DRR-RAG: Decompose and Refine Reasoning in Retrieval-Augmented Generation for Multi-hop Question Answering

Anonymous ACL submission

Abstract

Retrieval-augmented generation systems are effective in addressing hallucinations and domain-specific challenges in vertical domains. However, these systems often struggle to fully utilize the language capabilities of large language models (LLMs) in handling complex questions that require matching relevant documents from different sources and managing intricate dependencies. In this paper, we introduce a novel framework, Decompose and Refine Reasoning in Retrieval-Augmented Generation, that leverages the power of LLMs to decompose complex queries and efficiently manage the relationships between sub-questions, enhancing document retrieval and addressing multi-hop question challenges. We conduct experiments using a local model, a closed-source model with prompt, and fine-tuning. Through extensive experimentation on diverse multi-hop datasets, we demonstrate that our approach not only outperforms existing methods in handling complex queries and improving retrieval performance but also proves effective, easy to implement, and highly usable. These results highlight the robustness and practicality of our framework.

1 Introduction

Large language models (LLMs) have demonstrated exceptional capabilities in understanding and generating natural language across diverse tasks (Achiam et al., 2023; Touvron et al., 2023; Jiang et al., 2023). However, they often face limitations such as domain-specific knowledge gaps and hallucinations (Zhang et al., 2023; Huang et al., 2023; Yang et al., 2023), especially for queries requiring up-to-date or nuanced information. Retrieval-augmented generation (RAG) (Lewis et al., 2020) effectively mitigates these issues by grounding LLM responses in retrieved factual evidence, enhancing accuracy and reliability.

In the early stages of RAG research, scholars focus on various ways to improve the basic RAG framework. They explore different retrieval algorithms and strategies to better match the input query with the most relevant documents (Thorne et al., 2018; Trischler et al., 2017; Rajpurkar et al., 2016). In addition, efforts are made to optimize the generation process so that the LLM could more effectively integrate the retrieved information into the final answer.

However, when it comes to handling complex Multi-hop questions, these early methods faced considerable difficulties. Multi-hop question answering (MHQA) tasks (Mavi et al., 2024) necessitate integrating diverse information sources and logical reasoning to derive accurate answers. To address these challenges, mainstream approaches increase the number of iterations, continuously decomposing the question and solving subquestions in each iteration. These methods typically follow the retrieve-and-read paradigm (Zhu et al., 2021), which consists of two key components: a passage retriever that filters irrelevant information and a reader that iteratively refines the retrieved content to extract the correct answer (Tu et al., 2020; Xiong et al., 2021; Wu et al., 2021; Trivedi et al., 2022a; Li et al., 2023). These approaches enhance the accuracy and efficiency of question-answering systems by systematically improving retrieval and analysis in each iteration.

Nevertheless, existing methods face several critical challenges: **1)** During problem decomposition, current methodologies only derive and process the first sub-question from the original Multi-hop question, subsequently generating follow-up sub-questions and sub-answers based on retrieval results. This process fails to account for complex interdependencies between sub-questions adequately and lacks a verification mechanism to assess the validity and rationality of the decomposition. Errors introduced during decomposition may lead to retrieval errors at later stages; **2)** In the retrieval phase, excessively incorporating preceding multi-turn dialogue content as input risks introducing substantial irrelevant or misleading information, which may prevent the retriever from effectively capturing the necessary diverse evidence and the LLM from summarizing the correct answer from an overly long context.

Inspired by the impressive capabilities of current reasoning models such as OpenAI o1 (Jaech et al., 2024) and Deepseek R1 (Guo et al., 2025), as well as their principles for solving complex questions, this paper introduces an innovative framework—**D**ecompose and **R**efine **R**easoning in **R**etrieval-**A**ugmented **G**eneration(DRR-RAG)—to address the aforementioned challenges:

(1) Our **Question Decompose** module leverages LLMs to decompose complex questions into simpler sub-questions and manage dependencies between them. With dependency relationships established, when supplementing information for the current sub-question, it is only necessary to provide information from its preceding sub-question. This effectively mitigates the impact of excessive irrelevant information on retrieval and answer generation, as mentioned in **2)** above.

(2) Regarding the rationality of question decomposition raised in **1)**, we incorporate a **Self-Refine** mechanism after each iterative retrieval and answer generation cycle. By extracting relevant evidence from paragraphs and sub-questions, this mechanism not only verifies the correctness of each sub-answer generated by the LLMs but also assesses the reasonableness of the question decomposition during the evidence extraction process, allowing for potential re-decomposition of the original multi-hop question when necessary.

(3) Comparative experiments conducted on multiple datasets demonstrate that our method outperforms existing RAG approaches, achieving state-of-the-art performance on MHQA tasks. Further **analytical experiments** validate that the proposed framework demonstrates strong adaptability under resource constraints and supports flexible, lightweight implementations through diverse approaches.

In summary, our key contributions include:

- A innovation framework that decomposes complex questions into small ones, considers the dependencies between sub-questions and solves them sequentially.
- An adaptive Self-Refine strategy to mitigate the impact of irrelevant or noisy documents and determine the decomposition errors during the iterative retrieval process, ensuring factual consistency and logical coherence in generated answers.
- Extensive experiments demonstrate the effectiveness, ease of implementation, and robustness of our approach in MHQA, thereby fully substantiating the superiority of our method.

2 Related Work

2.1 Retrieval-augmented Generation

RAG augments the input space of LLMs with retrieved text documents (Lewis et al., 2020; Guu et al., 2020), leading to significant improvements in knowledge-intensive tasks. RAG typically consists of several key components: vectorization of queries and documents, relevant document retrieval, and the generation of answers by LLMs. The central challenge of RAG lies in effectively identifying relevant documents to the query. In recent years, several improvements have been proposed to enhance the recall accuracy in this process. Initial methods include Chain-of-Thought (CoT) reasoning (Wei et al., 2022), keyword-based retrieval, and relevance-based reranker model of the retrieved documents to improve the similarity between the retrieved passages and the query.

At the same time, some researchers have observed the gap between Retriever (i.e. Embedding) and LLM. For instance, R^2AG (Ye et al., 2024) addresses the semantic gap between Retriever and LLM by introducing R^2-Former, an intermediate module that bridges this gap without necessitating fine-tuning of either the Retriever or the LLM.

