KiRAG: Knowledge-Driven Iterative Retriever for Enhancing Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Iterative retrieval-augmented generation (iRAG) models offer an effective approach for multi-hop question answering (QA). However, their *retrieval process* faces two key challenges: (1) it can be disrupted by irrelevant documents or factually inaccurate chain-of-thoughts; (2) their retrievers are not designed to dynamically adapt to the evolving information needs in multi-step reasoning, making it difficult to identify and retrieve the missing information required at each iterative step. Therefore, we propose **KiRAG**[1], which uses a knowledge-driven iterative retriever model to enhance the retrieval process of iRAG. Specifically, KiRAG decomposes documents into knowledge triples and performs iterative retrieval with these triples to enable a factually reliable retrieval process. Moreover, KiRAG integrates reasoning into the retrieval process to dynamically identify and retrieve knowledge that bridges information gaps, effectively adapting to the evolving information needs. Empirical results show that KiRAG significantly outperforms existing iRAG models, with an average improvement of 9.40% in R@3 and 5.14% in F1 on multi-hop QA.

1 Introduction

Retrieval-augmented generation (RAG) models have demonstrated superior performance in question answering (QA) tasks (Lewis et al., 2020; Ram et al., 2023; Lin et al., 2024). While standard RAG models excel at single-hop questions, they often struggle with multi-hop questions (Trivedi et al., 2023), which require reasoning over multiple interconnected pieces of information to derive correct answers. The key limitation is that their single-step retrieval process often fails to retrieve all the relevant information needed to answer multi-hop questions (Shao et al., 2023), leading to knowledge gaps in the reasoning process. To address this limitation, iterative RAG (iRAG) models have been proposed (Trivedi et al., 2023; Asai et al., 2024; Su et al., 2024; Yao et al., 2024). These models employ multiple steps of retrieval and reasoning to iteratively gather the necessary information for addressing multi-hop questions.

Question	According to the 2001 census, what was the population of the city in which Kirton End is located?
KiRAG (Ours)	**Step 1**: Kirton End is a hamlet in the civil parish of Kirton in the Boston district of Lincolnshire, England. ... **Step 2**: Boston is a town and small port in Lincolnshire, on the east coast of England... while the town itself had a population of 35,124 at the 2001 census.
IRCoT	**Step 1**: Kirton End is a hamlet in the civil parish of Kirton in the Boston district of Lincolnshire, England. ... **Step 2**: Kirton is a village in Nottinghamshire, England... According to the United Kingdom Census 2001 it had a population of 273, reducing to 261 at the 2011 census.
IRDoc	**Step 1**: Kirton End is a hamlet in the civil parish of Kirton in the Boston district of Lincolnshire, England. ... **Step 2**: Ollerton is a small town in Nottinghamshire... The population of this parish at the 2011 census was 9,840.

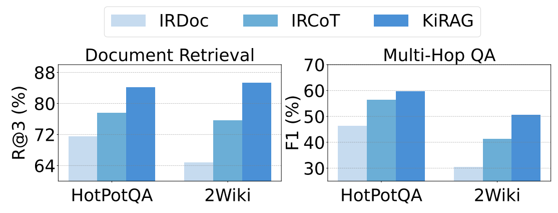


Figure 1: *(top)* Example of top-ranked documents at each step, with relevant content marked in blue and distracting content in orange. We compare KiRAG with IRCoT (Trivedi et al., 2023) and its variant IRDoc, where we replace generated thoughts with top-ranked documents. *(Bottom)* The corresponding retrieval and QA performance on HotPotQA and 2Wiki datasets.

Despite the effectiveness of existing iRAG models, their *retrieval process* faces two key challenges: (1) These models perform iterative retrieval by iteratively augmenting the query with either previously retrieved documents (Zhao et al., 2021) or generated chain-of-thoughts (Trivedi et al., 2023). However, retrieved documents often include noise or irrelevant information (Yoran et al., 2024), while generated chain-of-thoughts can contain factually inaccurate content (Wang et al., 2023; Luo et al., 2024). The propagation of these distracting contexts can degrade retrieval quality and ultimately

[1]Code: `https://anonymous.4open.science/r/kirag`

hinder overall RAG performance. (2) Answering a multi-hop question requires multi-step reasoning, where the information needed to derive the correct answer evolves with each iteration. For example, to answer the question in Figure 1, the first iterative step requires retrieving the location of Kirton End (Boston). Once this information is obtained, the next step shifts to retrieving Boston's population in 2001, demonstrating how the information needed to answer a multi-hop question evolves with each iteration. However, existing iRAG models often rely on off-the-shelf retrieval models that retrieve information based on semantic similarity. These retrievers are not designed to dynamically adapt to the evolving information needs in multi-step reasoning, making it difficult to identify and retrieve the missing pieces of information needed at each iteration, thereby hindering the overall retrieval effectiveness. Figure 1 illustrates these two key challenges, highlighting the necessity of developing a retrieval approach that can mitigate the impact of irrelevant documents or inaccurate thoughts, and dynamically adapt to evolving information needs.

To this end, we propose **KiRAG**, which leverages a **K**nowledge-driven iterative retriever model to enhance the retrieval process of **iRAG** models. Specifically, to address the challenge of irrelevant documents and inaccurate thoughts, inspired by prior works (Fang et al., 2024a,b) that use knowledge triples for enhanced reasoning, KiRAG decomposes documents into knowledge triples, formatted as ⟨*head entity*, *relation*, *tail entity*⟩, and performs iterative retrieval with these triples. By leveraging knowledge triples, which are compact and grounded in documents, KiRAG enables a more focused and factually reliable retrieval process.

Moreover, to address the challenge of evolving information needs, KiRAG employs a knowledge-driven iterative retrieval framework to retrieve relevant knowledge triples from the corpus systematically. This framework *integrates reasoning into retrieval process*, enabling the system to identify and retrieve knowledge that bridges information gaps dynamically. Specifically, the iterative retrieval process incrementally builds a *knowledge triple-based reasoning chain*, such as "⟨*Kirton End; location; Boston*⟩,⟨*Boston*; *population in 2001 census*; *35,124*⟩", by retrieving triples step-by-step. At each iteration, given the current step reasoning chain, e.g., "⟨*Kirton End; location; Boston*⟩", KiRAG dynamically identifies and retrieves the missing knowledge triples needed to coherently extend the chain towards answering the question. This targeted approach can effectively guide the retrieval process in acquiring multiple interconnected pieces of information needed for addressing a question.

We evaluate KiRAG on five multi-hop and one single-hop QA datasets. KiRAG outperforms existing iRAG models, achieving average improvements of 9.40% in R@3 and 7.59% in R@5 on multi-hop QA, which lead to an improvement of 5.14% in F1. Despite that KiRAG is designed for multi-hop QA, it achieves comparable retrieval and QA performance with state-of-the-art baseline on the single-hop QA dataset, demonstrating its effectiveness across different types of questions.

Our contributions can be summarised as follows: (1) We propose KiRAG, which performs iterative retrieval with knowledge triples to enhance the retrieval process of iRAG models; (2) KiRAG uses a knowledge-driven iterative retrieval framework to dynamically adapt the retrieval process to the evolving information needs in multi-step reasoning; (3) Empirical results show that KiRAG achieves superior performance on multi-hop QA.

2 Problem Formulation

Our approach builds on the iRAG process. Given a question q and its answer a, iRAG is formalised as:

$$p_{\theta,\phi}(a|q,\mathcal{C}) \sim p_{\phi}(a|q,\mathcal{D}_q)p_{\theta}(\mathcal{D}_q|q,\mathcal{C}), \quad (1)$$
$$p_{\theta}(\mathcal{D}_q|q,\mathcal{C}) \sim \textstyle\prod_{i=1}^{L} p_{\theta}(\mathcal{D}_q^i|q,\mathcal{D}_q^{<i}), \quad (2)$$

where p_θ denotes the *retriever model* that iteratively retrieves documents $\mathcal{D}_q$=$\{\mathcal{D}_q^i\}_{i=1}^L$ from a corpus $\mathcal{C}$ and p_ϕ is the *reader model*. At the i-th iteration, the retriever model retrieves documents $\mathcal{D}_q^i$ based on question q and previously retrieved documents $\mathcal{D}_q^{<i}$. In this paper, we primarily focus on enhancing the *retriever model*, p_θ, to effectively retrieve relevant documents from the corpus. To evaluate the effectiveness of our approach, we focus on multi-hop QA, a standard type of benchmark for assessing iRAG systems (Gao et al., 2023).

3 KiRAG

This section begins with an overview of KiRAG in §3.1. Next, we present a detailed explanation of each component from §3.2 to §3.3. Finally, the training strategy is introduced in §3.4.

