Defu Lian. 2025. O1 embedder: Let retrievers think before action. *arXiv preprint arXiv:2502.07555*.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Tian Yu, Shaolei Zhang, and Yang Feng. 2024. Auto-rag: Autonomous retrieval-augmented generation for large language models.

Caojin Zhang, Qiang Zhang, Ke Li, Sai Vidyaranya Nuthalapati, Benyu Zhang, Jason Liu, Serena Li, Lizhu Zhang, and Xiangjun Fan. 2025a. Gem: Empowering llm for both embedding generation and language understanding. *arXiv preprint arXiv:2506.04344*.

Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, Fangchao Liu, and Zhao Cao. 2024. Generative retrieval via term set generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 458–468.

Wenhao Zhang, Mengqi Zhang, Shiguang Wu, Jiahuan Pei, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Pengjie Ren. 2025b. Excluir: Exclusionary neural information retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 13295–13303.

Yingchen Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2025c. Does generative retrieval overcome the limitations of dense retrieval? *arXiv preprint arXiv:2509.22116*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An ultimate retriever on corpus with a model-based indexer. *arXiv preprint arXiv:2208.09257*.

## A Prompt Templates

In this section, we present the prompt used to instruct LLM in summarizing documents into multiple knowledge triples (Figure 5), and the prompt used to guide LLM in generating thought-retrieval chains for SFT training data (Figure 6).

## B Detailed Statistics of Datasets

Among the datasets used, HotpotQA and MoreHopQA involve 2-hop questions, MuSiQue covers questions with 2-4 hops, while 2WikiMultiHopQA has up to 6 ground-truth documents for one question. MoreHopQA does not provide training set, thus it is used to evaluate the model's performance in out-of-domain setting. It is worth noting that HotpotQA has been found to have the over-specification issue (Trivedi et al., 2022), where the questions contain too many ground-truth document fragments, leading to high lexical overlap between questions and supporting documents. This allows models to easily identify relevant documents by matching fragments rather than performing multi-step thought, making it difficult to accurately assess their capabilities. We show some examples of HotpotQA in Appendix F to illustrate this issue and further discuss its impact in the experimental results analysis. Nonetheless, we still include it as a reference for evaluation.

To construct corpus, for each dataset, we collected the supporting and non-supporting documents provided for each instance and used them as the corpus. Since the original corpus of MoreHopQA contains only about 1K documents, which deviates from real-world retrieval scenarios, we combine it with the 2WikiMultiHopQA corpus to increase the difficulty and ensure the reliability of the evaluation.

Table 3 provides detailed statistics of the datasets used in our experiments, including corpus size, number of training/test data, and the number of hops covered by the queries.

| Dataset | HotpotQA | 2Wiki. | Musique | Morehopqa |
|---|---|---|---|---|
| # Corpus | 5M | 398K | 118K | 399K |
| # Train | 84,812 | 167,181 | 19,116 | - |
| # Test | 5,447 | 12,576 | 2,417 | 1,118 |
| # Hops | 2 | 2-6 | 2-4 | 2 |

Table 3: Detailed statistics of the datasets used in our experiments, including corpus size, number of training/test data, and the number of hops covered by the queries.

## C Implementation Details

ThinkGR is trained using Llama-3.1-8B-Instruct (Grattafiori et al., 2024) as the base model. We employ the Llama-Factory framework (Zheng et al., 2024) for sft training, using DeepSpeed zero-3 for full fine-tuning with a learning rate of 1e-6, batch size of 512, cutoff length of 2048, and warmup ratio of 0.05. For the reinforcement learning (RL) phase, we train ThinkGR using KTO with a learning rate of 4e-7, batch size of 128, and set the risk aversion hyperparameter $\beta$ to 0.1. We select the HotpotQA and Musique datasets for training in this phase, as the model already achieves high retrieval accuracy on the 2WikiMultihopQA dataset after the first training phase. The threshold $\tau$ for distinguishing desirable and undesirable responses is set to 0.5. This threshold was selected based on pilot experiments to ensure a clear and robust learning signal.

