Talk to Right Specialists: Iterative Routing in Multi-agent System for Question Answering

Anonymous ACL submission

Abstract

Large language model (LLM)-based applications have increasingly leveraged retrieval-augmented generation (RAG) techniques to provide reliable responses, particularly for queries demanding knowledge of private domains. Practical constraints, such as data sovereignty regulations, can hinder the centralized aggregation of private knowledge. This can create challenges in situations where (1) a user comes with a question but has no idea which applications have the related knowledge to answer, or (2) the question requires cross-domain knowledge to answer.

In this work, we abstract each RAG application with private knowledge as an RAG agent. We propose RIRS, an iterative routing framework with an efficient and accurate routing mechanism and an iterative refining-solving mechanism to orchestrate multiple RAG agents with private knowledge bases. The server routes queries to the most relevant agents by identifying the most related knowledge clusters by similarities in a vector space. For complicated questions, the server can iteratively aggregate responses to derive intermediate results and refine the question to bridge the gap toward a comprehensive answer. Extensive experiments demonstrate the effectiveness of RIRS, including how our routing algorithm precisely selects the agents and provides accurate responses to single-hop queries and how an iterative strategy achieves accurate, multi-step resolutions for complex queries.

1 Introduction

Large language models (LLMs) have revolutionized natural language processing (NLP) by demonstrating superior performance in question-answering (QA) tasks, often surpassing traditional systems in both accuracy and contextual understanding. Based on LLMs, retrieval-augmented generation (RAG) is a technique to integrate external knowledge sources, extracting the most relevant information for any input query to enable LLMs to answer questions beyond their training data and reduce their hallucination (Wu et al., 2024; Asai et al., 2023a; Lewis et al., 2020; Jiang et al., 2023; Izacard and Grave, 2020; Mallen et al., 2022; Kasai et al., 2024; Xiong et al., 2024). To further improve retrieval and response quality, many RAG applications are built in the form of agents (referred to as RAG agents in this paper) (Weng, 2023; Roucher, 2024; Joshi et al., 2024). However, because RAG agents' reliability is limited to the domain of their knowledge sources, there are two major inconveniences when serving users. (1) The domain and the boundary of the knowledge source are difficult to clearly define and usually unknown to users, so manual attempts with different agents may be required to obtain reliable answers. (2) Answering some questions may require cross-domain knowledge from different RAG agents.

One straightforward solution to overcome the inconveniences is to build a unified, large-scale knowledge repository by collecting all knowledge from worldwide existing RAG agents, as illustrated in Figure 1a. A unified RAG agent could be built with the repository to handle queries spanning multiple domains by pooling all available information. However, it is confronted with two critical limitations. Firstly, the computational complexity and scalability issues inherent in managing such a vast and diverse knowledge base could undermine the system's overall performance and responsiveness (Fan et al., 2024; Asai et al., 2023a). Secondly, it is impractical or even infeasible to invade knowledge sovereignty and construct a centralized knowledge repository, especially when the knowledge is intellectually protected or sensitive. For example, due to privacy concerns, ophthalmology hospital data derived from internal medical records is unavailable for merging.

An alternative is to deploy a distributed multi-agent system, as illustrated in Figure 1b, which con-

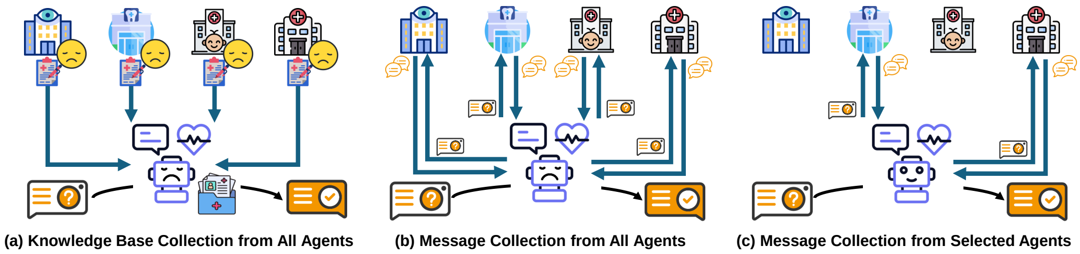


Figure 1: Collaboration Strategies of Multi-agent System for QA.

sists of a central server and multiple RAG agents. The server forwards a user's query to *all* available RAG agents and subsequently aggregates their responses. In this framework, each agent retains its local, domain-specific knowledge base, ensuring that its knowledge sovereignty is well protected and that its data are not merged into a centralized repository. This approach eliminates the need for a massive, unified knowledge base. However, routing every query to *all* agents incurs redundant computational overhead and cost, as not every agent is equipped to address every query. Consequently, it is worth considering building a distributed multi-agent system with *a routing mechanism* that selects the most appropriate agents in response to a query, as illustrated in Figure 1c.