3.1 Overview

Figure 2 provides an overview of our approach. KiRAG uses a *knowledge-driven iterative retrieval*

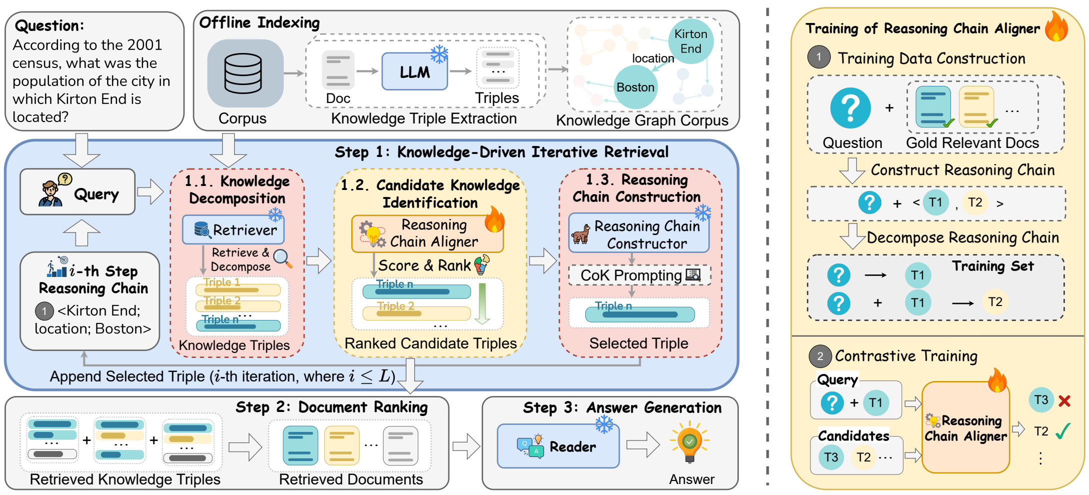

Question:
According to the 2001 census, what was the population of the city in which Kirton End is located?
Offline Indexing
Corpus
Doc
LLM
Triples
Knowledge Triple Extraction
Kirton End
location
Boston
Knowledge Graph Corpus
Step 1: Knowledge-Driven Iterative Retrieval
Query
i-th Step Reasoning Chain
1 <Kirton End; location; Boston>
1.1. Knowledge Decomposition
Retriever
Retrieve & Decompose
Triple 1
Triple 2
Triple n
Knowledge Triples
1.2. Candidate Knowledge Identification
Reasoning Chain Aligner
Score & Rank
Triple n
Triple 2
Ranked Candidate Triples
1.3. Reasoning Chain Construction
Reasoning Chain Constructor
CoK Prompting
Triple n
Selected Triple
Append Selected Triple (i-th iteration, where i ≤ L)
Step 2: Document Ranking
Retrieved Knowledge Triples
Retrieved Documents
Step 3: Answer Generation
Reader
Answer
Training of Reasoning Chain Aligner
1 Training Data Construction
Question
Gold Relevant Docs
Construct Reasoning Chain
< T1 , T2 >
Decompose Reasoning Chain
Training Set
T1
T1
T2
2 Contrastive Training
Query
T1
Candidates
T3
T2
Reasoning Chain Aligner
T3 ✗
T2 ✓

Figure 2: *(left)* Overview of KiRAG. Given a question, it employs a knowledge-driven iterative retrieval process (**Step 1**) to retrieve relevant knowledge triples, including three iterative steps: knowledge decomposition, candidate knowledge identification and reasoning chain construction. The retrieved triples are used to rank documents (**Step 2**), which are passed to the reader for answer generation (**Step 3**). *(right)* Training strategy for the Reasoning Chain Aligner, designed to optimise the identification of relevant knowledge triples at each step of the retrieval process.

framework to systematically retrieve a comprehensive set of relevant knowledge triples $\mathcal{T}_q$ (see §3.2). Next, it leverages the retrieved knowledge triples $\mathcal{T}_q$ to identify and rank documents based on their relevance to the question (see §3.3). Therefore, the retriever model of KiRAG can be formulated as:

$$p_\theta(\mathcal{D}_q|q,\mathcal{C}) \sim p_\theta(\mathcal{D}_q|q,\mathcal{T}_q)\prod_{i=1}^{L} p_\theta(\mathcal{T}_q^i|q,\mathcal{T}_q^{<i},\mathcal{C}), \quad (3)$$

where $\mathcal{T}_q^i$ is the set of knowledge triples retrieved at the i-th iteration, $\mathcal{T}_q^{<i}$ represents all previously retrieved triples and L is the maximum number of iterations. Once we obtain the retrieved documents $\mathcal{D}_q$, KiRAG employs an LLM-based reader model q_ϕ to generate the answer to the question.

3.2 Knowledge-Driven Iterative Retrieval

KiRAG retrieves relevant knowledge triples from the corpus by progressively building a *knowledge triple-based reasoning chain*, i.e., a sequence of logically connected knowledge triples that support answering a given question. For instance, the chain ⟨*Kirton End; location; Boston*⟩,⟨*Boston; population in 2001 census; 35,124*⟩ provides relevant knowledge for answering the question in Figure 2. The reasoning chain is built iteratively by selecting triples step-by-step. At the i-th iteration, given the *i-th step reasoning chain*, which is a sequence of triples obtained up to the i-th iteration, such as ⟨*Kirton End; location; Boston*⟩, the framework retrieves and selects the next triple to extend the reasoning chain through the following three steps:

Knowledge Decomposition. To enable a factually reliable retrieval process, KiRAG decomposes documents into knowledge triples. At the i-th iteration, the query q_i is formed by concatenating the question with the i-th step reasoning chain in the format "*{question}. knowledge triples: {triple1}...*". KiRAG employs an off-the-shelf *Retriever* model to retrieve K_0[2] documents from the corpus, providing an initial pool of information for extracting relevant knowledge (see Step 1.1 in Figure 2).

Building on recent advancements in extracting knowledge triples using LLMs (Edge et al., 2024; Fang et al., 2024b), we employ in-context learning to prompt an LLM to extract knowledge triples for each retrieved document independently. Since the extraction process is query-independent, triples can be precomputed offline for all documents in the corpus[3]. This enables the construction of a *knowledge graph (KG) corpus*, effectively improving retrieval efficiency[4]. The prompt used for extracting knowledge triples is provided in Appendix A.1, where the LLM is instructed to extract all knowledge triples contained within a document in a single pass. We denote the set of knowledge triples extracted from all the retrieved documents at step i as $\tilde{\mathcal{T}}^i$.

Candidate Knowledge Identification. To adapt the retrieval process to evolving information needs, KiRAG retrieves a subset of candidate knowledge triples, i.e., $\mathcal{T}_q^i$, from all the extracted triples that are most likely to address the information gaps in the i-th step reasoning chain. These candidate

[2]We provide analysis of the effect of K_0 in Appendix C.6.
[3]The knowledge triples for retrieved documents can be obtained using the document IDs during the retrieval process.
[4]Efficiency analysis of KiRAG is in Appendix C.8.

triples are selected based on their relevance to the question and their potential to form a coherent reasoning process with the i-th step reasoning chain.

To achieve this, we propose a *Reasoning Chain Aligner*, which is designed to identify candidate triples that advance the reasoning process (see Step 1.2 in Figure 2). We instantiate the Aligner as a bi-encoder model. At the i-th iteration, the Aligner encodes the query q_i, comprising the question and the i-th step reasoning chain, and each triple t in $\widetilde{\mathcal{T}}^i$ independently into a shared space. The score of each triple for addressing the information gaps in the i-th step reasoning chain is computed by taking the inner-product of the query and triple embeddings: $s_\theta(q_i, t) = f_\theta(q_i)^\top f_\theta(t),\ \forall t \in \widetilde{\mathcal{T}}^i$, where $f_\theta(\cdot)$ denotes the embedding function parameterised by θ. The top-N[5] triples with the highest scores are selected as candidate triples to extend the i-th step reasoning chain, i.e., $\mathcal{T}_q^i$. The reasoning chain Aligner is trained to retrieve triples that contribute to building a coherent reasoning chain. Details of the training process are provided in §3.4.

Reasoning Chain Construction. Given the candidate triples $\mathcal{T}_q^i$ from the Aligner at the i-th iteration, KiRAG employs an LLM-based *Reasoning Chain Constructor* to select a single triple from the candidates to extend the i-th step reasoning chain (see Step 1.3 in Figure 2). Our approach is inspired by IRCoT (Trivedi et al., 2023), which iteratively generates individual sentences in a chain-of-thought (CoT). However, instead of relying on potentially inaccurate CoTs, we instruct the LLM to generate a chain-of-knowledge (CoK) (Wang et al., 2024a), where free-form thoughts are replaced with document-grounded knowledge triples to ensure factual reliability.

The prompt used by the Constructor is provided in Appendix A.2. The inputs include the question, the i-step reasoning chain and candidate triples $\mathcal{T}_q^i$. The Constructor selects triples from $\mathcal{T}_q^i$ to complete the i-th step reasoning chain. The first triple in the generated result is appended to the i-th step reasoning chain, forming a new chain that serves as input for subsequent iterations. Note that the Constructor aims to complete the whole chain, but we only take the first triple. Asking the Constructor to complete the whole chain reduces hallucination, and avoids a sub-optimal greedy approach.