For constructing semantic triples as document identifiers, we use Llama-3.1-8B-Instruct to summarize each document into knowledge triples. For generating thought-retrieval chains as SFT training data, we employ Llama-3.3-70B-Instruct to ensure high-quality thought trajectories.

We reproduced baselines based on official code and FlashRAG (Jin et al., 2024). To ensure fairness and comparability, we constrained all baselines to retrieve no more than 10 documents per query. Specifically, for conventional non-iterative retrievers (incapable of multi-hop reasoning), we set the number of retrieved documents to the number of ground-truth documents plus one. For iterative retrieval methods, we limited the number of hops to 5 and the number of documents retrieved per step to 2. LLM-driven multi-step retrieval methods require an off-the-shelf LLM and retriever. Except for R3-RAG and Auto-RAG, which use their own fine-tuned LLM, we uniformly use Llama-3.3-70B-Instruct as LLM and Contriever-MSMARCO (Izacard et al., 2021) as retriever to ensure fair comparison. All experiments are conducted on NVIDIA A800 GPUs.

## D Influence of Base Models

To assess the robustness of ThinkGR across different parameter scales, we systematically evaluate its performance using various base models. We extend the Llama3.1-8B-Instruct model used in our main experiments to multiple models from the Llama3

Prompt for Knowledge Triples Generation

**Instruction:**
Extract informative triplets directly from the passage following the examples. Do not add any extra words, line breaks, or spaces.
Note that the passage may not only contain information about the title entity, if there is information about other entities, it should also be extracted.

IMPORTANT: Consider the overall context of the passage when extracting triplets AND extract information from ALL parts of sentences, including:
- Main clause components (subject-verb-object)
- Modifiers (adjectives and adverbs)
- Prepositional phrases
- Subordinate clauses
- Attributive phrases and clauses

**Few-Shot Demonstration:**
{Examples}

When extracting triplets, make sure to:
1. Consider the main topic or setting of the passage
2. Connect events, people, and facts to their broader context
3. Extract information from ALL parts of sentences, including modifiers and subordinate clauses
4. Transform information in attributive phrases into separate triplets
5. Look for implicit relationships that may be expressed through modifiers
6. Be comprehensive and extract all relevant information
7. For each entity pair, provide BOTH general relationship types (like "location", "type") AND more specific relationship descriptions (like "harbor", "designated as")

**Input:**
Target Passage:
{passage}

**Output:**
Triplets:

Figure 5: Prompt templates for knowledge triples generation.

Prompt for SFT Training Data Generation

**Instruction:**

You will be provided a multi-hop question and multiple knowledge triples, your task is to select the necessary triples that are needed to answer question and construct a concise reasoning chain. The reasoning chain should demonstrate how to retrieve these knowledge triples to answer the multi-hop question. Follow these guidelines to ensure accurate and coherent output:

Pay attention to the following requirements:
- Use the reasoning chain to answer the multi-hop question. If the question is unanswerable based on the provided triples, add a statement at the end of the reasoning chain: "I cannot answer this question based on the provided knowledge triples."
- Imagine that you are completely unaware of the information in the triples until it is generated in the reasoning chain. Any new entities and information not present in the question must be introduced by generating corresponding triples in the reasoning chain.
- If necessary, you should flip the order of the entities in triples to *ensure the order of entities follows the logical reasoning from known to unknown*, never starting a triple with an entity not mentioned in the question. Note that you only need to directly use the flipped triples to construct the reasoning chain without telling me the flipping process.
- Keep the reasoning chain straightforward, short and concise. No need to confirm details, deduce the answer with as few triples as possible.
- IMPORTANT: *Only select necessary triples* to keep the number of triples in the reasoning chain to a minimum, avoid unnecessary triples.
- Only respond with the best final reasoning chain, do not need any explanation or redundant words.

**Few-Shot Demonstration:**

{Examples}

Respond with the final reasoning chain, do not need any explanation or redundant words.