In this work, we propose RIRS, a framework with a Routing and Iterative Refining-Solving mechanism designed to effectively reconcile multiple RAG agents. For the routing mechanism, each agent partitions its local knowledge base into disjoint clusters, and the central server collects these clustered knowledge representations. When a user query arrives, the server computes its similarity to the collected clusters and forwards the query only to those agents whose knowledge clusters are most relevant. This training-free mechanism does not require additional training and is inherently privacy-preserving, and raw knowledge remains confined within the individual agents. Moreover, our method employs an iterative refining-solving strategy to handle complex queries that involve multiple reasoning steps or span across different domains. Specifically, the server sequentially routes a query to the appropriate agents and, after each response, simplifies the query by removing the addressed portion.

Contributions. The major contributions of this work are listed as follows:

- To the best of our knowledge, this is the first work that considers knowledge sovereignty issues under multi-agent, enabling efficient collaboration across various specialized agents without collecting unnecessary information.
- We introduce RIRS, a training-free iterative routing framework that selects the most proper agents and collaborates on complex user queries.
- Our experimental results demonstrate the effectiveness of RIRS in handling various queries: The routing mechanism enables precise agent selections, while the iterative refining-solving mechanism achieves accurate, multi-step resolutions for complex queries.

2 Related Works

Due to space limits, this section briefly discusses the key differences and contributions compared to the most related works. A detailed literature review is provided in Appendix A.

Retrieval Augmented Generation (RAG). RAG methods focus on integrating external knowledge sources into LLMs to generate accurate and reliable responses. *RankRAG* (Yu et al., 2024), for example, ranks the knowledge pieces and generates a response with genuinely important ones. To handle a complex query that involves multiple knowledge pieces, *EfficientRAG* (Zhuang et al., 2024) iteratively simplifies the query by removing the portion with relevant knowledge until it is resolved. While these methods primarily focus on single-agent retrieval frameworks, our work differentiates itself by addressing the challenge of coordinating knowledge retrieval across multiple agents, thereby enabling a more comprehensive handling of diverse and cross-domain queries.

Routing Mechanism in Multi-agent Systems. LLM-based multi-agent systems leverage the collective intelligence and specialized capabilities of multiple expert agents to collaboratively tackle complex problems. In such systems, a robust routing mechanism is essential to direct each query to the most appropriate agent based on its unique expertise. Recent works include Chameleon (Lu et al., 2024), which relies on detailed textual descriptions of agent capabilities for routing decisions; RouterDC (Chen et al., 2024a), employing a lightweight model for dynamic query distribution among a fixed agent pool. In contrast, our approach accurately characterizes each agent's knowledge capacity based on their own data, thereby eliminating the need for exhaustive textual descriptions and labor-intensive dataset labeling.

3 Multi-agent Framework

Preliminary: RAG Agent. An RAG agent is an advanced application that integrates retrieval mechanisms with LLMs to deliver accurate and contextually rich responses. Specifically, an RAG agent processes a question through three steps: (i) *Knowledge Retrieval:* It extracts relevant knowledge pieces using both sparse and dense retrievers from external sources (Robertson et al., 2004; Izacard et al., 2021; Xu et al., 2023); (ii) *Reranking:* It filters out unhelpful or misleading information, allowing the generation model to focus on the most pertinent content (Yu et al., 2024); (iii) *Response Generation:* It combines the question and the pertinent external knowledge to produce an informed response with its backbone LLM.

Motivations and Problem Statement. To protect knowledge sovereignty and harness domain-specialized expertise, we propose a distributed multi-agent framework. In this framework, each RAG agent holds its own specialized knowledge base, while a central server coordinates query processing. Upon receiving an input query, the server routes it to all agents and aggregates their responses into a final answer, as described in Figure 1b. However, this standard operating procedure for QA tasks faces two major challenges:

• ***(i) Irrelevant Agent Involvement:*** When a query falls outside an agent's domain expertise, its participation not only introduces unnecessary computational and communication overhead but may also generate misleading information.

• ***(ii) Incomplete Knowledge Fusion:*** A query that spans multiple domains requires seamless integration of responses from various agents and likely from multiple reasoning steps. Without effective coordination, the final answer could be incomplete.

These challenges highlight the need for *an intelligent routing mechanism* that strategically directs queries to the most relevant agents and aggregates their responses wisely by filtering out irrelevant knowledge. Such a mechanism is critical to reducing redundant processing, enhancing scalability, ensuring accurate, holistic query resolution, and maintaining data privacy in a decentralized setting.