2.2 Multi-hop Question Answering

While these methods perform well for simple, non-interdependent queries, they face challenges in handling Multi-hop question answering(MHQA). In such cases, researchers have proposed frameworks with Iterative Retrieval such as ITER-RETGEN (Shao et al., 2023), GenGround (Shi et al., 2024), and EfficientRAG (Zhuang et al., 2024) to better address these challenges.

ITER-RETGEN concatenates the input question with the generated output from the previous iteration to form a new query for the next. This method addresses the issue of insufficient information retrieval when dealing with complex multi-hop questions. However, directly concatenating the answer

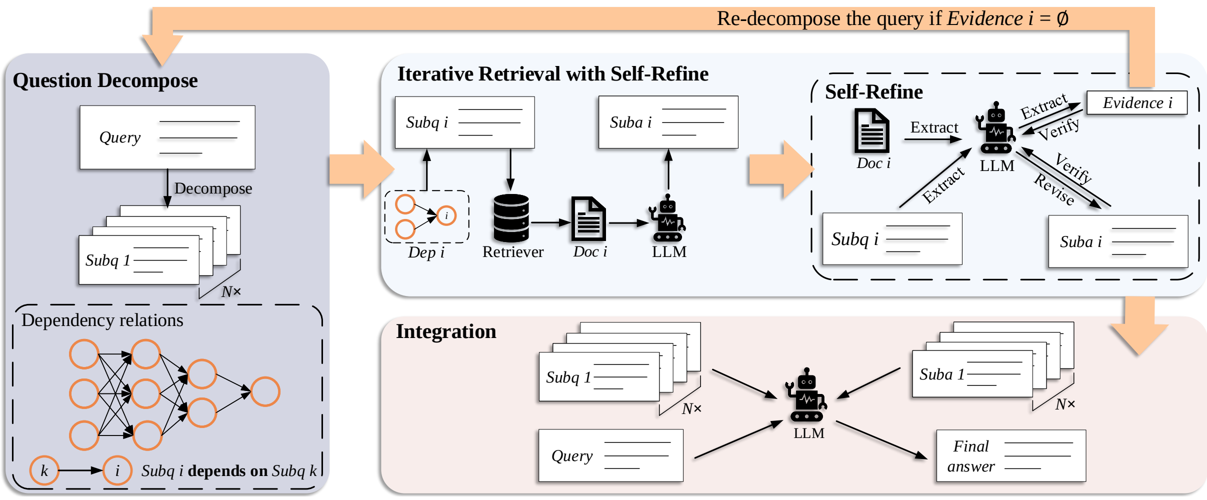

Figure 1: Decompose and Refine Reasoning in Retrieval-Augmented Generation framework diagram.

generated by the previous iteration with the query will inherently contain some noise in the retrieval process (Xu et al., 2023).

GenGround introduces a novel framework that improves the accuracy of complex queries by iteratively decomposing the question and refining the answer. This framework involves a cyclical process of question decomposition and answer correction. Despite its promising performance, the approach has significant data requirements, as it necessitates the availability of ground-truth data, which limits its applicability across various domains due to the dependence on high-quality labelled datasets.

Similarly, EfficientRAG targets multi-hop questions by fine-tuning the model on question decomposition steps provided by the dataset. The model is trained to generate the subsequent hop question and judge the relevance of retrieved documents. While this method enhances multi-hop reasoning, it also has considerable data requirements, as it needs the dataset to provide steps for both question generation and answer evaluation.

Overall, existing iterative retrieval methods have made innovative progress in question decomposition and answer verification. However, deficiencies remain in understanding the internal structure of complex questions and validating the decomposition process.

3 Method

In this section, we present a detailed explanation of the DRR-RAG framework. As illustrated in Fig. 1, the entire process consists of the following phases: Question Decomposition, Iterative Retrieval with Self-Refine and Integration. First, the multi-hop question is decomposed into different types of sub-questions. These sub-questions are then addressed in an order determined by their dependency relationships, ensuring that they only obtain necessary information from their dependencies. Throughout the retrieval and resolution process, the system performs self-refine to verify both the correctness of sub-question answers and the validity of the question decomposition, allowing for re-decomposition if needed. Finally, after all sub-questions are resolved, their answers are integrated to solve the original question.

3.1 Question Decompose

The Decompose module, serving as the most critical component of the process, represents a departure from previous approaches. As illustrated in Fig. 2, our methodology generates all subquestions during the initial phase, as well as their interdependent relationships.

This module consists of two steps: question classification and decomposition. Prior to question decomposition, we initially conduct question classifying. As shown in Fig. 3, Multi-Hop questions can be categorized into three types: inference-based, comparison-based, and hybrid types. In the case of inference-based questions, the decomposed sub-problems exhibit a chain-like dependency relationship. For instance, a set of decomposed sub-questions $\{subq_i\}_{i=1}^{t}$ from question $\mathcal{Q}$, where $subq_{i-1} \rightarrow subq_i$. Regarding a comparison-based relationship, there is no dependency between the sub-questions: $subq_i \parallel subq_j$ for $i \neq j$. The hybrid type represents a combination of the above two fundamental types, including

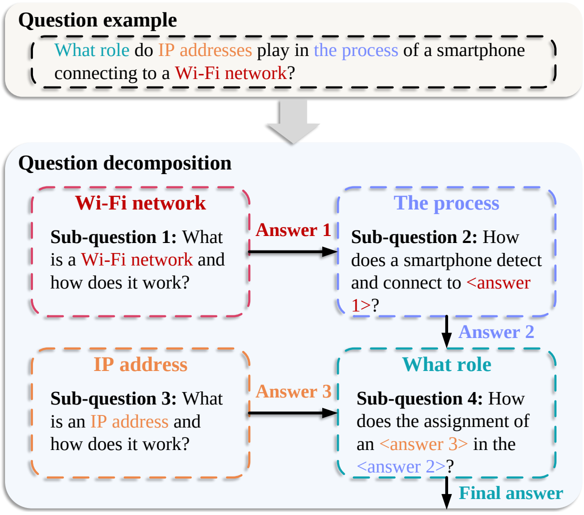

Figure 2: Question decompose example. <answer_i> is a placeholder for the answer to the i-th subquestion, which will be filled with the corresponding answer during the subsequent Iterative Retrieval process.

nested inference, nested comparison, and inference-comparison hybrid structures. Then, we input $\mathcal{Q}$ into the LLM, denoted as $\mathcal{M}$, to obtain $\{subq_i\}_{i=1}^{t}$ and the dependency relations $\{dep_i\}_{i=1}^{t}$ among all sub-questions:

$$\{subq_i\}_{i=1}^{t}, \{dep_i\}_{i=1}^{t} = \mathcal{M}(\mathcal{P}_D(\mathcal{Q})) \quad (1)$$

$\mathcal{P}_D$ represents the Decompose instruction prompt shown in Appendix 9.