**Input:**

Question:
{question}
Knowledge Triples:
{triples}

**Output:**

Figure 6: Prompt templates for SFT training data generation.

| Base Model | HotpotQA | 2WikiMultihopQA | Musique | Morehopqa | Average |
|---|---|---|---|---|---|
| *Llama3* | | | | | |
| Llama3.1-8B | 76.09 | 93.19 | 63.98 | 80.50 | 78.44 |
| Llama3.2-3B | 66.55 | 89.66 | 50.70 | 74.73 | 70.41 |
| Llama3.2-1B | 60.94 | 88.43 | 45.88 | 71.47 | 66.68 |
| *Qwen3* | | | | | |
| Qwen3-14B | 70.43 | 91.05 | 53.98 | 76.12 | 72.90 |
| Qwen3-8B | 66.70 | 89.90 | 51.54 | 75.13 | 70.82 |
| Qwen3-4B | 65.90 | 89.01 | 50.50 | 72.81 | 69.56 |
| Qwen3-1.7B | 62.48 | 88.46 | 46.99 | 72.50 | 67.61 |
| Qwen3-0.6B | 60.14 | 87.22 | 45.74 | 71.06 | 66.04 |

Table 4: Experimental results for different base models.

| Method | HotpotQA | 2WikiMultihopQA | Musique | Average |
|---|---|---|---|---|
| *LLM-Driven Multi-Step Retrieval* | | | | |
| Selfask | 37.81 | 33.88 | 28.90 | 33.53 |
| IRCoT | 9.26 | 9.66 | 8.66 | 9.19 |
| ITER-RETGEN | 50.87 | 45.82 | 51.55 | 49.41 |
| Auto-RAG | 4.37 | 6.74 | 5.22 | 5.44 |
| R3-RAG | 0.93 | 1.36 | 1.21 | 1.17 |
| *Reasoning-Augmented Dense Retrieval* | | | | |
| MDR | 0.13 | 0.06 | 0.05 | 0.08 |
| GritHopper | 65.51 | 3.53 | 1.36 | 23.47 |
| *Our Method* | | | | |
| ThinkGR | 1.13 | 0.60 | 1.38 | 1.04 |

Table 5: Experimental results of latency comparison. Lower values indicate superior efficiency.

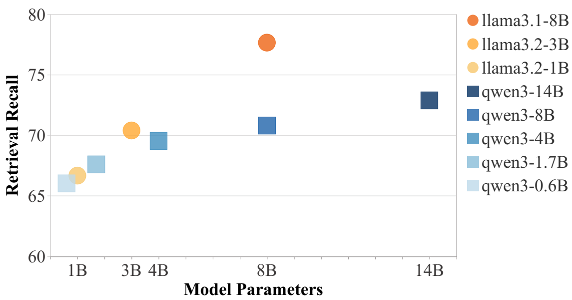


Figure 7: Performance of ThinkGR based on different base models. The x-axis represents the model size, while the y-axis shows the average retrieval recall across four datasets.

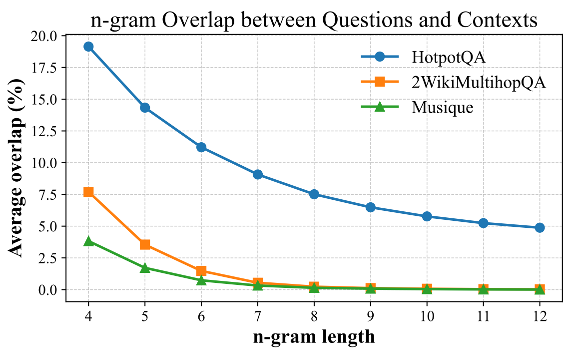


Figure 8: N-gram overlap between questions and ground-truth documents across three datasets. HotpotQA shows significantly higher overlap rates, indicating severe over-specification.