4 RIRS

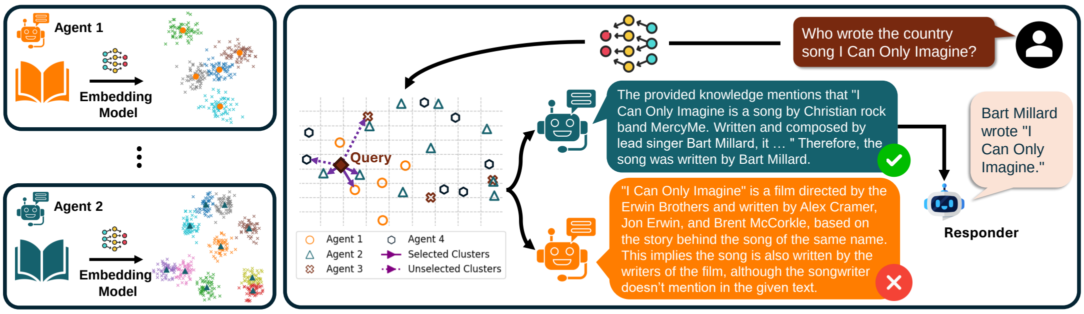
To address the challenges, we propose an intelligent routing mechanism to deal with both single-hop and multi-hop questions. Section 4.1 outlines the design of a query routing algorithm, where the server routes a query to a subset of agents according to their knowledge coverage represented in a vector space. Section 4.2 extends our design to address more challenging queries that require multi-round or cross-agent knowledge.

4.1 Routing Algorithm

The primary objectives of our routing algorithm are twofold: high quality of the final answer and high efficiency in terms of both minimum additional latency of routing and minimum token consumption. To accomplish these objectives, an ideal server must possess three key abilities: (i) accurately assessing each agent's knowledge capabilities, (ii) selecting a necessary and sufficient subset of agents to reduce computational and communication costs, and (iii) verifying the validity of each response, encompassing both the analysis and the final answer. Notably, the second capability is closely linked to the server's understanding of the knowledge boundaries inherent to each agent.

To satisfy the requirements, we design a routing mechanism as shown in Figure 2, which consists of two primary stages: (a) knowledge clustering and (b) the query forwarding workflow. In the knowledge clustering stage (Figure 2a), each agent encodes its local knowledge (e.g., text chunks) into vectors using an identical embedding model, then partitions the knowledge into disjoint clusters and generates a representation for each cluster with the embeddings. The central server then collects these clustered knowledge representations and uses them to determine which agents are most relevant to an incoming query. As shown in Figure 2b, the server forwards the query to those agents whose clusters exhibit the highest similarity to the query. The selected RAG agents subsequently process the query based on their own knowledge, and the server aggregates and evaluates their responses. Finally, the server synthesizes the final response to the user. More details are as follows.

Knowledge Clustering. Suppose an RAG agent contains m distinct knowledge pieces, represented as $e_1, \ldots, e_m$ in a vector space. To evaluate the agent's knowledgeability, we partition the knowledge pieces into n disjoint clusters, $c_1, \ldots, c_n$. The goal is to ensure the nodes' similarities within a cluster are as large as possible, while the nodes from different clusters should be less similar. Toward the goal, we formulate an objective function to obtain the satisfied n clusters:



(a) Knowledge Clustering (b) Query Forwarding

Figure 2: Routing Mechanism

$$\arg \max_{\substack{c_1,\ldots,c_n \\ \bigcup_{i\in[n]}\{c_i\} \\ =\bigcup_{j\in[m]}\{e_j\}}} \; \min_{\substack{e_a,e_b\in c_k, \\ k\in\{1,\ldots,n\}, \\ e_a\neq e_b}} \mathsf{sim}(e_a,e_b) \qquad (1)$$

Here, $\mathsf{sim}(e_a, e_b)$ denotes the cosine similarity between two knowledge pieces e_a and e_b. By maximizing the minimum similarity within a cluster, we ensure that knowledge pieces within the same cluster are as similar as possible, leading to more informative cluster representations.

The RAG agent follows a four-step process to solve this clustering objective and report its knowledgeability to the server:

• ***Step 1: Compute Embeddings.*** An RAG agent computes the embeddings for their own knowledge, which can be reused later in the knowledge retrieval of a dense method to an input query (Izacard and Grave, 2020).

• ***Step 2: Knowledge Clustering.*** Using hierarchical clustering, the agent partitions the m knowledge pieces into n disjoint clusters. The distance between any two knowledge pieces is measured by their embeddings' cosine similarity.

• ***Step 3: Calculate Cluster Representations.*** For each cluster, the agent calculates a centroid by averaging the embeddings of all knowledge pieces within that cluster. This centroid serves as a representative summary of the cluster.

• ***Step 4: Push Representations to Server.*** The agent sends the centroids of all local clusters to the server, who utilizes this information to make routing decisions.