The iterative process terminates when the Constructor generates a reasoning chain containing "*the answer is*" or reaches the maximum number of iterative steps L. The candidate knowledge triples collected during the iterative process, i.e., $\mathcal{T}_q = \{\mathcal{T}_q^i\}_{i=1}^L$, along with their associated scores, are output for document retrieval and ranking.

[5]We provide analysis of the effect of N in Appendix C.7.

3.3 Document Ranking

Since the retrieved knowledge triples $\mathcal{T}_q$ may lack certain contextual information, we use these triples to identify and rank their source documents, i.e., $p(\mathcal{D}_q|q, \mathcal{T}_q)$ in Eq. 3, to provide a more comprehensive and precise context. Specifically, the retrieved documents $\mathcal{D}_q$ are collected by aggregating all the documents from which the triples in $\mathcal{T}_q$ are derived. To rank these documents, we assign each document the score of its associated triple(s) $s_\theta(q_i, t)$ from the iterative process. For a document associated with multiple triples, its score is determined by taking the highest one. These documents are ranked in descending order of their scores, with top-K documents returned as the final retrieval results.

Given the question q and the ranked documents $\mathcal{D}_q$, KiRAG leverages an LLM-based reader model to directly generate the answer. The prompt used for answer generation is provided in Appendix A.3, which instructs the model to leverage the context provided by the documents to answer the question.

3.4 Training Strategy

In KiRAG, the Reasoning Chain Aligner is the key component that requires training to effectively identify candidate triples for extending reasoning chain, while the other components, i.e., Retriever and Constructor, remain frozen. This section outlines the training strategy for the Aligner. Due to the lack of existing datasets specifically designed for this task, we construct a silver training dataset by adapting data from existing multi-hop QA datasets. Specifically, given a question and its ground-truth relevant documents, we construct a knowledge triple-based reasoning chain that supports answering the question. The reasoning chain and the question will serve as the labeled data for training the Aligner.

To train the Aligner, we decompose the complete reasoning chain into multiple incomplete reasoning chains and the corresponding next triples (see the right part of Figure 2). For each incomplete reasoning chain, the correct next triple is treated as the positive sample, while the other triples from the candidate set $\widetilde{\mathcal{T}}^i$ are treated as negative samples. The aligner is trained with contrastive learning loss:

$$\mathcal{L} = -\sum_{(q,r,t^+)\in\mathcal{P}} \log \frac{g_\theta(q_r, t^+)}{g_\theta(q_r, t^+) + \sum\limits_{t^- \in \tilde{\mathcal{T}}^{|r|}} g_\theta(q_r, t^-)}, \quad (4)$$

where $\mathcal{P}$ is the training set. Each data-point includes a question q, an incomplete reasoning chain r and a positive triple t^+. The query q_r is the concatenation of q and r, and the function $g_\theta(q_r, t) = \exp(s_\theta(q_r, t))/\tau$ computes the logits, with τ being the temperature. Further details on the training data and training process are provided in Appendix B.3.

4 Experiments

4.1 Experimental Setup

Datasets. We conduct experiments on five multi-hop QA datasets: **HotPotQA** (Yang et al., 2018), **2WikiMultiHopQA (2Wiki)** (Ho et al., 2020), **MuSiQue** (Trivedi et al., 2022), **Bamboogle** (Press et al., 2023) and **WebQuestions (WebQA)** (Berant et al., 2013). We also use a single-hop QA dataset: **Natural Questions (NQ)** (Kwiatkowski et al., 2019). We report the performance on the *full* test sets of these datasets. For datasets with non-pulic test sets (HotPotQA, 2Wiki and MuSiQue), we use their development sets as test sets and report corresponding results. Detailed statistics and corpus information are provided in Appendix B.1.

Baselines. Since KiRAG aims to improve the retrieval performance of iRAG models, we primarily compare it with iRAG models. We compare KiRAG with models from the following categories: (1) Standard RAG model; (2) iRAG models, such as IRCoT (Trivedi et al., 2023), FLARE (Jiang et al., 2023b), and DRAGIN (Su et al., 2024); (3) Enhanced retrieval models, which improve the retrieval performance by using feedback from earlier retrieval steps, such as BeamDR (Zhao et al., 2021) and Vector-PRF (Li et al., 2023). Moreover, to evaluate the effectiveness of using knowledge triples for iterative retrieval, we introduce two variants: *KiRAG-Doc* and *KiRAG-Sent*, where the triples are replaced with documents and sentences, respectively. Both variants follow the same procedure as KiRAG to retrieve documents. More details about the baselines can be found in Appendix B.2.

Evaluation. To evaluate the retrieval performance, we follow previous works (Trivedi et al., 2023; Gutiérrez et al., 2024) and use **R@{3, 5}** as the metrics. To evaluate the QA performance, we use **Exact Match (EM)** and **F1** as evaluation metrics, which are the standard metrics for these datasets.

Model	HotPotQA		2Wiki		MuSiQue	
	R@3	R@5	R@3	R@5	R@3	R@5
RAG	65.47	70.78	60.87	65.20	41.29	46.53
Vector-PRF	65.37	70.06	60.60	64.85	40.93	45.46
BeamDR	67.07	71.89	36.07	42.08	24.17	28.18
FLARE	54.79	59.72	60.84	70.04	39.79	45.81
DRAGIN	69.95	75.85	61.30	70.43	<u>48.67</u>	<u>54.67</u>
IRCoT	<u>71.44</u>	<u>77.57</u>	<u>64.30</u>	<u>75.56</u>	45.61	52.21
KiRAG-Doc	67.80	72.20	45.85	63.07	25.86	39.49
KiRAG-Sent	54.43	69.26	43.53	59.33	31.08	43.69
KiRAG	**80.32**†	**84.08**†	**77.76**†	**85.32**†	**54.53**†	**61.16**†

Table 1: Retrieval performance (%) on multi-hop QA datasets, with the best and second-best results marked in bold and underlined, respectively, and † denotes p-value<0.05 compared with best-performing baseline.

Model	HotPotQA		2Wiki		MuSiQue	
	EM	F1	EM	F1	EM	F1
RAG	34.54	47.35	14.78	30.48	9.10	16.98
Vector-PRF	34.40	47.31	14.96	30.37	9.23	16.98
BeamDR	38.34	51.64	14.42	27.25	7.08	14.42
FLARE	35.58	47.74	<u>26.36</u>	<u>41.82</u>	13.07	21.94
DRAGIN	41.74	55.69	25.58	40.83	<u>16.87</u>	<u>26.71</u>
IRCoT	<u>42.38</u>	<u>56.38</u>	25.12	41.36	15.76	24.94
KiRAG-Doc	33.87	46.43	14.37	27.54	7.49	15.34
KiRAG-Sent	34.14	46.63	14.22	27.50	10.51	18.21
KiRAG	**45.09**†	**59.76**†	**30.72**†	**50.57**†	**19.16**†	**30.00**†

Table 2: QA performance (%) on multi-hop QA datasets, with the best and second-best results marked in bold and underlined, respectively. † denotes p-value<0.05 compared with best-performing baseline.

Training and Implementation Details. To train the Aligner, we use TRACE (Fang et al., 2024b), which constructs knowledge triple-based reasoning chains from a fixed set of documents, to generate ground-truth reasoning chains. The reasoning chain that leads to the correct answer is used for training. Training data is generated from the training sets of three multi-hop QA datasets: HotPotQA, 2Wiki and MuSiQue. The combined data is used to train the Aligner. The Aligner is initialised with E5 (Wang et al., 2022) and finetuned with the constructed training data.

KiRAG uses Llama3 (Dubey et al., 2024) to extract triples and serve as the Constructor to build reasoning chains. It uses frozen E5 or BGE (Xiao et al., 2024) as the Retriever. We use different readers, including Llama3, Qwen2.5 (Yang et al., 2024), Flan-T5 (Chung et al., 2024) and TRACE (Fang et al., 2024b) to generate answers. We mainly report results using E5 as the retriever and Llama3 as the reader, with additional results from other retrievers and readers provided in Appendix C.1. For fair comparison, RAG baselines employ the same retriever and reader as KiRAG. More training and implementation details are in Appendix B.3.

Model	Bamboogle		WebQA		NQ	
	R@3	R@5	R@3	R@5	R@3	R@5
RAG	20.80	25.60	64.91	70.32	<u>73.07</u>	<u>78.56</u>
Vector-PRF	20.60	24.80	64.86	69.54	72.82	78.03
BeamDR	12.00	15.20	41.63	50.25	33.88	42.16
FLARE	32.80	37.60	55.91	60.97	68.98	73.43
DRAGIN	<u>36.80</u>	<u>40.40</u>	65.11	70.03	68.98	73.43
IRCoT	28.00	32.80	<u>65.50</u>	<u>70.42</u>	**73.38**	**78.59**
KiRAG-Doc	20.80	27.20	62.40	68.60	68.59	74.99
KiRAG-Sent	26.40	32.00	62.16	68.06	67.48	74.13
KiRAG	**45.60**†	**49.60**†	**69.05**†	**73.08**†	72.11	77.28

Table 3: Retrieval performance (%) on unseen multi-hop and single-hop QA datasets, where † denotes p-value<0.05 compared with best-performing baselines.