and Qwen3 families, covering parameter scales ranging from 0.6B to 14B. All models are trained under the same experimental settings, following our proposed two-phase training strategy. The results presented in Figure 7 demonstrate two key insights. First, our method maintains satisfactory performance across different base models, even with smaller models like Qwen3-0.6B. This indicates the architectural generality of our framework. Second, under the same model architecture, larger models tend to perform better. This is attributed to their stronger capabilities and richer parameter knowledge, which better support the complex thought-retrieval interactions. These findings sug-

| Question | Documents |
|---|---|
| Who did the professional tennis player currently ranked World No. 1 in men's singles defeat in the 2017 Barcelona Open Banco Sabadell - Singles? | ***Doc1:*** "2017 Barcelona Open Banco Sabadell – Singles: Rafael Nadal was the defending champion and successfully defended his title, defeating Dominic Thiem in the final, 6–4, 6–1."<br>***Doc2:*** "Rafael Nadal is a Spanish professional tennis player, currently ranked World No. 1 in men's singles." |
| When was the former science teacher at Mount Vernon Middle School born for whom the Mount Vernon district received national attention? | ***Doc1:*** "The Mount Vernon district received national attention when the board voted to fire John Freshwater for branding a student with a Christian cross and teaching creationism. [The district oversees] Mount Vernon Middle School..."<br>***Doc2:*** "John Freshwater (born June 22, 1956) is a former science teacher at Mount Vernon Middle School in Mount Vernon, Ohio, who was dismissed by the Board of Education for teaching creationism." |
| What primarily competed with Nintendo 64 and Sega Saturn has a horror-themed adventure game published by Jaleco in 1999? | ***Doc1:*** "Juggernaut is a horror-themed adventure game published by Jaleco in 1999 for the PlayStation."<br>***Doc2:*** "The PlayStation is a home video game console... It primarily competed with the Nintendo 64 and the Sega Saturn as part of the fifth generation of video game consoles." |
| The player voted SEC Player of the Year in 1990 played college football at what university? | ***Doc1:*** "The 1990 All-SEC football team: Florida quarterback Shane Matthews was voted SEC Player of the Year."<br>***Doc2:*** "Shane Matthews (born June 1, 1970) is an American former college and professional football player... He played college football for the University of Florida." |

Figure 9: Examples of over-specification issue in HotpotQA, where questions and supporting documents have high lexical overlap enabling direct matching without deep reasoning.

gest that ThinkGR exhibits broad transferability across various base models. While larger models yield better results, the framework remains practically feasible with smaller models, providing flexibility for deployment under different resource constraints. Table 4 shows the detailed performance of our method based on different base models.

## E Complete Results of Efficiency Comparison

This section provides the complete experimental results of inference latency (seconds per query) across four datasets. As detailed in Table 5, ThinkGR demonstrates significantly reduced latency compared to LLM-Driven Multi-Step Retrieval methods, by eliminating costly sequential LLM/retriever calls. While Reasoning-Augmented Dense Retrieval baselines generally show lower latency (0.08s average for MDR) due to implicit thought, ThinkGR achieves an optimal effectiveness-efficiency trade-off despite explicit thought generation. Notably, GritHopper exhibits anomalously high latency on HotpotQA stems from hardware limitations: its GPU-accelerated FAISS retrieval defaults to CPU execution due to HotpotQA's index being too large for available GPU memory. Crucially, in our method, the hybrid decoding strategy based on FM-index ensures constant time complexity, with latency determined solely by output token length. This architecture fundamentally decouples inference speed from corpus scale, which is a critical advantage for real-world deployments with growing corpora.

## F Analysis of Over-Specification Issue in HotpotQA

To quantitatively analyze the over-specification issue in HotpotQA, we conduct a statistical analysis of the n-gram overlap between questions and ground-truth documents across three datasets: HotpotQA, 2WikiMultihopQA, and Musique. The results are presented in Figure 8. As shown, HotpotQA exhibits significantly higher n-gram overlap rates compared to the other two datasets across all n-gram sizes (from 4-gram to 12-gram). For instance, the average 4-gram overlap of HotpotQA reaches 19.15%, which is 2.5× higher than 2WikiMultihopQA (7.70%) and 5× higher than Musique (3.82%). This gap becomes even more pronounced for larger n-grams: at 8-gram, HotpotQA maintains 7.51% overlap while 2WikiMultihopQA and Musique drop to only 0.23% and 0.14%, respectively. These statistics confirm that HotpotQA suffers from severe over-specification, where questions contain substantial fragments from the ground-truth documents. This allows models to retrieve relevant documents through simple lexical matching rather than performing genuine multi-step thought.