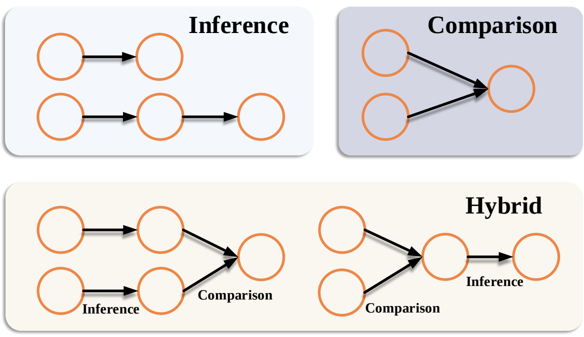

Figure 3: Different types of multi-hop questions.

3.2 Iterative Retrieval with Self-Refine

As can be seen from the Dependency relations in the Question Decompose module of Fig. 1, the interdependent sub-questions form a directed acyclic graph. We solve them sequentially according to the process outlined in Algorithm 1, which consists of three components: Initialization, Retrieval, and Self-Refine, where *Queue* represents the queue data structure, pop refers to removing the element from the front of the queue, push refers to adding an element to the back of the queue, $in\_deg(i)$ represents the in-degree and $subq_i$.format($suba_j$) denotes replacing the corresponding placeholder in $subq_i$ with the content of $suba_j$, $\mathcal{P}_I$ is the prompt that guides $\mathcal{M}$ to generate responses.

Algorithm 1: Iterative Retrieval with Self-Refine

Require : sub-questions $\{subq_i\}_{i=1}^{t}$,
Dependency relations $dep(i)$
Return : sub-answers $\{suba_i\}_{i=1}^{t}$

$Queue \leftarrow \emptyset$
for $i = 1$ *to* t **do**
  $in\_deg(i) \leftarrow |dep(i)|$
  if $in\_deg(i) = 0$ **then**
    $Queue$.push(i)
  end
end
while $Queue \neq \emptyset$ **do**
  $i \leftarrow Queue$.pop()
  for each $j \in dep(i)$ **do**
    $subq_i \leftarrow subq_i$.format($suba_j$)
  end
  $\hat{D}_i \leftarrow \mathcal{R}(subq_i, D_i, top\text{-}k)$
  $suba_i \leftarrow \mathcal{M}(\mathcal{P}_I(subq_i, \hat{D}))$
  Self-Refine($subq_i, suba_i, \hat{D}_i$)
  for each j *in* $subq_i \rightarrow subq_j$ **do**
    $--in\_deg[j]$
    if $in\_deg[j] = 0$ **then**
      $Queue$.push(j)
    end
  end
end
Function ***Self-Refine***($subq_i, suba_i, \hat{D}_i$)**:**
$E_i \leftarrow$ Extract($\hat{D}_i, subq_i$)
if $E_i = \emptyset$ **then**
  goto re-Decompose
else
  if $\neg Verify(suba_i, E_i)$ **then**
    $suba_i \leftarrow$ Revise($suba_i, E_i$)
  end
end

Initialization: We create an empty queue *Queue* and then iterate through each $subq_i$. For the $subq_i$, we set $in\_deg(i)$ to the size of $dep(i)$, and then add $subq_i$ to the Queue if $in\_deg(i) = 0$.

Retrieval: $subq_i$ is popped from the front of the Queue per Iteration. We initially augment it with contextual information $suba_j$ from its dependency set $dep(i)$, where $j \in dep(i)$. The retriever R

selects the top-k most relevant documents $\hat{D}_i$ from the candidate documents D_i for $subq_i$:

$$\hat{D}_i = \mathcal{R}(subq_i, D_i, top\text{-}k) \tag{2}$$

Then, the augmented $subq_i$ is concatenated with $\hat{D}_i$ as the input context for answer $suba_i$ generation, using Prompt $\mathcal{P}_I$ for instruction:

$$suba_i = \mathcal{M}(\mathcal{P}_I(subq_i, \hat{D}_i)) \tag{3}$$

Self-Refine: The entire module is guided by the instruction shown in the Appendix 11 to enable task execution through $\mathcal{M}$. As shown in line 23 of the Algorithm: First, extract supporting evidence E_i from $\hat{D}_i$ that substantiates $subq_i$. Subsequently, if no supporting E_i can be retrieved from $\hat{D}_i$, This indicates that there is no content related to $subq_i$ in $\hat{D}_i$, and further suggests that the original multi-hop question $\mathcal{Q}$ needs to be decomposed through another approach. In this case, the module provides structured Feedback to the decomposer to trigger the re-decomposition of the original problem. Otherwise, verify the correctness of the generated assertion $suba_i$ against this E_i. If $suba_i$ is identified as erroneous, the module initiates the process to revise $suba_i$ based on E_i.

3.3 Integration

Once all $\{subq_i\}_{i=1}^{t}$ have been addressed, $\{suba_i\}_{i=1}^{t}$ along with $\{suba_i\}_{i=1}^{t}$ and original $\mathcal{Q}$ are submitted to $\mathcal{M}$ to generate the final answer $\mathcal{A}$:

$$\mathcal{A} = \mathcal{M}(\mathcal{P}_f(\mathcal{Q}, \{subq_i, suba_i\}_{i=1}^{t})) \tag{4}$$

Here, $\mathcal{P}_f$ is the instruction for $\mathcal{A}$ generation.

4 Experiments

4.1 Datasets

To evaluate the system's retrieval efficacy and information extraction accuracy in handling multi-hop questions, we employed four established MHQA benchmarks—2WikiMultihopQA (2Wiki) (Ho et al., 2020), MuSiQue (Trivedi et al., 2022b), HotpotQA (Hotpot) (Yang et al., 2018), and ConcurrentQA (CQA) (Arora et al., 2023)—which are widely adopted for multi-round retrieval evaluation in RAG-related research. These datasets encompass multi-hop reasoning scenarios, including comparative analysis and hybrid inference tasks.