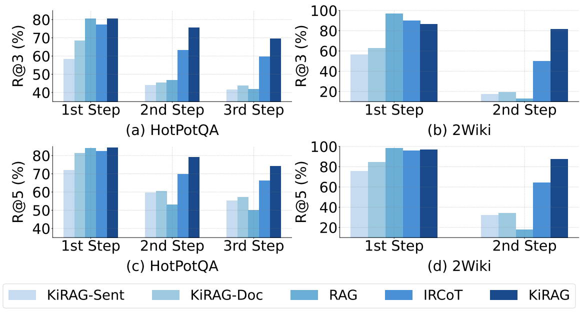


Figure 3: Retrieval performance (%) for relevant documents required across different steps, where most questions in 2Wiki have only two relevant documents.

4.2 Results and Analysis

We present our primary results in this section. Additional results are provided in Appendix C.

(RQ1): How does KiRAG perform in multi-hop QA compared with baselines? The retrieval and QA[6] results are provided in Table 1 and Table 2, respectively, which yield the following findings:
(1) KiRAG consistently outperforms all baselines in retrieval performance on all datasets. Compared to the strongest baselines, KiRAG achieves statistically significant average improvements of 9.40% in R@3 and 7.59% in R@5, demonstrating its superior ability to enhance retrieval performance.
(2) KiRAG consistently achieves the best QA performance on all datasets. It significantly outperforms best-performing baselines, with average improvements of 3.12% in EM and 5.14% in F1. The results validate the effectiveness of KiRAG in facilitating multi-hop QA through high-quality retrieval.
(3) Compared to KiRAG-Doc and KiRAG-Sent, which perform iterative retrieval at document and sentence levels, KiRAG achieves substantially higher retrieval performance. The suboptimal performance of these variants stems from the iterative retrieval process being misled by noise in documents and sentences. In contrast, KiRAG uses finer-grained knowledge triples, reducing the impact of noise and improving retrieval recall.
(4) Compared to IRCoT, which uses CoT for iterative retrieval, KiRAG achieves superior retrieval results, with an average improvement of 10.42% in R@3. This improvement stems from LLM's tendency to generate hallucinated CoT. By using document-grounded knowledge triples, KiRAG ensures a more reliable and faithful retrieval process.

[6]QA performance is based on the top-3 retrieved documents. The results for the top-5 retrieved documents are provided in Appendix C.2, which demonstrate similar results.

(RQ2): Why does KiRAG improve retrieval performance for multi-hop questions? To explain why KiRAG achieves superior retrieval performance, we analyze its ability to retrieve relevant documents required at different steps of the reasoning process. For a multi-hop question, there are multiple logically ordered relevant documents. For instance, the first relevant document for the question in Figure 2 is about "Kirton End", while the second relevant document relates to "Boston". At step i, we only consider the document required at that specific step as relevant and compute its recall. This approach allows us to assess how well KiRAG retrieves the necessary information at each step.

The results on HotPotQA and 2Wiki are shown in Figure 3, yielding the following findings: (1) The recall of both KiRAG and baselines declines with increasing steps, highlighting the growing challenge of retrieving relevant documents for later steps in the reasoning process; (2) Compared to RAG and IRCoT, KiRAG shows comparable, and occasionally slightly lower, retrieval recall at the first step. However, it achieves substantially higher retrieval recall in subsequent steps, which contributes to its overall retrieval effectiveness. This improvement stems from KiRAG's iterative retrieval process, which dynamically adapts to evolving information needs, enabling the effective retrieval of relevant documents required at each step.

(RQ3): Can KiRAG effectively generalise to unseen multi-hop and single-hop QA datasets? To evaluate the generalisation ability of KiRAG, we conduct additional experiments on two multi-hop QA datasets, Bamboogle and WebQA, as well as a single-hop QA dataset, NQ, none of which were included during training. The retrieval results[7] are presented in Table 3, which shows that KiRAG sig-

[7]The QA performance, presented in Table 13 of the Appendix, shows consistent results with retrieval performance.

Model	Retriever	Aligner	Constructor	HotPotQA			2Wiki			MuSiQue		
				R@3	R@5	F1	R@3	R@5	F1	R@3	R@5	F1
KiRAG	✓	✓(trained)	✓	**80.32**	**84.08**	**59.76**	**77.76**	**85.32**	**50.57**	**54.53**	**61.16**	**30.00**
KiRAG w/o Retriever	✗	✓(trained)	✓	74.09†	78.03†	57.07†	77.29	83.65†	48.76†	53.93	60.68	27.14†
KiRAG w/o Aligner	✓	✗	✓	73.34†	75.79†	53.47†	67.66†	70.73†	39.06†	45.60†	49.62†	21.80†
KiRAG w/o Constructor	✓	✓(trained)	✗	74.96†	79.51†	55.67†	72.64†	80.89†	45.69†	46.98†	55.12†	23.71†
KiRAG w/o Training	✓	✓(w/o training)	✓	76.35†	81.56†	59.03†	75.37†	82.50†	48.33†	51.33†	58.66†	28.08†

Table 4: Ablation studies of KiRAG, where † indicates p-value < 0.05 compared with KiRAG.

Model	Retrieval		QA	
	R@3	R@5	EM	F1
E5	29.50	43.25	23.00	31.56
KiRAG w/o Constructor	76.50†	79.25†	31.00†	48.22†
KiRAG	**84.25**†	**86.50**†	**39.00**†	**53.63**†

Table 5: Performance in retrieving relevant knowledge triples for the 100 manually labeled questions on 2Wiki, where † denotes p-value$<$0.05 compared with E5.

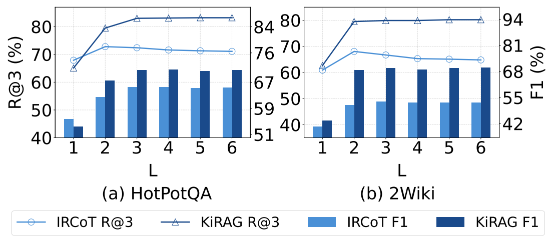


Figure 4: The effect of the number of iterative steps L.

nificantly outperforms all baselines on two multi-hop QA datasets, and demonstrates comparable retrieval performance to IRCoT, the best-performing baseline, on the single-hop QA dataset NQ. These findings highlight the strong generalisation ability of KiRAG in handling diverse QA tasks.

(RQ4): What are the effects of each component and the training strategy in KiRAG? To evaluate the impact of the Retriever, we introduce *KiRAG w/o Retriever*, where the retriever is removed and candidate triples are directly retrieved from the knowledge graph corpus using the Reasoning Chain Aligner. Table 4 shows that removing the Retriever leads to a significant performance drop on HotPotQA while maintaining comparable performance on 2Wiki and MuSiQue. This demonstrates the Aligner's effectiveness in identifying relevant knowledge triples but highlights the limitations of relying solely on the knowledge graph, which may loss contextual information present in documents.

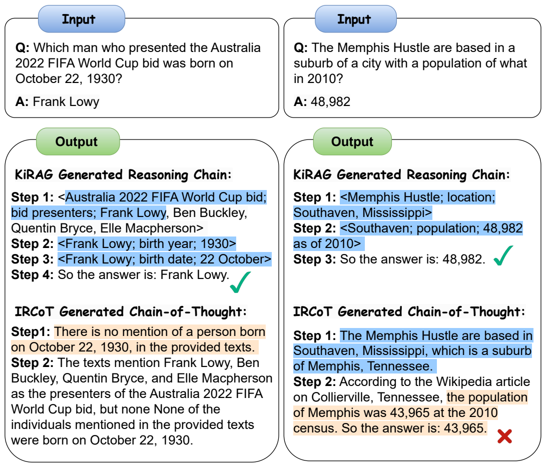
To assess the impact of the Aligner, we introduce *KiRAG w/o Aligner*, where the Aligner is removed and all knowledge triples from the retrieved documents are passed to the Reasoning Chain Constructor. Table 4 shows that *KiRAG w/o Aligner* suffers an average decrease of 8.67% in R@3 and 11.47% in R@5 compared to KiRAG. This decline is due to the absence of filtering or ranking by the Aligner, resulting in noisy and irrelevant triples that hinder the Reasoning Chain Constructor's ability to build coherent reasoning chains, which is essential for guiding the iterative retrieval process effectively.

Moreover, to evaluate the impact of the Constructor, we introduce *KiRAG w/o Constructor*, which constructs reasoning chain using only the top-ranked triple identified by the Aligner. Table 4 indicates that removing the Constructor leads to significantly inferior performance, highlighting the importance of the LLM-based Constructor in building coherent reasoning chains through its advanced reasoning and contextual understanding capability.