Choice on the number of clusters. Since different RAG agents hold varying amounts of knowledge, the number of clusters n should not be constant across agents. Intuitively, agents with more knowledge pieces may have overlapping or redundant knowledge, while agents with fewer pieces might specialize in sparse, distinct knowledge domains. To account for this, we set $n = \lfloor \sqrt{m} \rfloor$, aligning with the hypothesis that a larger number of knowledge pieces should correspond to more clusters while maintaining manageable granularity. In fact, the choice of n has proved its effectiveness in the field of inverse file indexing in practice.

Discussion: Update of Agent Knowledge. Throughout the paper, the proposed RIRS builds upon an implicit assumption that each agent's knowledge does not update frequently. Nevertheless, agents may revise or expand their knowledge bases after the initial clustering, which in turn alters the corresponding cluster representations. To accommodate these continual edits, online methods (Cohen-Addad et al., 2021; Monath et al., 2023) could be the best solutions because they support the updating of each cluster and its representation on-the-fly.

Query Forwarding. Once the server has gathered the knowledgeability of all RAG agents, it must effectively coordinate the agents to handle user queries. This involves selecting the most suitable agents based on the similarity between the query and the centroids provided by each agent.

Let us define $\bar{c}_j^{(i)}$ as the j-th centroid of RAG agent i, and define a function $f(\cdot)$ such that $f\left(\bar{c}_j^{(i)}\right) = i$. This function maps a centroid to the corresponding agent. Suppose there are M RAG agents in the multi-agent system. For each agent $i \in \{1, \ldots, M\}$, there are n_i centroids, denoted by the set $\left\{\bar{c}_j^{(i)}\right\}_{j=1}^{n_i}$. Let x be the embedding of a query. The goal is to identify k clusters whose centroids have the highest similarity scores with the query embedding. This can be formulated as:

$$\{\bar{c}_j\}_{j=1}^{k} \triangleq \arg \mathop{\mathsf{Top}k}_{\bar{c}\in\bigcup_{i\in[M]}\left\{\bar{c}_j^{(i)}\right\}_{j=1}^{n_i}} \mathsf{sim}(x, \bar{c}) \qquad (2)$$

The agents corresponding to the centroids in the set $\{f(\bar{c}_j)\}_{j=1}^{k}$ are then invited to answer the query.

Therefore, we define a routing-then-answer function RTANS(QUERY), which the server calls to proceed through the following steps to generate the final response and ensure the response is accurate and well-supported to the input query:

• ***Step 1: Agent Selection.*** The server selects the most relevant agents based on the similarity between the query and the centroids, as described.

• ***Step 2: Response Generation.*** The selected RAG agents generate responses that include a piece of supporting evidence and an answer.

• ***Step 3: Evaluation of Responses.*** The server collects the responses and evaluates them based on the quality of the analysis provided. It categorizes the answers as "Addressed" or "Not Addressed."

• ***Step 4: Final Answer Curation.*** The server utilizes the "Addressed" answers and finalizes the response to the user.

4.2 Iterative Refining-solving

A multi-hop question inherently requires multiple reasoning steps, with each step drawing on distinct pieces of supporting knowledge (Welbl et al., 2018). In our setting, such queries are especially challenging because the required information can be distributed among different agents. The single-step routing process, which relies solely on an initial similarity match to select agents, may be inadequate for handling multi-hop questions because it cannot effectively identify or integrate the sequential pieces of evidence required for a complete answer. Thus, a more robust solution is needed to address complex and often cross-domain queries.

Iterative Routing. Previous works (Zhuang et al., 2024; Yang et al., 2024b; Press et al., 2022; Jiang et al., 2023) introduce a simple yet effective solution to handle a multi-hop question by repeatedly generating new queries based on the extracted knowledge. Inspired by that, we propose an iterative routing process to manage multi-hop questions. In detail, we repeat the following two steps until a terminate signal $\boxed{\textbf{terminate}}$ is raised or the iteration reaches a given limit. Appendix B provides a more detailed explanation of the design principle.

Step 1: Invoking RtAns(Query). The RtAns function, defined in Section 4.1, is initially designed for single-hop questions. For complex queries, we modify its output into a dictionary containing three response types: "fully addressed," "partially addressed," and "not addressed." A "partially addressed" response indicates that while some aspects of the question have been answered, a final, complete answer has not yet been reached.

Step 2: Answer Generation or Problem Refinement. Based on the output from RtAns(Query), we handle the response in one of three ways:

• ***Fully Addressed:*** If one or more responses are marked as "fully addressed," the system immediately compiles these responses to generate a final, comprehensive answer to the user's query and raises $\boxed{\textbf{terminate}}$ afterwards. In this scenario, all necessary information and supporting evidence are available, so no further refinement is required, ensuring efficient processing and prompt delivery of the final answer.

• ***Partially Addressed:*** If no responses are fully addressed but some are marked as "partially addressed," the system evaluates whether the current information is sufficient to resolve the query. If so, the server will raise $\boxed{\textbf{terminate}}$ and compile for a final answer. Otherwise, the server simplifies the question as a new query and repeats these two steps with the new query.