We conduct experiments for each dataset using the validation set, as the validation questions are more complex than those in the training set. Among the fields the dataset provides, we only utilize the question and candidate documents for answer generation. Subsequently, the generated answers are then evaluated for correctness using the ground-truth answers. Additionally, we use the evidence documents for recall rate calculation to assess the retrieval effectiveness.

	#Training	#Val.	#Type	#Candidate docs
2Wiki	167,454	8,757	2,4-hop	10
MuSiQue	19,938	2417	2,3,4-hop	20
Hotpot	90,447	7405	2-hop	10
CQA	15239	1600	2-hop	2-22 (u=10.25)

Table 1: Statistic of the 4 datasets. "#candidate docs" indicates the documents pools for each question. u = 10.25 indicates that the average number of Candidate docs is 10.25.

4.2 Few-Shot Prompting

Our prompt engineering framework employs a few-shot prompting methodology to enhance task decomposition capabilities. The proposed prompt template is structured into three main components: the instruction, the JSON format, and the examples, as detailed in Appendix C.

4.3 Fine-tuning Details

Data Construction: It is imperative to construct a high-quality dataset that adheres to format specifications and ensures the correctness of decomposition to ensure the effectiveness of fine-tuning. Initially, we utilized the GLM-4-Flash to generate corresponding decomposed sub-questions from the original questions on the training sets of several datasets rather than directly using the generated sub-questions as labels for training data. After integrating responses to all sub-questions and filtering out the decomposed data that correctly answered the original question, we employ this verified and accurate data to fine-tune GLM-4-9B (GLM et al., 2024).

Train Details: The detailed training settings are provided in Appendix A.

4.4 Baselines

In the mainstream RAG enhancement directions, we have selected a variety of representative methods to verify the effectiveness of our approach:

- **Single-round retrieval**: BaseRAG: standard RAG process (Lewis et al., 2020) without any optimization; CoT: a RAG variant incorporating Chain-of-Thought reasoning; Self-RAG (Asai

Method/Dataset	2Wiki			MuSiQue			Hotpot			CQA		
	EM	F1	R@3	EM	F1	R@3	EM	F1	R@3	EM	F1	R@3
Single-round Retrieval												
BaseRAG	44.05	44.19	70.52	25.06	27.45	50.86	46.18	48.45	61.58	53.68	55.30	61.58
CoT	49.32	47.66	72.44	26.93	29.33	55.47	52.18	55.30	66.82	62.18	58.30	67.32
R^2AG	74.44	63.51	81.69	35.06	32.58	67.74	63.75	65.30	73.80	64.38	60.33	73.80
SelfRAG	64.28	66.73	83.42	34.03	38.91	80.43	52.89	61.87	87.90	67.89	63.67	88.90
Graph Retrieval												
GEAR	59.74	62.35	82.27	28.94	35.66	78.83	55.4	65.4	85.42	67.54	63.32	77.96
Multi-round Retrieval												
Iter-RetGen	68.79	62.11	83.05	37.67	37.25	82.26	58.07	54.07	74.29	63.25	64.07	84.29
EfficientRAG	69.90	60.93	81.84	35.91	31.94	84.51	62.82	64.33	84.08	70.90	67.37	86.57
Auto-RAG	64.82	66.74	86.63	33.41	30.34	83.51	60.46	62.37	80.25	68.52	66.32	84.08
Ours												
DRR-RAG	79.06	73.04	**99.5**	48.43	47.07	**93.21**	67.76	68.45	**95.87**	74.14	68.57	92.22
DRR-RAG*	**79.21**	**73.14**	99.37	**48.60**	**47.21**	93.12	67.81	68.45	95.6	72.74	67.23	91.93
DRR-RAG†	77.03	71.82	98.69	47.09	46.29	92.68	**68.12**	**68.69**	95.12	**75.13**	**68.90**	**92.52**

Table 2: Results of various methods across four benchmarks. The highest values of each metric are highlighted in bold. *:Use glm-4-flash model with few-shot prompting. †:Use glm-4-9b model with LoRA fine-tuning.

et al., 2023) which enhances the judgment of retrieved document; R^2AG that narrows the gap between Retriever and LLMs.

- **Multi-round retrieval**: ITER-RETGEN, EfficientRAG, and Auto-RAG (Yu et al., 2024), which also leverage multiple retrieval rounds to solve the sub-question and refine the response quality.
- **Graph retrieval**: GEAR (Shen et al., 2024) that combines graph-processing techniques to model complex relationships between entities.

4.5 Evaluation Metrics

Consistent with prior research, we respectively utilize Recall@k (R@k), Exact Match (EM) and F1 scores to assess RAG's retrieval and generation capabilities. R@k represents the proportion of relevant documents among the top k retrieved documents for a given query. EM means whether the passage-level prediction is the same as the ground truth, while retrieval F1 is the harmonic mean of precision and recall.

4.6 Implementation Details

The question decomposition module employs three distinct implementation approaches: the closed-source model GLM-4-Flash and the GLM-4-9B which can be deployed on consumer-grade GPUs, both utilizing few-shot prompting, as well as the GLM-4-9B fine-tuned with LoRA (Hu et al., 2021). In other LLM applications, we deploy GLM-4-9B. For document retrieval, we employ the BGE-large-en-v1.5 (Xiao et al., 2024) as the retriever, selecting the *top*-3 documents for each question. Additionally, we guide the LLM to output results in JSON format for ease of processing. We further aligned and corrected the format using the LLM for outputs that did not conform to the required format (which could potentially disrupt the experiment). The detailed model deployment is presented in the Appendix B.

5 Results and Analysis

5.1 Main Results

Table 2 presents the evaluation results across four benchmarks, highlighting the superior performance of DRR-RAG. Compared to baseRAG, DRR-RAG improves performance by over 20%, with a 30% increase on the 2Wiki dataset. It outperforms existing methods by 3–6% on the 2Wiki, Hotpot, and CQA datasets. Notably, DRR-RAG excels on the MuSiQue dataset, particularly known for its complex question structures, showing a more than 10% advantage over all other approaches. These results confirm the enhanced capability of DRR-RAG in handling MHQA.