To assess the impact of training the Aligner for retrieving and integrating triples, we introduce *KiRAG w/o Training*, where the Aligner is replaced with a frozen E5, which is trained for general text retrieval. Table 4 shows that *KiRAG w/o Training* exhibits a significant decline in both retrieval and QA results. These results highlight the effectiveness of our training strategy in enabling the Aligner to identify relevant knowledge triples.

(RQ5): Can KiRAG retrieve relevant knowledge triples to address multi-hop questions? To evaluate the quality of knowledge triples retrieved by KiRAG, we randomly select 100 questions from the 2Wiki test set and manually identify knowledge triples that are useful in answering these questions. These manually selected triples are considered relevant[8]. We use R@K to measure retrieval performance and compute QA metrics (EM and F1) using the retrieved triples as context. We compare KiRAG with E5, which directly retrieves knowledge triples from the knowledge graph corpus, and the Reasoning Chain Aligner, which iteratively retrieves triples using the trained Aligner. Table 5 shows that KiRAG significantly outperforms E5 in both retrieval and QA performance, demonstrating its effectiveness in retrieving relevant knowledge triples. This superior performance is attributed to the Aligner and the Constructor, which are specifically designed to identify relevant triples.

[8]Appendix C.5 provides details and examples of the manually curated data, which will be released alongside the code.


Input

Q: Which man who presented the Australia 2022 FIFA World Cup bid was born on October 22, 1930?

A: Frank Lowy

Output

KiRAG Generated Reasoning Chain:

Step 1: <Australia 2022 FIFA World Cup bid; bid presenters; Frank Lowy, Ben Buckley, Quentin Bryce, Elle Macpherson>
Step 2: <Frank Lowy; birth year; 1930>
Step 3: <Frank Lowy; birth date; 22 October>
Step 4: So the answer is: Frank Lowy. ✓

IRCoT Generated Chain-of-Thought:

Step1: There is no mention of a person born on October 22, 1930, in the provided texts.
Step 2: The texts mention Frank Lowy, Ben Buckley, Quentin Bryce, and Elle Macpherson as the presenters of the Australia 2022 FIFA World Cup bid, but none None of the individuals mentioned in the provided texts were born on October 22, 1930.

Input

Q: The Memphis Hustle are based in a suburb of a city with a population of what in 2010?

A: 48,982

Output

KiRAG Generated Reasoning Chain:

Step 1: <Memphis Hustle; location; Southaven, Mississippi>
Step 2: <Southaven; population; 48,982 as of 2010>
Step 3: So the answer is: 48,982. ✓

IRCoT Generated Chain-of-Thought:

Step 1: The Memphis Hustle are based in Southaven, Mississippi, which is a suburb of Memphis, Tennessee.
Step 2: According to the Wikipedia article on Collierville, Tennessee, the population of Memphis was 43,965 at the 2010 census. So the answer is: 43,965. ✗

Figure 5: Case study of KiRAG and IRCoT on HotPotQA test set, where the relevant and irrelevant context are marked in blue and orange, respectively.

(RQ6): How does the number of iterative steps L affect the performance of KiRAG? Table 4 shows the retrieval and QA results of KiRAG with different values of L on HotPotQA and 2Wiki development sets. The results show that as the value of L increases, both retrieval and QA performance initially improve and then reach a plateau, with KiRAG achieving optimal performance at a moderate value of L. This highlights the importance of selecting a proper value of L in KiRAG to balance high accuracy and efficiency in multi-hop QA.

Case Study. We conduct a case study to examine the reasoning chains generated by KiRAG. Figure 5 shows examples of the reasoning chains produced by KiRAG and the CoTs generated by IRCoT. The examples show that KiRAG can generate coherent and contextually relevant reasoning chains for answering multi-hop questions, which are essential for effectively guiding the iterative retrieval process. In contrast, IRCoT may struggle with missing information or hallucinations, hindering its ability to retrieve the necessary knowledge.

5 Related Work

RAG Models. RAG models have shown superior performance in QA tasks (Lewis et al., 2020; Izacard and Grave, 2021b; Ram et al., 2023). These models typically employ the *retriever-reader* architecture, which consists of a retriever (Karpukhin et al., 2020; Wang et al., 2022; Fang et al., 2023) and a reader (Izacard and Grave, 2021b; Jiang et al., 2023b). Efforts to improves RAG models generally follows three main directions: (1) enhance the retriever for better retrieval performance (Izacard and Grave, 2021a; Shi et al., 2023; Wang et al., 2024b); (2) enhance the reader for better comprehension and answer generation (Lin et al., 2024; Xu et al., 2024; Wang et al., 2024c); (3) introduce additional modules to bridge the retriever and the reader (Yu et al., 2023; Xu et al., 2023; Ye et al., 2024).

Iterative RAG Models for Multi-Hop QA. Iterative RAG models (Trivedi et al., 2023; Shao et al., 2023; Asai et al., 2024; Liu et al., 2024; Yao et al., 2024) address multi-hop QA by performing multiple steps of retrieval and reasoning. For instance, IRCoT (Trivedi et al., 2023) use LLM-generated chain-of-thoughts for retrieval, while DRAGIN (Su et al., 2024) dynamically decides when and what to retrieve based on the LLM's information needs. However, these models all rely on LLM-generated thoughts, making them prone to hallucination. In contrast, KiRAG employs knowledge triples and a trained retriever to actively identify and retrieve missing information, enabling a more reliable and accurate retrieval for multi-hop QA.

KG-Enhanced RAG Models. Recently, KGs have been integrated into RAG models (Peng et al., 2024). Some studies leverage information from existing KGs (Vrandečić and Krötzsch, 2014) for additional context (Yu et al., 2022; Sun et al., 2024), while others generate KGs from documents to improve knowledge organisation (Edge et al., 2024; Gutiérrez et al., 2024; Chen et al., 2024) or enhance reader comprehension (Li and Du, 2023; Fang et al., 2024a,b; Panda et al., 2024). These models primarily follow the standard RAG pipeline, whereas our work focuses on the iRAG pipelines. Moreover, while they rely on single-step retrieval with pre-existing retrievers, KiRAG employs a trained retriever tailored for iterative retrieval, allowing it to dynamically adapt to the evolving information needs in multi-step reasoning.

6 Conclusion

This paper proposes KiRAG to enhance retrieval process of iRAG models. KiRAG decomposes documents into knowledge triples and employs a knowledge-driven iterative retrieval framework to systematically retrieve relevant knowledge triples. The retrieved triples are used to rank documents, which serve as inputs for answer generation. Empirical results show that KiRAG achieves significant retrieval and QA improvements, with an average increase of 9.40% in R@3 and 5.14% in F1, highlighting its effectiveness in multi-hop QA.

Limitations

We identify the following limitations of our work: (1) The Aligner model is trained using silver data constructed from only three multi-hop QA datasets. While our results demonstrate its effectiveness, we leave the exploration of methods to construct larger-scale and higher-quality training data for future work; (2) In KiRAG, we train only the Aligner model and keep the Constructor model frozen. While further training the Constructor could potential improve performance, we choose to keep it frozen to maintain our framework' adaptability to different LLMs, rather than relying on a specific fine-tuned LLM. Appendix C.9 provides a detailed analysis of the performance using different LLM-based Constructor within our framework.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Weijie Chen, Ting Bai, Jinbo Su, Jian Luan, Wei Liu, and Chuan Shi. 2024. KG-Retriever: Efficient knowledge indexing for retrieval-augmented large language models. *arXiv preprint arXiv:2412.05547*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25:70:1–70:53.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2023. KGPR: Knowledge graph enhanced passage ranking. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3880–3885.

Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2024a. REANO: Optimising retrieval-augmented reader models through knowledge graph generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2094–2112.

Jinyuan Fang, Zaiqiao Meng, and Craig MacDonald. 2024b. TRACE the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 8472–8494.

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

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. *arXiv preprint arXiv:2405.14831*.

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.

Gautier Izacard and Edouard Grave. 2021a. Distilling knowledge from reader to retriever for question answering. In *International Conference on Learning Representations*.

Gautier Izacard and Edouard Grave. 2021b. 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*, pages 874–880.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. FlashRAG: A modular toolkit for efficient retrieval-augmented generation research. *arXiv preprint arXiv:2405.13576*.

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. 2023. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. *ACM Transation on Information System*, 41(3):62:1–62:40.

Ruosen Li and Xinya Du. 2023. Leveraging structured information for explainable multi-hop question answering and reasoning. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 6779–6789.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, and 1 others. 2024. RA-DIT: Retrieval-augmented dual instruction tuning. In *International Conference on Learning Representations*.