• ***Not Addressed:*** If all responses are marked as "not addressed," the query is flagged as unanswerable, and the server raises $\boxed{\textbf{terminate}}$. The system then informs the user and, if applicable, provides any partial insights that might be derived from the existing responses.

Advantages. This proposed method appears to have twofold advantages from the perspective of *efficiency* and *adaptiveness*. For multi-hop questions that require sequential reasoning, our approach dynamically refines the query based on the acquired knowledge from agents. Compared to those question decomposition methods (Zhou et al., 2022; Verma et al., 2024; Chan et al., 2024), the proposed RIRS reduces unnecessary query rounds and allows an agent to address multiple reasoning steps within its expertise. For those questions requiring parallel reasoning, particularly across diverse domains, the server enables multiple specialized agents to work concurrently, each leveraging its smaller, more efficient knowledge base, which significantly accelerates the overall inference process.

5 Experiments

5.1 Experimental Setup

Due to space limits, this subsection provides a brief overview of the experimental setup to interpret the quantitative results presented in the main body. Full details are provided in Appendix C.

Datasets. Our experiments cover both single-hop and multi-hop open-ended QA tasks within a unified evaluation framework. We employ the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019) for single-hop QA to emphasize the effectiveness of the routing mechanism. For evaluating the performance on answering complex questions, we assess performance on four multi-hop QA benchmarks: HotpotQA (Yang et al., 2018), 2WikiMultiQA (Ho et al., 2020), MusiQue (Trivedi et al., 2022b), and Multi-Hop RAG (MHR) Benchmark (Tang and Yang, 2024). Except for MHR, all

# Sel. Clusters	Avg Sel. Agents (Eff. Rate)	Agent-Level Ans. Rate	Avg Viewed Doc. (Eff. Rate)	Doc.-Level Ans. Rate	LLaMA-3.1-8B		Qwen-Plus	
					LM	Acc.	LM	Acc.
1	1.0 (52.69%)	52.69%	3.32 (21.43%)	50.58%	43.53%	60.91%	59.90%	77.72%
5	3.32 (37.47%)	76.47%	10.12 (11.61%)	73.01%	50.98%	63.96%	62.05%	80.05%
10	5.28 (26.76%)	84.52%	16.95 (7.07%)	80.12%	50.18%	64.74%	62.90%	81.05%

Table 1: Performance under different number of selected clusters on the Natural Questions task

questions in these datasets are answerable using knowledge from Wikipedia.

Models. Our experimental setup employs a suite of models to support both retrieval and generation tasks. Specifically, we use `text-embedding-v2` (Zhang et al., 2024) as our embedding model to generate dense representations for effective text retrieval. The agents are powered by two large language models, i.e., `llama-3.1-8b-instruct` (Touvron et al., 2023a,b; Dubey et al., 2024) and `qwen-plus-2024-12-20` (Yang et al., 2024a).

Baselines. Throughout the experiments, we compare the proposed RIRS with several baseline methods. *RankRAG* and *EfficientRAG* represent approaches in single-RAG-agent systems, where all knowledge is centralized within a single agent. In contrast, *Chameleon* and *RouterDC* operate in a multi-RAG-agent system and serve as alternative routing strategies. Specifically, *Chameleon* maintains a description of each agent and prompts an LLM to identify the most relevant agents for a given input query. *RouterDC*, on the other hand, associates each agent with a list of answerable questions and selects candidate agents by retrieving those whose past questions are most similar to the current query. Moreover, *GoldRouter* assumes prior knowledge of the optimal agent(s) for each query, thereby eliminating routing uncertainty and serving as an upper bound in the multi-agent scenario. Further details on these baselines are provided in Appendix C.4.

Multi-RAG-Agent Systems. Throughout the experiments, we construct two multi-RAG-agent systems tailored to different datasets. (i) *WikiAgents* for Wikipedia-related QA assigns each agent a specialized domain of knowledge and includes a knowledgeable agent in case that none of the RAG agents can answer an input query. (ii) *NewsAgents* for the MHR dataset includes two variants: *NewsAgent-Source* and *NewsAgent-Domain*, which partition the same knowledge base based on news sources and news topics, respectively. Detailed configurations of both systems are provided in Appendix C.3.

Evaluation Metrics. Our experiments measure the effectiveness of the proposed RIRS by comparing the baselines for the following metrics:

	LLaMA-3.1-8B	Qwen-Plus
Evaluator Correctness	45.60%	79.09%
Fully Correct	16.48%	54.03%
All Addressed Correct	16.85%	59.98%

Table 2: Response Evaluator Performance

- ***Lexical Match (LM):*** This metric measures the percentage of questions where the groundtruth answer appears within the generated response. Since the LLM-generated answers may be longer than the groundtruth, we focus on whether the groundtruth is included in the predicted answer rather than requiring an exact match.
- ***GPT Evaluation (Acc.):*** We adopt *gpt-4o-2024-08-06* (Achiam et al., 2023) to evaluate the correctness of the generated responses by comparing the generated responses against designated groundtruth answers. This metric captures cases where the generated response conveys the same meaning as the groundtruth, even if the wording is different.
- ***Time:*** We compute the wall-clock time from the question that appears to be a valid response. It is noted that the time may not be accurately measured because of the existence of network or threading congestion, especially if an API call is required.