A horizontal comparison of the three DRR-RAG implementations demonstrates their robust performance across benchmark datasets: 1) when comparing two few-shot implementation models, the 9B model deployable on consumer-grade GPUs

Dataset	Metric	Base	+ Question Decompose	++ Question Classify	+ + + Self-Refine
2Wiki	EM	54.05	76.36 (**22.31**↑)	78.46 (**2.10**↑)	79.21 (**0.75**↑)
	F1	54.19	70.41 (**16.22**↑)	72.6 (**2.19**↑)	73.14 (**0.54**↑)
MuSiQue	EM	25.06	42.13 (**17.07**↑)	44.19 (**2.06**↑)	48.60 (**4.41**↑)
	F1	27.45	40.21 (**12.76**↑)	42.51 (**2.30**↑)	47.21 (**4.70**↑)
Hotpot	EM	46.18	63.73 (**17.55**↑)	66.35 (**2.62**↑)	67.81 (**1.46**↑)
	F1	48.30	64.42 (**16.12**↑)	66.88 (**2.46**↑)	68.45 (**1.57**↑)
CQA	EM	53.68	68.40 (**14.72**↑)	72.26 (**3.86**↑)	73.74 (**1.48**↑)
	F1	55.30	65.20 (**9.90**↑)	67.07 (**1.87**↑)	67.23 (**0.16**↑)

Table 3: Ablation study results. For each dataset, values in parentheses indicate the performance gap between each ablated variant and the previous variant. The arrows show the direction of the difference.

achieves comparable performance to the closed-source Flash model; 2) In comparing few-shot versus LoRA fine-tuning approaches for the 9B model: the LoRA method shows slightly inferior performance on 2Wiki and MuSiQue datasets while demonstrating marginal advantages on the Hotpot and CQA dataset.

5.2 Analysis

To investigate how our proposed method improves the evaluation metrics, we test the recall rates of the target documents for all approaches under *top*-3 setting. The recall rates for different methods are presented in Table 2 for a clear comparison. Our proposed method achieved a target document recall rate exceeding 90%, significantly surpassing previous methods. Such a high recall rate is critical in ensuring the system's ability to generate answers from recalled documents correctly.

5.3 Ablation Experiments

We conduct ablation experiments on the Question Decompose, Self-Refine, and Classify modules to assess the impact of different modules. As shown in Table 3, the Question-Decompose module has the most significant effect, improving performance by over 10% across all benchmarks. The Classify module provides a consistent average gain of 2.4%, while the Self-Refine module showed a general improvement of over 1%, with a notable 4.5% boost on the MuSiQue benchmark.

5.4 Strong Adaptablity to Constraints

In this subsection, we explore the impact of the number of document recalls and model size.

Retrieval Constraint: The number of retrieved documents plays a critical role in the performance of RAG systems. To evaluate the stability of the RAG system under different retrieval settings, we tested the EM score across multiple multi-hop datasets with the number of retrieved documents varying from 1 to 5. As shown in Fig. 4, the results demonstrate consistently high scores across different retrieval counts. Our system exhibits strong adaptability to varying retrieval numbers, with only minimal accuracy loss when the retrieval count is reduced to 1. This indicates that the system is highly robust to variations in the retriever's performance, maintaining reliable effectiveness even under constrained retrieval conditions.

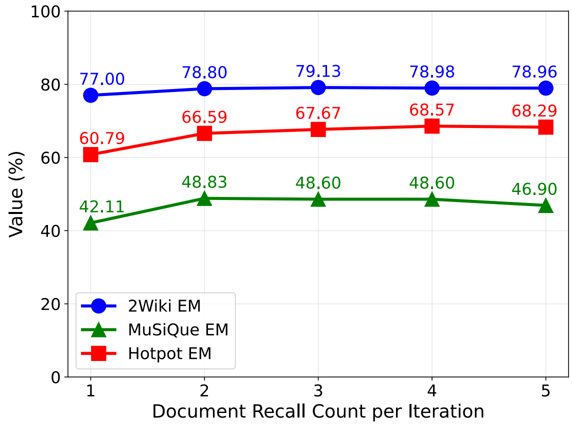

Figure 4: Evalution performance of DRR-RAG across different document numbers provided per iteration.

Model Size Constraint: We test the question-decomposition module using compact Qwen-2.5 (Yang et al., 2024) models (1.5B/3B parameters) and compare with Auto-RAG using Qwen1.5-32B-Chat. As shown in Fig. 5, performance degradation remains moderate when scaling down models: 3%-6% reduction when halving model size from 9B→3B and 3B→1.5B across all datasets. This controlled performance decline (max 6% across four evaluation metrics) demonstrates our

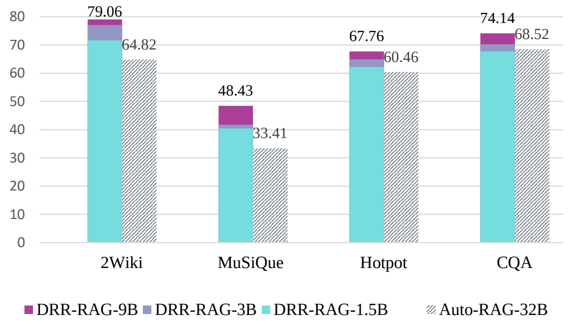

Figure 5: Performance of different size models in QuestionDecompose module.

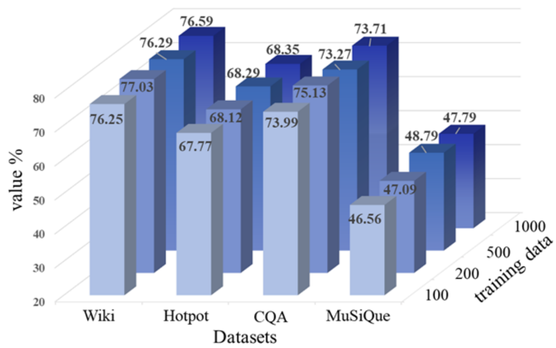

Figure 6: Performance of decompose module under different amounts of training data in four datasets.

framework's practical adaptability, enabling the effective deployment of smaller models in resource-constrained scenarios while maintaining over 90% of the capability of our framework.

5.5 Generalizability in Downstream Tasks

We propose cross-downstream experiments across three datasets, where we train the model on one dataset and evaluate it on the other with EM metric. The results shown in Table 4 indicate that our method has strong generalizability in downstream tasks and even surpasses the model trained on the original data in some cases.