Yanming Liu, Xinyue Peng, Xuhong Zhang, Weihao Liu, Jianwei Yin, Jiannan Cao, and Tianyu Du. 2024. RA-ISF: learning to answer and understand from retrieval augmentation via iterative self-feedback. In *Findings of the Association for Computational Linguistics*, pages 4730–4749. Association for Computational Linguistics.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Pranoy Panda, Ankush Agarwal, Chaitanya Devaguptapu, Manohar Kaul, and Prathosh A P. 2024. HOLMES: hyper-relational knowledge graphs for multi-hop question answering using LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13263–13282.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

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

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

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*, pages 9248–9274.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. REPLUG: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12991–13013.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-Graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 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.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, and 1 others. 2023. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2024a. Boosting language models reasoning with chain-of-knowledge prompting. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4958–4981.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Shuting Wang, Xin Yu, Mang Wang, Weipeng Chen, Yutao Zhu, and Zhicheng Dou. 2024b. RichRAG: Crafting rich responses for multi-faceted queries in retrieval-augmented generation. *arXiv preprint arXiv:2406.12566*.

Yuhao Wang, Ruiyang Ren, Junyi Li, Xin Zhao, Jing Liu, and Ji-Rong Wen. 2024c. REAR: A relevance-aware retrieval-augmented framework for open-domain question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5613–5626.

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

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. RECOMP: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

Shicheng Xu, Liang Pang, Mo Yu, Fandong Meng, Huawei Shen, Xueqi Cheng, and Jie Zhou. 2024. Unsupervised information refinement training of large language models for retrieval-augmented generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 133–145.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Weichuan Liu, Lei Hou, and Juanzi Li. 2024. SEAKR: Self-aware knowledge retrieval for adaptive retrieval augmented generation. *arXiv preprint arXiv:2406.19215*.

Fuda Ye, Shuangyin Li, Yongqi Zhang, and Lei Chen. 2024. R^{2}AG: Incorporating retrieval information into retrieval augmented generation. *arXiv preprint arXiv:2406.13249*.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making retrieval-augmented language models robust to irrelevant context. In *International Conference on Learning Representations*.

Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2022. KG-FiD: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 4961–4974.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations*.

Chen Zhao, Chenyan Xiong, Jordan L. Boyd-Graber, and Hal Daumé III. 2021. Multi-step reasoning over unstructured text with beam dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4635–4641.

A Prompts

A.1 Prompt for Knowledge Triple Extraction

The prompt used for extracting knowledge triples from a document is illustrated in Figure 6.

A.2 Prompt for Reasoning Chain Construction

The prompt used by the reasoning chain constructor to build reasoning chains is illustrated in Figure 7, where we instruct the Reasoning Chain Constructor to complete the i-th step reasoning chain with the provided candidate knowledge triples.

A.3 Prompt for Answer Generation

The prompt used by the reader to generate answers is illustrated in Figure 8.

Prompt Used for Knowledge Triple Extraction

Instruction: You are a knowledge graph constructor tasked with extracting knowledge triples in the form of <head entity; relation; tail entity> from a document. Each triple denotes a specific relationship between entities or an event. The head entity and tail entity can be the provided title or phrases in the text. If multiple tail entities share the same relation with a head entity, aggregate these tail entities using commas. Format your output in the form of <head entity; relation; tail entity>.

Examples:
Title: *Dana Blankstein*
Text: *Dana Blankstein- Cohen(born March 3, 1981) is the director of the Israeli Academy of Film and Television. She is a film director, and an Israeli culture entrepreneur.*
Knowledge Triples: *<Dana Blankstein; full name; Dana Blankstein-Cohen>, <Dana Blankstein; birth date; March 3, 1981>, <Dana Blankstein; nationality; Israeli>, <Dana Blankstein; position; director of the Israeli Academy of Film and Television>, <Dana Blankstein; profession; film director, culture entrepreneur>*

Inputs:
Title: *{document title}*
Text: *{document text}*
Knowledge Triples:

Figure 6: Prompt used for extracting knowledge triples.

Prompt Used by Reasoning Chain Constructor

Instruction: Follow the examples to answer the input question by reasoning step-by-step. Output both reasoning steps and the answer.
Examples:
Question: *Consider the racer for whom the bend at the 26th Milestone, Isle of Man is dedicated. When were they born?*
Thought: *<26th Milestone, Isle of Man; named after; Joey Dunlop>,<Joey Dunlop; date of birth; 25 February 1952>. So the answer is 25 February 1952.....*
Inputs:
Context: {*candidate triples*}
Question: {*question*}
Thought: {*i-th step reasoning chain*}

Figure 7: Prompt used by Reasoning Chain Constructor.

Prompt Used for Answer Generation

Instruction: Given some context and a question, please only output the answer to the question.
Inputs:
Context: {*retrieved documents*}
Question: {*question*}
Answer:

Figure 8: Prompt used by the Reader.

B Experimental Details

B.1 Datasets

In our experiments, we employ five multi-hop QA datasets: HotPotQA, 2WikiMultiHopQA (2Wiki), MuSiQue, Bamboogle as well as WebQuestions (WebQA), and one single-hop QA dataset: Natural Questions (NQ). For HotpotQA, we use the corpus provided by its authors for retrieval. For 2WikiMultihopQA and MuSiQue, we construct the retrieval corpus following the exact same procedure outlined by Trivedi et al. (2023). For all other datasets, we leverage the Wikipedia corpus introduced by Karpukhin et al. (2020).

For datasets with public test sets (Bamboogle, WebQA and NQ), we report performance on their full test sets. For those with non-public test sets (HotPotQA, 2Wiki and MuSiQue), we use their full development sets as test sets and report the corresponding performance. Since these three datasets are also used for training, we randomly select 500 questions from their original training sets to serve as development sets, while the remaining questions are used for training. The statistics of experimental datasets can be found in Table 6. Moreover, in KiRAG, we precompute knowledge triples for all the documents in the corpus. The statistics of the resulting KG corpus are also provided in Table 6.

B.2 Baselines

Standard RAG model follows the vanilla retriever-reader pipeline, where the retriever model first retrieves top-K documents from the corpus and the reader model then generates answers based on these retrieved documents. For *IRCoT* and *FLARE*, we use the implementations provided by FlashRAG (Jin et al., 2024). For other models, including *DRAGIN*, *BeamDR* and *Vector-PRF*, we adapt the code released by their authors to align with our experimental setup. Notably, for fair comparison, both our KiRAG and baselines use the same retriever for retrieving documents from the corpus and the same reader for generating answers.

	HotPotQA			2Wiki			MuSiQue			Bamboogle	WebQA	NQ
	Train	Dev.	Test	Train	Dev.	Test	Train	Dev.	Test	Test	Test	Test
Statistics of Experimental Datasets												
# Questions	89,947	500	7,405	166,954	500	12,576	19,438	500	2,417	125	2,032	3,610
Statistics of Retrieval Corpus												
Corpus		HotPotQA			2WikiMultiHopQA			MuSiQue		Wikipedia	Wikipedia	Wikipedia
# Documents		5M			431K			117K		21M	21M	21M
Statistics of the Extracted Knowledge Graph Corpus												
Avg. # Entities per Document		6.93			8.16			9.40		11.12	11.12	11.12
Avg. # Triples per Document		5.91			7.32			8.20		8.33	8.33	8.33

Table 6: Statistics of experimental datasets, retrieval corpus, and pre-computed knowledge graph corpus.

	Train	Dev.
# Questions	115,567	815
Avg. Chain Length	2.36	2.35

Table 7: Statistics of the data used for training the Reasoning Chain Aligner.

B.3 Training and Hyperparameter Details

Training Data Construction. We generate training data for the Reasoning Chain Aligner using existing multi-hop QA datasets. Specifically, for each multi-hop question and its ground-truth relevant documents, we apply TRACE (Fang et al., 2024b) (using the default hyperparameter setting) to construct five potential knowledge triple-based reasoning chains for answering the question. For each chain, we use Llama3 as the reader to generate an answer based on the context provided by the chain. The first chain that successfully produces the correct answer is selected as the ground-truth reasoning chain for that question. The question and its ground-truth reasoning chain will serve as labeled data for training. We filter out questions where all reasoning chains fail to produce the correct answer. In practice, we build training data from the *training sets* of three multi-hop QA datasets: HotPotQA, 2Wiki and MuSiQue. In addition, we use the same procedure to construct development data from the development sets of these three datasets for hyperparameter tuning. The statistics of the data used to train the Aligner are presented in Table 7.

Training Details. For an incomplete reasoning chain r, we treat the correct next triple as positive sample. To generate negative samples, we follow the procedure described in "Knowledge Decomposition" section to obtain a set of candidate triples $\tilde{\mathcal{T}}^i$. The training process uses the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 2e-5 and a weight decay of 0.01. We set the batch size to 64, include 7 negative samples per data point, and use a temperature parameter τ of 0.01. The Aligner is trained for 10 epochs, and we select the checkpoint with the best performance (R@5) on the development set.