To further illustrate this issue, we present specific examples in Figure 9. These examples demonstrate how questions in HotpotQA often share

| Method | HotpotQA | 2WikiMultihopQA | Musique | Morehopqa |
|---|---|---|---|---|
| *LLM-Driven Multi-Step Retrieval* | | | | |
| Selfask | 62.97 | 32.82 | 32.82 | 40.43 |
| IRCoT | 74.78 | 58.08 | 47.75 | 46.78 |
| ITER-RETGEN | 76.98 | 62.20 | 51.14 | 52.42 |
| Auto-RAG | 69.91 | 54.57 | 38.39 | 41.68 |
| R3-RAG | 77.09 | 70.59 | 47.12 | 51.34 |
| *Reasoning-Augmented Dense Retrieval* | | | | |
| MDR | 87.28 | 43.73 | 27.02 | 39.80 |
| GritHopper | **91.08** | 53.10 | 52.50 | 48.66 |
| *Our Method* | | | | |
| ThinkGR | 79.99 | **79.52** | **57.94** | **53.40** |

Table 6: Experimental results for the complete QA task. We evaluate the QA performance using accuracy (Acc) as the metric, where Llama-3.3-70B-Instruct is used to answer questions based on the retrieved documents and judge the correctness.

| Method | HotpotQA | 2WikiMultihopQA | Musique | Morehopqa |
|---|---|---|---|---|
| R3-RAG (Contriever) | 58.56 | 82.82 | 51.70 | 70.44 |
| R3-RAG (BGE-large) | 60.48 | 83.78 | 52.88 | 77.28 |
| **ThinkGR** | **76.09** | **93.19** | **63.98** | **80.50** |

Table 7: Performance comparison of R3-RAG with different retrievers and ThinkGR.

significant lexical overlap with the supporting documents, allowing retrieval models to succeed through simple pattern matching rather than complex thought.

## G Analysis of Triple Collisions

Since we use knowledge triples as document identifiers, it is possible for the same triple to appear in multiple documents. We allow such collisions and retrieve all documents containing the matched triple. Table 8 presents the proportion of triple overlaps across the four datasets. As shown, the collision rate is very low (less than 3% for triples appearing in $\geq 2$ documents), indicating that triples are generally discriminative enough to identify documents.

| Dataset | Triples in ≥2 Docs | Triples in ≥3 Docs |
|---|---|---|
| HotpotQA | 0.91% | 0.21% |
| 2Wiki. | 1.80% | 0.43% |
| Musique | 2.51% | 0.45% |
| MoreHopQA | 1.79% | 0.43% |

Table 8: Proportion of triple overlaps across datasets.

## H Case Study Analysis

We provide a specific case study in Figure 10 to illustrate the impact of retrieval-grounded thought optimization. ThinkGR (top) successfully generates the correct docids, while the ablation variant without RL (bottom) incorrectly predicts the first docid, leading to subsequent thought errors.

## I Impact of Advanced Retriever on Baselines

We investigate whether using a more advanced retriever can bridge the performance gap between LLM-Driven Multi-Step Retrieval methods and ThinkGR. Specifically, we replace the Contriever in R3-RAG (the strongest LLM-driven baseline) with BGE-large (Xiao et al., 2023) and evaluate its performance. The results are presented in Table 7. As shown, while the advanced retriever improves R3-RAG's performance, particularly on MorehopQA (+6.84%), the gains on other datasets are limited. ThinkGR still significantly outperforms R3-RAG equipped with BGE-large across all datasets. This confirms that the superiority of ThinkGR stems from its unified thought-retrieval framework rather than the choice of the underlying retriever.