Train/Eval	MuSiQue	Hotpot	CQA
MuSiQue	**46.29**	69.17 (0.48↑)	67.80 (1.10↓)
Hotpot	45.49 (0.80↓)	**68.69**	65.96 (2.94↓)
CQA	46.04 (0.25↓)	69.00 (0.31↑)	**68.90**

Table 4: Cross-Downstream EM results on four datasets, where trained on one dataset evaluated on the other.

5.6 Flexibility and Simplicity in Implementation

To ensure diversity and simplicity in implementation, we explored three approaches for the critical decomposition module. As shown in Table 2, the performance differences among these approaches are minimal, highlighting the module's flexibility.

Regarding implementation complexity, prompt engineering requires only the construction of appropriate prompts with 3 examples (using a 3-shot setting). LoRA fine-tuning follows a similarly straightforward process, as demonstrated in the experiments and supporting theoretical arguments.

Experimental Support: High-quality data is essential for LLMs, but collecting it remains challenging. To assess the data requirements for the decomposition module, we fine-tune the model with varying dataset sizes (from 0.1k to 1k) and evaluate using EM metrics. In Fig. 6, the model performed remarkably well even with as few as 100 samples, with results differing by less than 3% from those obtained with larger datasets. Auto-RAG requires 0.5k samples to achieve stable performance [1], while EfficientRAG needs over 10k data points per training set [2]. Additionally, under the conditions described in Section 4.3, the average training time per dataset is approximately 40 minutes.

Theoretical Support: LLMs acquire knowledge during pretraining and learn desired behaviours through supervised fine-tuning. For question decomposition, the primary goal of the model is to understand the structural patterns of decomposition. The underlying principle is that decomposing multi-hop questions involves recognising structural patterns rather than performing complex reasoning. As a result, extensive training data is not necessary for effective learning.

6 Conclusion

This paper introduced a novel question decomposition framework designed to handle complex, multi-hop questions by breaking them into manageable sub-questions. By leveraging LLMs, the system decomposes multi-hop questions into smaller ones. Throughout the retrieval and resolution process, self-refinement is utilized to verify both the correctness of sub-question answers and the validity of the question decomposition. Abundant experiments have demonstrated the high efficiency of our method, as well as its ease of implementation and the high robustness of the system. These factors collectively illustrate the superiority of our method.

[1] https://arxiv.org/pdf/2411.19443

[2] https://aclanthology.org/2024.emnlp-main.199.pdf

7 Limitations

The following are the limitations associated with our proposed framework: First, The accuracy of the framework relies on the correctness of question decomposition. As shown in Table 2, although our method has made significant progress, it has not surpassed an accuracy of 50% on the MuSiQue dataset, which presents the most complex question structures. There is still room for improvement in question decomposition accuracy across different datasets. Second, to ensure the correct execution of the process, we guide LLMs to output in JSON format. Although LLMs could correct errors in the format, a small number of errors will cause the entire process to throw exceptions. Finally, due to the additional steps introduced by the decomposition process, the framework incurs more processing stages than standard RAG systems, potentially leading to increased response latency. Future work could focus on enhancing the model's handling of even more intricate question dependencies, exploring further optimizations for reducing response delays, and expanding the framework's generalization capabilities.

8 Ethics Statement

In this research, we focus on improving the generation capabilities of LLMs for MHQA. Our proposed method integrates retrieved documents and retrieval information to enhance LLM performance while strictly adhering to ethical guidelines established by the broader academic and open-source community. We ensure transparency by using publicly available datasets and open-source models, such as Wikipedia, for training and evaluation. All data used in this work comes from existing, publicly accessible sources, and we have made every effort to minimize bias and promote fairness. The questions used for evaluation were sourced from established benchmarks, ensuring reproducibility. We acknowledge that the datasets may contain personally identifying or offensive content, but we focus on improving the models rather than filtering such content. No conflicts of interest exist for any of the authors, and we take care to avoid any harm or potential misuse of information in developing this framework.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Simran Arora, Patrick Lewis, Angela Fan, Jacob Kahn, and Christopher Ré. 2023. Reasoning over public and private data in retrieval-based systems. *Transactions of the Association for Computational Linguistics*, 11:902–921.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Self-reflective retrieval augmented generation. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

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*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

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.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *CoRR*.

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*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

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

Xin-Yi Li, Wei-Jun Lei, and Yu-Bin Yang. 2023. From easy to hard: Two-stage selector and reader for multi-hop question answering. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Vaibhav Mavi, Anubhav Jangra, Adam Jatowt, et al. 2024. Multi-hop question answering. *Foundations and Trends® in Information Retrieval*, 17(5):457–586.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

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. *arXiv preprint arXiv:2305.15294*.

Zhili Shen, Chenxin Diao, Pavlos Vougiouklis, Pascual Merita, Shriram Piramanayagam, Damien Graux, Dandan Tu, Zeren Jiang, Ruofei Lai, Yang Ren, et al. 2024. Gear: Graph-enhanced agent for retrieval-augmented generation. *arXiv preprint arXiv:2412.18431*.

Zhengliang Shi, Shuo Zhang, Weiwei Sun, Shen Gao, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. 2024. Generate-then-ground in retrieval-augmented generation for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7339–7353.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200.

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

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9073–9080.

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.

Bohong Wu, Zhuosheng Zhang, and Hai Zhao. 2021. Graph-free multi-hop reading comprehension: A select-to-guide strategy. *CoRR*, abs/2107.11823.

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.

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

Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2023. Search-in-the-chain: Towards accurate, credible and traceable large language models for knowledgeintensive tasks. *CoRR, vol. abs/2304.14732*.

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

Fangkai Yang, Pu Zhao, Zezhong Wang, Lu Wang, Bo Qiao, Jue Zhang, Mohit Garg, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2023. Empower large language model to perform better on industrial domain-specific question answering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 294–312.

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. *arXiv preprint arXiv:1809.09600*.

Fuda Ye, Shuangyin Li, Yongqi Zhang, and Lei Chen. 2024. Rˆ 2ag: Incorporating retrieval information into retrieval augmented generation. *arXiv preprint arXiv:2406.13249*.