Model	Huggingface Checkpoint
E5	`intfloat/e5-large-v2`
BGE	`BAAI/bge-large-en-v1.5`
Llama3	`meta-llama/Meta-Llama-3-8B-Instruct`
Mistral	`mistralai/Mistral-7B-Instruct-v0.2`
Gemma2	`google/gemma-2-9b-it`
Qwen2.5	`Qwen/Qwen2.5-7B-Instruct`
Flan-T5	`google/flan-t5-xl`

Table 8: The specific huggingface checkpoints used in our experiments.

Implementation and Hyperparameter Details. Throughout the experiments, we set the maximum number of iterative steps L to 5. The details of each component in our KiRAG are outlined as follows:

For the *Retriever* model, we use either E5 (Wang et al., 2022) or BGE (Xiao et al., 2024) to retrieve documents. The number of retrieved documents per iteration (i.e., K_0) is 10. For the *Knowledge Decomposition* component, we use Llama3 (Dubey et al., 2024) to extract knowledge triples for each retrieved document. For the *Reasoning Chain Aligner*, given the question and partial reasoning chain, it selects top-20 (i.e., $N = 20$) knowledge triples that are likely to extend the existing chain.

For the *Reasoning Chain Constructor*, we try different LLMs, including Llama3, Mistral (Jiang et al., 2023a) and Gemma2 (Team et al., 2024), to select a triple to extend the partial reasoning chain for subsequent retrieval. We main report the performance of using Llama3 as the Constructor as it achieves the best performance (see Appendix C.9). Moreover, when completing the partial reasoning chain, we filter triples that are not present in the provided candidate set to ensure factual reliability.

Moreover, we leverage different readers to evaluate the QA performance, which includes Llama3, Qwen2.5 (Yang et al., 2024), Flan-T5 (Chung et al., 2024) and TRACE (Fang et al., 2024b). The specific huggingface checkpoints we used in our ex-

Model	HotPotQA		2Wiki		MuSiQue	
	R@3	R@5	R@3	R@5	R@3	R@5
RAG	64.46	69.71	60.50	64.91	39.40	45.16
Vector-PRF	64.38	69.34	60.19	64.37	39.16	43.86
FLARE	53.63	58.83	60.28	69.30	37.10	43.16
DRAGIN	71.71	76.93	62.42	71.14	45.78	52.44
IRCoT	69.96	75.62	60.20	72.23	42.13	48.91
KiRAG-Doc	52.42	67.81	41.42	56.55	28.64	39.82
KiRAG-Sent	47.81	62.99	41.42	56.55	28.27	38.26
KiRAG	**79.69**†	**83.61**†	**78.50**†	**88.94**†	**52.62**†	**58.39**†

Table 9: Retrieval performance (%) using BGE as the retriever model, where the best and the second-best results are marked in bold and underlined, respectively, and † denotes p-value<0.05 compared to the best-performing baseline. Results for BeamDR are omitted as it relies on its own trained BERT model for retrieval, yielding the same results as presented in Table 1.

Model	HotPotQA		2Wiki		MuSiQue	
	EM	F1	EM	F1	EM	F1
RAG	34.11	46.68	14.73	30.16	8.77	17.01
Vector-PRF	33.94	46.58	14.70	30.00	8.65	16.97
BeamDR	38.34	51.64	14.42	27.25	7.08	14.42
FLARE	34.49	46.65	25.01	40.59	13.07	21.38
DRAGIN	41.73	55.68	24.62	40.69	16.43	26.29
IRCoT	43.18	57.08	24.25	40.12	14.89	23.99
KiRAG-Doc	30.47	42.52	11.97	23.97	7.03	14.59
KiRAG-Sent	30.44	42.00	12.82	25.22	8.56	16.16
KiRAG	**45.16**†	**59.85**†	**35.02**†	**54.01**†	**18.87**†	**29.17**†

Table 10: QA performance (%) using BGE as the Retriever. The best and second-best performance are highlighted in bold and underlined, respectively. † indicates p-value<0.05 compared with best-performing baseline.

periments are provided in Table 8.

C Additional Experimental Results and Analysis

C.1 Overall Performance of Using Different Retrievers and Readers

To validate the effectiveness of KiRAG, we provide additional results using different retrievers and readers. Specifically, we replace the E5 Retriever with BGE Retriever for retrieving documents from the corpus and the other components remain unchanged. The corresponding retrieval and QA performance are presented in Table 9 and Table 10, respectively. The results are consistent with those obtained using the E5 Retriever, demonstrating the adaptability and effectiveness of our KiRAG across different retriever models.

Moreover, to assess the quality of the documents retrieved by KiRAG, we report QA performance using different reader models in Table 11. The results suggest that KiRAG consistently outperforms all the baselines across different readers, demonstrating its ability to provide high-quality retrieval results that enhance downstream QA performance.

Reader	Model	HotPotQA		2Wiki		MuSiQue	
		EM	F1	EM	F1	EM	F1
Qwen2.5	**RAG**	34.69	46.15	33.37	38.51	9.14	17.17
	Vector-PRF	34.54	46.12	33.41	38.51	8.90	16.91
	BeamDR	39.61	51.51	22.41	29.95	6.70	14.06
	FLARE	36.30	47.33	36.73	44.38	12.58	21.22
	DRAGIN	44.07	56.91	36.75	44.49	18.16	28.68
	IRCoT	43.44	56.46	38.39	45.97	15.60	25.36
	KiRAG-Doc	35.68	47.15	29.23	34.51	7.61	15.27
	KiRAG-Sent	34.21	45.64	29.78	34.92	9.64	17.83
	KiRAG	**47.89**†	**61.41**†	**47.42**†	**56.02**†	**19.73**†	**30.79**†
Flan-T5	**RAG**	37.08	47.32	17.42	22.05	8.94	15.06
	Vector-PRF	37.02	47.23	31.58	36.26	8.98	15.16
	BeamDR	41.89	52.83	18.73	23.29	7.03	12.21
	FLARE	39.81	50.46	33.31	39.95	13.28	19.74
	DRAGIN	46.75	58.52	34.19	40.68	18.12	25.22
	IRCoT	47.32	59.05	35.90	42.31	16.42	23.61
	KiRAG-Doc	36.18	46.45	25.03	29.58	7.45	13.53
	KiRAG-Sent	38.86	49.59	31.27	36.58	11.34	17.98
	KiRAG	**49.31**†	**61.38**†	**39.99**†	**46.51**†	**19.07**	**27.29**†
TRACE	**RAG**	39.18	51.82	21.10	34.28	11.63	19.49
	Vector-PRF	38.85	51.54	21.91	34.81	11.58	19.59
	BeamDR	43.21	56.28	21.01	33.23	10.67	18.26
	FLARE	39.31	51.40	31.85	45.50	14.89	23.75
	DRAGIN	44.29	57.64	31.88	45.55	18.11	27.41
	IRCoT	45.29	58.77	32.05	46.55	16.84	25.78
	KiRAG-Doc	45.36	58.82	29.41	43.99	13.69	22.44
	KiRAG-Sent	43.38	56.68	27.51	41.92	16.84	25.59
	KiRAG	**46.41**	**60.22**†	**33.13**	**48.49**†	**19.32**	**29.10**†

Table 11: QA performance (%) using different Reader models, where † indicates p-value<0.05 compared with best-performing baseline.

Model	HotPotQA		2Wiki		MuSiQue	
	EM	F1	EM	F1	EM	F1
RAG	34.34	47.72	12.95	29.94	9.43	17.50
Vector-PRF	34.56	47.87	13.76	30.42	10.10	17.74
BeamDR	38.60	52.24	14.49	28.71	7.74	15.13
FLARE	35.08	48.07	24.87	41.95	13.20	22.23
DRAGIN	41.16	55.48	24.47	41.37	17.54	27.74
IRCoT	41.61	56.01	24.89	42.90	14.85	24.48
KiRAG-Doc	36.87	50.68	15.54	32.17	9.23	17.50
KiRAG-Sent	36.58	50.09	15.49	31.59	12.16	20.66
KiRAG	**43.81**†	**58.42**†	**27.26**†	**47.59**†	**17.58**	**28.92**†

Table 12: QA performance (%) using top-5 retrieved document as context. The best and second-best results marked in bold and underlined, respectively. † denote p-value<0.05 compared with best-performing baseline.

C.2 QA Performance based on Top-5 Documents

Table 12 presents the QA performance using the top-5 retrieved documents as the context, demonstrating similar results to those obtained with the top-3 retrieved documents.

C.3 Retrieval Performance at Different Steps on MuSiQue Dataset

Figure 9 presents the retrieval performance of KiRAG and baseline methods at different steps on the MuSiQue dataset, showing similar trends to those observed on the HotPotQA and 2Wiki datasets.