| $\tau$ | HotpotQA | 2WikiMultihopQA | Musique | MoreHopQA | Average |
|---|---|---|---|---|---|
| 0.1 | 74.05 | 92.82 | 61.93 | **80.72** | 77.38 |
| 0.3 | 74.67 | 92.61 | 62.56 | 79.70 | 77.39 |
| 0.5 | **76.09** | **93.19** | **63.98** | 80.50 | **78.44** |
| 0.7 | 74.89 | 93.00 | 63.24 | 80.23 | 77.84 |

Table 9: Sensitivity analysis of the threshold $\tau$ for distinguishing desirable and undesirable responses. Results are reported as Recall.

| Dataset | Corpus Size | ThinkGR | GritHopper | Improvement |
|---|---|---|---|---|
| HotpotQA | 5M | 1.94 GB | 80 GB | **41x smaller** |
| 2WikiMultihopQA | 398K | 480 MB | 6.24 GB | **13x smaller** |
| Musique | 118K | 68 MB | 1.85 GB | **27x smaller** |
| MoreHopQA | 399K | 480 MB | 6.25 GB | **13x smaller** |

Table 10: Memory footprint comparison of the index between ThinkGR and GritHopper across different datasets.

The Nicole and Natalie album's band is named after who?

**ThinkGR**

I first need to know the band of the Nicole and Natalie album. So I retrieve <docid start>Nicole and Natalie, artist, Nina Sky<docid end>. Then, I need to find out who Nina Sky is named after. I retrieve <docid start>Nina Sky, composed of, Nicole Albino<docid end> and <docid start>Nina Sky, composed of, Natalie Albino<docid end>.

**ThinkGR w/o RL**

I first need to know the band of the Nicole and Natalie album. So I retrieve <docid start>Nicole and Natalie, released on, July 31, 2012<docid end> and <docid start>July 2010, month of end, Working Lunch<docid end> and <docid start>Working Lunch, type, television programme<docid end>. Then, I need to find out who the band is named after. I retrieve <docid start>Working Lunch, broadcast on, BBC Two<docid end> and <docid start>Working Lunch, aired by, BBC Two<docid end>.

Figure 10: A case study on Musique illustrating the impact of retrieval-grounded thought optimization. ThinkGR (top) successfully generates the correct docids, while the ablation variant without RL (bottom) incorrectly predicts the first docid, leading to subsequent thought errors.

## J Sensitivity Analysis of Threshold $\tau$

The threshold $\tau$ controls the boundary for labeling model-generated responses as undesirable during the retrieval-grounded thought optimization phase. To analyze the sensitivity of this hyperparameter, we conduct experiments with $\tau \in \{0.1, 0.3, 0.5, 0.7\}$ and report the results in Table 9. As shown, ThinkGR exhibits relatively stable performance across different threshold values, with the average recall ranging from 77.38% to 78.44%. This robustness reduces the burden of extensive hyperparameter tuning in practice. The optimal performance is achieved at $\tau = 0.5$, which we adopt in our main experiments. A lower threshold (e.g., $\tau = 0.1$) classifies only completely failed retrievals as undesirable, providing insufficient negative supervision. Conversely, a higher threshold (e.g., $\tau = 0.7$) may incorrectly label partially correct responses as undesirable, introducing noisy signals that hinder learning. The moderate threshold $\tau = 0.5$ strikes a balance by ensuring clear differentiation between successful and failed retrievals, leading to a more robust and effective optimization signal.

## K Index Storage Scalability Analysis

In addition to time efficiency, we also evaluate the index storage scalability of our method. The FM-index used in ThinkGR is highly space-efficient. As shown in Table 10, our index is significantly smaller than GritHopper's dense retrieval index. For instance, on the HotpotQA dataset with 5M documents, ThinkGR requires only 1.94 GB of index storage, which is 41× smaller than GritHopper's 80 GB. This substantial reduction in index size makes ThinkGR highly feasible for large-scale retrieval scenarios involving hundreds of millions or even billions of documents.