Tian Yu, Shaolei Zhang, and Yang Feng. 2024. Auto-rag: Autonomous retrieval-augmented generation for large language models. *arXiv preprint arXiv:2411.19443*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's song in the AI ocean: A survey on hallucination in large language models. *CoRR*, abs/2309.01219.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. Efficientrag: Efficient retriever for multi-hop question answering. *arXiv preprint arXiv:2408.04259*.

A Lora Train details

The LoRA fine-tuning format follows the user-assistant paradigm, as illustrated in the following example in Figure 7.

For fine-tuning, we use the PEFT library with LoRA configuration. The input settings include a maximum input length of 2048 tokens and an output length of 512 tokens. Training parameters are set to a learning rate of 5e-5, a maximum of 10,000 steps, and a batch size of 1. Checkpoints are saved every 1,000 steps, and evaluations are performed with the same frequency using the "predict-with-generate" option. The generation configuration allow up to 512 new tokens. LoRA-specific settings include a rank of 16, a LoRA alpha of 32, a dropout rate of 0.1, and modifications targete at the ["query_key_value"] modules. The convergence of the model is shown in Figure 8.

B Model Deployment Detail

We run all the models on two RTX 4090 GPUs.To ensure consistent experimental conditions, identical hyperparameter configurations are maintained across both models, with the temperature set to 0.1 to ensure response stability by reducing randomness, and a maximum token limit of 8,192 was configured to prevent output truncation in multi-step reasoning tasks. The GLM-4-Flash implementation utilized the official API endpoints provided by the BigModel platform, while GLM-4-9B was deployed via a vLLM-accelerated inference (Kwon et al., 2023) server to optimize computational resource utilization.

C Prompt Template

Lora fine-tuning format example

The user input:

```
You are assigned a multi-hop question decomposition task.
Your mission is to decompose a multi-hop question into a list of single-hop
    sub_questions.
The JSON output must contain the following keys:
- "question": a string, the original multi-hop question.
- "decomposed_questions": a dict of sub_questions and answers. The key should be
     the sub_question number(string format), and each value should be a dict
    containing:
    - "sub_question": a string, the decomposed single-hop sub_question. It MUST
        NOT contain information more than the original question.
    - "answer": a string, the answer of the sub_question.

The origin multi-hop questions is: What nationality is the composer of film
    Tonight For Sure?
```

The assistant input:

```
{
  "question": "What nationality is the composer of film Tonight For Sure?",
  "decomposed_questions": {
    "1": {
      "sub_question": "Who is the composer of film Tonight For Sure?",
      "answer": "{sub_question_1_answer}"
    },
    "2": {
      "sub_question": "What nationality is {sub_question_1_answer}?",
      "answer": "{sub_question_2_answer}"
    }
  }
}
```

Figure 7: Lora fine-tuning format example.

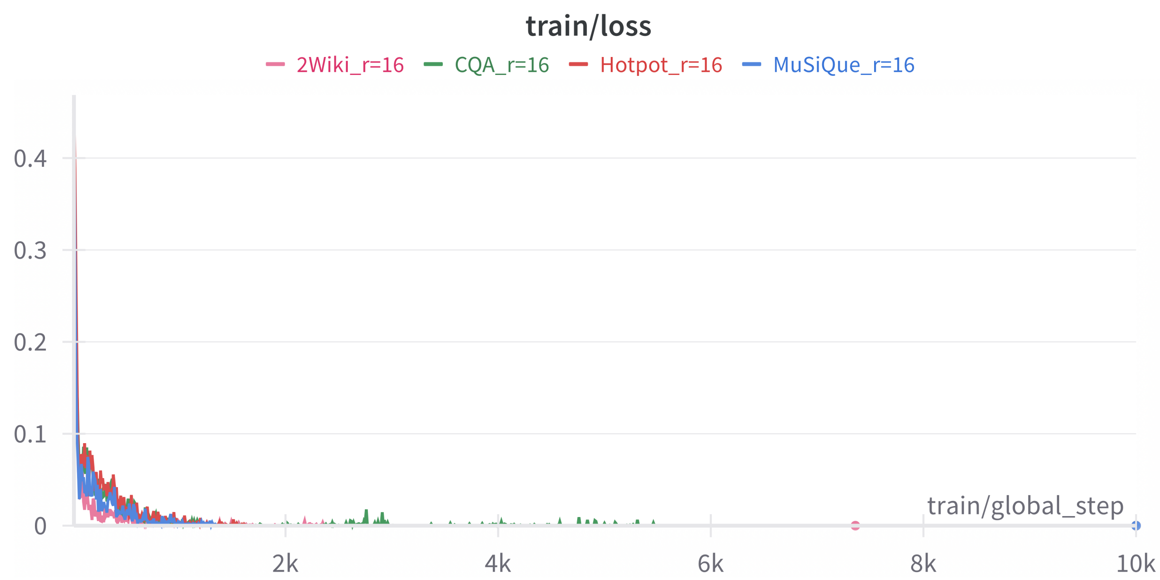

Figure 8: Lora fine-tuning details.

Question Decompose Prompt

```
You are assigned a multi-hop question decomposition task.
Your mission is to decompose a multi-hop question into a list of single-hop
    sub_questions.
The JSON output must contain the following keys:
    - "question": a string, the original multi-hop question.
    - "decomposed_questions": a dict of sub_questions and answers.
The key should be the sub_question number(string format), and each value should
    be a dict containing:
        - "sub_question": a string, the decomposed single-hop sub_question. It
            MUST NOT contain information more than the original question and its
             dependencies. NEVER introduce information from documents.
        - "answer": a string, the answer of the sub_question.


The origin multi-hop questions is: Who is Rhescuporis I (Odrysian)'s paternal
    grandfather?
Your response:
{{
    "question": "Who is Rhescuporis I (Odrysian)'s paternal grandfather?",
    "decomposed_questions": {{
        "1": {{
            "sub_question": "Who is Rhesuporis I (Odrysian)'s father?",
            "answer": "{{sub_question_1_answer}}"
        }},
        "2": {{
            "sub_question": "Who is {{sub_question_1_answer}}'s father?",
            "answer": "{{sub_question_2_answer}}"
        }}
    }}
}}

The origin multi-hop questions is: Do both films The Falcon (Film) and Valentin
    The Good have the directors from the same country?
Your response:
{{
    "question": "Do both films The Falcon (Film) and Valentin The Good have the
        directors from the same country?",
    "decomposed_questions": {{
        "1": {{
            "sub_question": "Who is the director of The Falcon (Film)?",
            "answer": "{{sub_question_1_answer}}"
        }},
        "2": {{
            "sub_question": "Who is the director of Valentin The Good?",
            "answer": "{{sub_question_2_answer}}"
        }},
        "3": {{
            "sub_question": "What is the nationality of {{sub_question_1_answer
                }}?",
            "answer": "{{sub_question_3_answer}}"
        }},
        "4": {{
            "sub_question": "What is the nationality of {{sub_question_2_answer
                }}",
            "answer": "{{sub_question_4_answer}}"
        }},
        "5": {{
            "sub_question": "Are {{sub_question_3_answer}} and {{
                sub_question_4_answer}} the same country?",
            "answer": "{{sub_question_5_answer}}"
        }}
    }}
}}

The origin multi-hop question is: {question}
Your response:
```

Figure 9: Question Decompose Prompt.