5.2 Ablation Studies

Choice of TopK. Table 1 indicate that increasing the number of selected clusters improves both the agent-level and document-level answerable rates, which in turn boosts overall performance. Besides, the reason why test accuracy is greater than the answerable rate is attributed to the LLM's ability in question answering. When the agents are considered unable to answer the question, a CoT agent is invoked to give a response upon its own ability. Moreover, the phenomenon that document-level answerable rates are always lower than agent-level answerable rates reflects that an agent may not always correctly retrieve the most relevant documents from its local knowledge base.

Evaluation Capacities of Different Models. In this experiment, we first collect the original questions, the responses from the selected agents, and the evaluator's judgments. We then have GPT-4o annotate the correctness of the selected agents' responses. As the proposed method does not take an extra step to fine-tune the LLMs for evaluation, the experiment demonstrates LLMs' capability to follow instructions and evaluate the correctness

Methods	Models	Natural Questions			HotpotQA					2WikiMultiHopQA					MuSiQue				
		LM	Acc.	Time	LM	F1	Acc.	Rounds	Time	LM	F1	Acc.	Rounds	Time	LM	F1	Acc.	Rounds	Time
Without RAG																			
CoT	Qwen-Plus	52.99	74.55	2.94	39.68	43.82	54.58	1.00	3.06	42.38	44.89	37.31	1.00	3.25	16.76	20.66	33.86	1.00	3.32
	LLaMA-3.1-8B	44.41	58.60	1.64	30.96	36.02	44.23	1.00	3.07	31.40	34.46	25.99	1.00	1.55	13.30	17.76	21.86	1.00	2.90
Single-RAG-Agent System																			
RankRAG	Qwen-Plus	62.73	82.61	14.33	51.49	55.78	73.63	1.00	10.82	51.91	54.06	60.10	1.00	11.39	21.37	25.08	40.73	1.00	12.04
	LLaMA-3.1-8B	56.48	70.10	7.42	38.17	42.56	53.21	1.00	5.23	32.49	34.84	36.69	1.00	5.85	15.37	19.41	27.63	1.00	6.33
EfficientRAG[3]	Qwen-Plus	62.73	82.61	14.33	57.41	63.01	76.55	1.52	22.69	70.41	72.77	75.85	1.81	25.71	26.62	31.41	46.42	1.88	24.28
	LLaMA-3.1-8B	56.48	70.10	7.42	47.38	52.96	61.17	2.48	14.67	48.45	52.04	51.13	2.33	14.69	21.38	26.22	29.13	2.47	17.13
Multi-RAG-Agent System																			
GoldRouter[1]	Qwen-Plus	63.41	83.02	12.97	58.17	63.99	81.01	1.22	22.85	70.11	72.37	76.33	1.36	30.93	29.01	33.75	47.55	1.51	29.87
	LLaMA-3.1-8B	55.13	69.70	6.29	50.60	56.26	64.94	1.55	17.07	49.73	52.99	52.07	1.58	20.57	23.27	28.18	31.46	1.86	21.68
Chameleon	Qwen-Plus	55.48	73.92	24.90	52.12	57.55	73.19	1.43	39.05	56.68	59.18	60.12	1.70	37.53	**24.21**	**28.88**	39.87	1.77	48.46
	LLaMA-3.1-8B	49.68	62.73	15.27	41.08	46.59	50.20	1.85	24.77	40.52	43.49	40.27	1.84	26.71	15.69	20.11	23.74	2.08	32.81
RouterDC	Qwen-Plus	53.93	71.09	24.91	51.34	56.84	67.24	1.46	44.12	50.97	53.88	52.12	1.67	45.61	23.38	27.69	37.43	1.71	52.98
	LLaMA-3.1-8B	47.26	60.43	13.11	38.31	43.11	47.74	1.78	23.63	34.48	37.35	30.59	1.67	26.58	15.08	19.11	21.58	1.98	29.66
RIRS	Qwen-Plus	**62.05**	**80.05**	23.30	**54.01**	**59.62**	**75.63**	1.44	34.71	**61.14**	**63.71**	**65.78**	1.73	39.76	23.81	28.63	**41.24**	1.67	43.98
	LLaMA-3.1-8B	<u>50.98</u>	<u>63.96</u>	10.51	<u>47.57</u>	<u>52.56</u>	<u>58.69</u>	2.04	19.93	<u>45.65</u>	<u>49.10</u>	<u>45.65</u>	2.02	22.26	<u>20.16</u>	<u>24.22</u>	<u>25.82</u>	2.48	25.14
	Mixed[2]	55.98	73.32	20.98	51.68	56.65	69.33	1.83	41.04	59.48	61.94	62.06	1.85	39.37	24.50	28.57	39.75	2.08	47.51

[1] The inference is completely based on the selected agents.
[2] The RAG agents use LLaMA-3.1-8B, while other modules in the server use Qwen-Plus.
[3] Under the Natural Questions dataset, which fixes to a single routing step, EfficientRAG is equivalent to RankRAG.