Model	Bamboogle		WebQA		NQ	
	EM	F1	EM	F1	EM	F1
RAG	15.20	22.66	18.41	31.04	35.93	41.18
Vector-PRF	15.20	23.73	18.31	31.02	36.09	41.25
BeamDR	11.20	15.29	15.50	25.46	21.63	25.31
FLARE	24.00	31.93	20.57	31.47	31.22	35.17
DRAGIN	26.20	37.68	20.37	32.31	35.43	39.87
IRCoT	21.60	33.69	19.39	31.31	**37.34**	**42.50**
KiRAG-Doc	17.60	27.92	18.36	30.75	33.63	39.22
KiRAG-Sent	16.00	28.15	19.14	31.27	33.60	38.12
KiRAG	**29.60**†	**42.00**†	**20.67**	**32.87**	36.29	41.49

Table 13: QA performance (%) on unseen multi-hop and single-hop QA datasets, where † denotes p-value<0.05 compared with best-performing baselines.

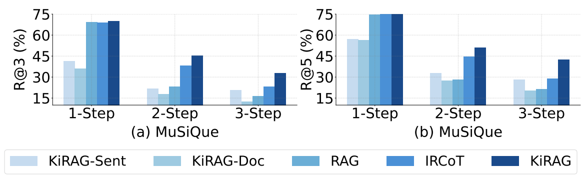


Figure 9: Retrieval performance (%) at different steps on MuSiQue dataset.

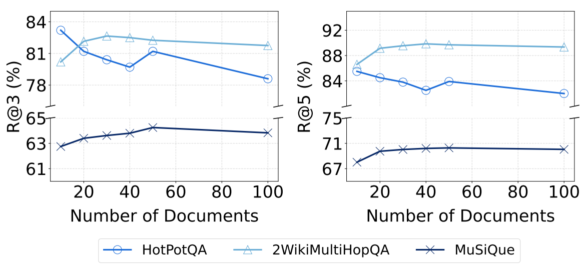


Figure 10: Retrieval performance (%) of KiRAG under different values of K_0 on three multi-hop QA datasets.

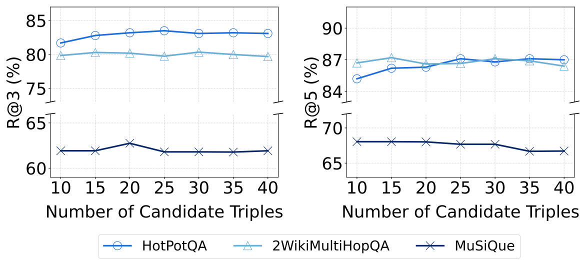


Figure 11: Retrieval performance (%) of KiRAG under different values of N on three multi-hop QA datasets.

C.4 QA Performance on Unseen Datasets

Due to page limit, we present the QA performance on unseen multi-hop and single-hop QA datasets in Table 13, which aligns with the retrieval performance reported in Table 3. The results highlight that KiRAG can effectively generalise to different types of QA tasks, maintaining high performance without overfitting to specific training data.

C.5 Details and Examples of Manually Labeled Relevant Knowledge Triples

To quantitatively evaluate the quality of knowledge triples retrieved using our proposed knowledge-driven iterative retrieval framework, we manually label relevant knowledge triples for 100 questions randomly sampled from the 2Wiki dataset. Specifically, for each multi-hop question and its ground-truth relevant documents, we use Llama3 to extract knowledge triples from these relevant documents, and then manually select a subset of knowledge triples that directly support answering the question. We provide some examples of the manually curated data in Table 14.

C.6 Effect of the Number of Initially Retrieved Documents

During the iterative retrieval process of KiRAG, the Retriever model initially retrieves K_0 documents from the corpus, from which relevant knowledge can be extracted. To examine the impact of K_0, we vary its value from 10 to 100. Figure 10 illustrates the retrieval performance of KiRAG under different values of K_0 on the development sets of three multi-hop QA datasets. The results indicate that increasing K_0 beyond a certain point can degrade performance. This occurs because a larger document pool raises the likelihood of including noisy or irrelevant knowledge triples, making it more challenging for the Reasoning Chain Aligner to accurately identify the triples essential for answering multi-hop questions. Therefore, it is crucial to select a proper K_0 to achieve superior retrieval performance.

C.7 Effect of the Number of Candidate Triples

In the iterative retrieval process of KiRAG, the Reasoning Chain Aligner selects N knowledge triples that are most likely to form a coherent reasoning chain with the existing chain. To investigate the effect of N, we vary its value from 10 to 40. Figure 11 shows the retrieval performance of KiRAG under different values of N on the development sets of three multi-hop QA datasets. The results indicate that KiRAG is not sensitive to the value of N, as the performance remains relatively stable across different values. This stability can be attributed to the powerful reasoning and contextual understanding abilities of the Reasoning Chain Constructor, which effectively identifies the most useful triple even from a potentially noisy set of candidates triples.

Question: Which film came out first, Blind Shaft or The Mask Of Fu Manchu?
Relevant Knowledge Triples: <Blind Shaft; release year; 2003>, <The Mask of Fu Manchu; release year; 1932>

Question: When did John V, Prince Of Anhalt-Zerbst's father die?
Relevant Knowledge Triples: <John V, Prince of Anhalt-Zerbst; father; Ernest I, Prince of Anhalt-Dessau>, <Ernest I, Prince of Anhalt-Dessau; death date; 12 June 1516>

Question: Which film has the director died first, Crimen A Las Tres or The Working Class Goes To Heaven?
Relevant Knowledge Triples: <Crimen a las tres; director; Luis Saslavsky>, <The Working Class Goes to Heaven; director; Elio Petri>, <Luis Saslavsky; death date; March 20, 1995>, <Elio Petri; death date; 10 November 1982>

Question: Who died first, Fleetwood Sheppard or George William Whitaker?
Relevant Knowledge Triples: <Fleetwood Sheppard; death date; 25 August 1698>, <George William Whitaker; death date; March 6, 1916>

Question: Who is the spouse of the director of film Eden And After?
Relevant Knowledge Triples: <Eden and After; director; Alain Robbe-Grillet>, <Alain Robbe-Grillet; spouse; Catherine Robbe-Grillet>

Table 14: Examples of manually labeled relevant knowledge triples for multi-hop questions on the 2Wiki dataset.

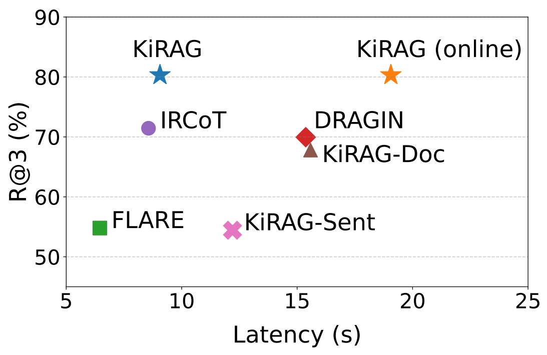


Figure 12: Retrieval performance (R@3) v.s. average latency per question for different models on the HotPotQA test set. KiRAG (online) represents a variant of our approach where knowledge triples extracted dynamically during iterative retrieval, without precomputation.

Model	HotPotQA		2Wiki		MuSiQue	
	R@3	R@5	R@3	R@5	R@3	R@5
IRCoT	71.44	77.57	64.30	75.56	45.61	52.21
KiRAG (Llama3)	**80.32**	**84.08**	**77.76**	**85.32**	54.53	61.16
KiRAG (Mistral)	74.14	79.51	74.14	82.30	49.10	56.65
KiRAG (Gemma2)	79.66	84.03	77.04	83.59	**54.82**	**62.42**

Table 15: Retrieval performance (%) of KiRAG using different LLM-based Reasoning Chain Constructor.

C.8 Efficiency Analysis

We evaluate the efficiency of KiRAG in comparison to the baseline models. Specifically, we conduct experiments on a 3.5 GHZ, 32-cores AMD Ryzen Threadripper Process paired with an NVIDIA A6000 GPU. For fair comparison, both KiRAG and baselines leverage the same E5 model for document retrieval and the same Llama3 model as the reasoning component. It is worth noting that the knowledge triple extraction in our KiRAG is query-independent and precomputed, which helps to improve efficiency. To evaluate the impact of precomputing triples, we introduce a variant: *KiRAG (online)*, where knowledge triples are dynamically extracted during the iterative retrieval process.

Figure 12 presents the average latency and retrieval performance of different models on the HotPotQA test set, which yields the following findings: (1) Compared with KiRAG (online), KiRAG substantially reduces latency without compromising retrieval performance, highlighting the efficiency benefits of precomputed knowledge triple extraction; (2) KiRAG exhibits latency comparable to IRCoT while achieving significantly better retrieval performance, indicating that our approach effectively enhances retrieval effectiveness without introducing substantial computational overhead. (3) KiRAG achieves a better balance between retrieval effectiveness and efficiency compared to baselines, as evidenced by the relatively lower latency and higher retrieval recall.

C.9 Performance of Using Different LLM-Based Constructor

KiRAG leverages a frozen LLM as the Reasoning Chain Constructor to maintain the adaptability of our framework. Figure 15 presents the retrieval performance of our KiRAG using different LLM-based Constructor. The results indicate that KiRAG consistently outperforms IRCoT across different Constructors, indicating the robustness of our approach in improving retrieval performance regardless of the specific LLM used.