Json Format Correct Prompt

```
You are assigned a json correct task.
Your mission is to correct the wrong json format into right format.
The your right json output must contain the following keys:
    - "question": a string, the original multi-hop question.
    - "decomposed_questions": a dict of sub_questions and answers. The key
should be the sub_question number(string format), and each value should be a
    dict containing:
        - "sub_question": a string, the decomposed single-hop sub_question.
It MUST NOT contain information more than the original question and its
    dependencies. NEVER introduce information from documents.
        - "answer": a string, the answer of the sub_question.
wrong json format will be embraced by <wrong_json> and </wrong_json> tags.
There are some examples for you to refer to:
<wrong_json>
{{
    "question": "What year was home brewing first allowed in the country where
        Prince of Thieves, who titular character John is depicted alongside, was
         made?",
    "decomposed_questions": {{
        "1": {{
            "sub_question": "In which country was the film Prince of Thieves
                made?",
            "             "answer": "{{sub_question_1_answer}}"
        }},
        "2": {{
            "sub_question": "What is the country where home brewing was first
                allowed?",
            "answer": "{{sub_question_2_answer}}"
        }},
        "3": {{
            "sub_question": "In what year was home brewing first allowed in {{
                sub_question_2_answer}}?",
            "answer": "{{sub_question_3_answer}}"
        }}
    }}
}}
</wrong_json>
Your response:
{{
    "question": "What year was home brewing first allowed in the country where
        Prince of Thieves, who titular character John is depicted alongside, was
         made?",
    "decomposed_questions": {{
        "1": {{
            "sub_question": "In which country was the film Prince of Thieves
                made?",
            "answer": "{{sub_question_1_answer}}"
        }},
        "2": {{
            "sub_question": "What is the country where home brewing was first
                allowed?",
            "answer": "{{sub_question_2_answer}}"
        }},
        "3": {{
            "sub_question": "In what year was home brewing first allowed in {{
                sub_question_2_answer}}?",
            "answer": "{{sub_question_3_answer}}"
        }}
    }}
}}

Now your wrong_json information are as follows.
<wrong_json>
{wrong_json}
</wrong_json>
Your response:
```

Figure 10: Json Format Correct Prompt.

Self-Refine Prompt

```
You are assigned a question-answer correctness Judge task.
Your task is to judge whether the answers are correct based on the documents and
     question.
the documents is wrap by <documents> and </documents>
Your answer only needs to contain the following format. No other content is
    required.
    - "evidence": the text that supports answering the question.Set to null if
        documents and question have no correlation
    - "correctness": a string, either "right" or "wrong", determines whether the
         answer is correct based on evidence.
    - "correct_answer": give the answer you think is correct based on evidence
        if you think the origin answer is wrong; leave it as the origin answer
        if you think the origin answer is correct;

There are some examples for you to refer to:
The question is: Who was married to Valerie Hobson?
The answer is: {{answer_example}}
<documents>
{{documents_example}}
</documents>
your response:
{{respose_example}}

Now, your question and documents are as follows.
<Question>: {question}
<Answer>: {answer}
<documents>: {documents}
</documents>
Your response:
```

Figure 11: Self-Refine Prompt.

Final Answer Inference Prompt

````
You are assigned a multi-hop question task.
Your mission is to use a list of single-hop sub_questions and their answers to
    answer the multi-hop question task.
Your answer should be after <Answer> in JSON format with key "answer" and its
    value should be string.
Your <Answer> must be wrapped by ```json and ```.
The given sub_questions and answers will be embraced by <
    sub_questions_and_answers> and </sub_questions_and_answers> tags. You can
    refer to the sub_questions_and_answers to infer the answer of the question.
    If can't infer the answer, answer "i don't know" directly.

There are some examples for you to refer to:
<sub_questions_and_answers>
"decomposed_questions":{{
    "1": {{
        "sub_question": "Who is the performer of song Soldier (Neil Young Song)
            ?",
        "answer": "Neil Young"
    }},
    "2": {{
        "sub_question": "Which country is Neil Young from?",
        "answer": "Canadian"
    }}
}}
</sub_questions_and_answers>
<Question>: Which country the performer of song Soldier (Neil Young Song) is
    from?
<Answer>:
```json
{{"answer": "Canada"}}
```

<sub_questions_and_answers>
"decomposed_questions": {{
        "1": {{
            "sub_question": "Who is the director of Charge It To Me",
            "answer": "Roy William Neill"
        }},
        "2": {{
            "sub_question": "Who is the director of Danger: Diabolik",
            "answer": "Mario Bava"
        }},
        "3": {{
            "sub_question": "When was Roy William Neill born?",
            "answer": "4 September 1887"
        }},
        "4": {{
            "sub_question": "When was Mario Bava born?",
            "answer": "31 July 1914"
        }}
        ,
        "5": {{
            "sub_question": "Which of the two times 4 September 1887 and 31 July
                1914 is earlier?",
            "answer": "4 September 1887"
        }}
}}
</sub_questions_and_answers>
<Question>: Which film whose director is younger, Charge It To Me or Danger:
    Diabolik?
<Answer>:
```json
{{"answer": "Charge It To Me"}}
```

Now your question and sub_questions_and_answers are as follows.
<sub_questions_and_answers>
{sub_questions_and_answers}
</sub_questions_and_answers>
<Question>: {question}
<Answer>:
````


Figure 12: Final Answer Inference Prompt.