Table 3: Performance comparison of different methods under various datasets and the knowledge of WikiAgents. The best results under Qwen-Plus and LLaMA-3.1-8B are **Bold** and <u>underline</u> out, respectively, among the methods to multiple RAG agents (i.e., Chameleon, RouterDC, and RIRS), which are highlighted in the grey background.

of generated responses based on natural language understanding. As shown in Table 2, Qwen-Plus significantly outperforms LLaMA-3.1-8B in zero-shot response evaluation, meaning that Qwen-Plus is more likely to avoid error propagation and identify useful information. The importance of the response evaluator will be discussed in Section 5.3.

5.3 Analysis with Wikipedia-related QA

Table 3 presents the performance of our proposed RIRS alongside various baselines on four Wikipedia-related QA tasks using the knowledge from the WikiAgents. In this section, we highlight key comparisons and insights based on the acquired experimental results.

Comparison with the scenario of Single RAG agent. A single-RAG-agent system consolidates all knowledge into a unified repository, allowing RAG methods such as RankRAG and EfficientRAG to retrieve from a comprehensive document set for each query. In expectation, RankRAG is better suited for single-hop QA tasks, while EfficientRAG excels at generating accurate responses for multi-hop queries. As shown in Table 3, EfficientRAG consistently outperforms RIRS, since single-RAG-agent systems are immune to the routing errors inherent in multi-RAG-agent systems. However, when routing errors are eliminated in a multi-RAG-agent system, as with GoldRouter, performance consistently surpasses that of RankRAG and EfficientRAG. This improvement stems from the partitioning of knowledge into domain-specific agents, which reduces the retrieval of irrelevant or distracting content and allows each agent to focus on a smaller, more relevant subset of documents.

Comparison with GoldRouter. As an error-free routing mechanism, GoldRouter serves as the upper bound in a multi-agent system, achieving the highest scores in LM, F1, and Accuracy, as well as the lowest values in rounds and response time. In contrast, all other methods (Chameleon, RouterDC, and RIRS) experience varying degrees of performance degradation. i.e., more query rounds but with lower accuracy. The degradation is owed to routing errors, highlighting the importance of designing an effective routing mechanism and underscoring the inherent challenge of dynamically estimating each agent's expertise on-the-fly.

Comparison within multi-agent routing methods. From Table 3, the accuracy ranking among the three routing methods consistently follows: RIRS > Chameleon > RouterDC, under the same dataset and backbone LLM. The dominance of the proposed RIRS would be more obvious for complex questions and when using a weaker LLM (e.g., LLaMA-3.1-8B), with up to 8% and 15% higher accuracy than Chameleon and RouterDC, respectively. These results underscore the reliability of RIRS in identifying agents' knowledge boundaries. Specifically, RIRS constructs rich cluster representations to model each agent's knowledge capacity, enabling more accurate routing. In contrast, RouterDC relies on a fixed set of historically answerable questions, which fails to fully capture an agent's expertise, and Chameleon depends on static textual descriptions that are often insufficient, leading to misrouting. Although RIRS incurs slightly more query rounds than the first runner-up Chameleon, it spends less time because it avoids additional LLM calls for agent selection.

Comparison with various LLMs. In the proposed RIRS, we observe that more powerful LLMs, such as Qwen-Plus, significantly outperform smaller models like LLaMA-3.1-8B. Furthermore, adopting a mixed-model setting, where

Methods	Models	NewsAgent-Source			NewsAgent-Domain		
		Acc.	Hall.[1]	Rounds	Acc.	Hall.[1]	Rounds
Single-RAG-Agent System [2]							
RankRAG	Qwen-Plus	78.52	3.37	1.0	-	-	-
	LLaMA-3.1-8B	65.77	8.71	1.0	-	-	-
EfficientRAG	Qwen-Plus	81.80	1.19	1.57	-	-	-
	LLaMA-3.1-8B	66.80	6.05	1.32	-	-	-
Multi-RAG-Agent System							
GoldRouter	Qwen-Plus	89.32	-	1.06	90.31	-	1.12
	LLaMA-3.1-8B	79.53	-	1.08	80.16	-	1.18
RIRS	Qwen-Plus	84.75	0.93	1.65	88.78	1.07	1.66
	LLaMA-3.1-8B	71.91	4.91	1.37	75.12	4.73	1.39

[1] Hall. means a hallucination rate, indicating the proportion of null queries incorrectly judged as answerable among all queries classified as answerable.
[2] Both NewsAgent-Source and NewsAgent-Domain share the same knowledge repository for the single RAG agent, thus producing identical results. We therefore report these single-agent outcomes under NewsAgent-Source only, leaving the corresponding entries for NewsAgent-Domain blank to avoid duplication.

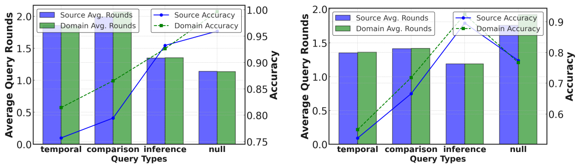
Table 4: Performance comparison of different methods using NewsAgent-Source and NewsAgent-Domain.

the server employs Qwen-Plus while agents run lightweight models, substantially enhances performance compared to using LLaMA models alone. With the results from Table 2, this improvement highlights the critical role of the response evaluator: a stronger evaluator can more accurately assess intermediate responses, thereby reducing error propagation. These findings underscore the practical advantages of RIRS, enabling efficient and privacy-preserving deployment. Specifically, lightweight models can be deployed locally on agents to maintain efficiency and knowledge sovereignty, while a centralized, more capable model handles complex reasoning tasks on the server without requiring sensitive data to be shared.

5.4 Analysis for Multi-hop RAG

In this section, we analyze the performance of our proposed method, RIRS, on the MHR benchmark under two NewsAgents settings: **NewsAgent-Source** (with 49 agents) and **NewsAgent-Domain** (with six agents). As shown in Table 4 and Figure 3, we focus on two key perspectives: (i) comparing RIRS with single RAG agents, and (ii) discussing the effect of varying the number of RAG agents.

Comparison with Single RAG Agent. In the single-agent scenario, all knowledge is consolidated into a single agent (e.g., RankRAG and EfficientRAG). While this setup can sometimes simplify retrieval by reviewing a comprehensive document pool, it also risks introducing irrelevant or distracting information. By contrast, RIRS partitions knowledge across multiple specialized agents and iteratively refines query routing, thereby reducing the likelihood of retrieving spurious content. As Table 4 indicates, RIRS achieves not only higher accuracy but also a notably lower hallucination rate compared to the single-agent methods. In particular, when dealing with multi-hop queries, the domain-specific knowledge bases (or source-specific segments) mitigate confusion and enhance



(a) Qwen-Plus (b) LLaMA-3.1-8B

Figure 3: Performance Comparison of Different Models for MHR Benchmarks under Different Query Types.

the reliability of retrieved evidence, helping RIRS avoid erroneous or fabricated answers.

Discussion of Different Numbers of RAG Agents. From Table 4 and Figure 3, we observe that increasing the number of agents provides finer-grained knowledge coverage and typically requires fewer query rounds. In contrast, using fewer agents increases routing complexity but can yield higher accuracy. Compared to the upper-bound method (GoldRouter), performance degradation is more obvious in NewsAgent-Source than in NewsAgent-Domain. This discrepancy can be attributed to suboptimal agent formation in NewsAgent-Source: knowledge is partitioned by news source, resulting in each agent covering a diverse mix of topics. The unstructured topic coverage limits the system's ability to route queries to proper agents effectively. Moreover, the number of agents in NewsAgent-Source is considerably more than in the domain-based configuration, thereby increasing routing uncertainty and deteriorating performance.

A closer comparison between the single-RAG-agent system and NewsAgent-Domain reveals the existence of an *optimal number* of agents for a given knowledge base. Using too few agents can lead to overly broad domain coverage, which reintroduces the challenge of retrieving irrelevant information. In contrast, the domain-based configuration results in more efficient and accurate resolution of complex, multi-hop questions.

6 Conclusion

This work introduces a novel iterative routing framework RIRS that coordinates multiple RAG agents in response to a query while preserving their knowledge sovereignty. Specifically, this framework consists of two mechanisms: the routing mechanism directs a user query to the most appropriate agents, and the iterative refining-solving mechanism enhances the system's ability to tackle complex, multi-hop queries by progressively synthesizing intermediate responses into a comprehensive final answer. Extensive experiments demonstrate the superiority of RIRS over the existing multi-agent routing approaches, illustrating its estimation ability in an agent's knowledge boundary.

Limitations

Despite the encouraging results, RIRS has a few limitations that suggest directions for future work. First, the proposed RIRS adopts a hierarchical clustering algorithm for knowledge clustering, which may not scale well when a single agent is loaded with an excessively large knowledge base. A scalable alternative could involve dividing the knowledge base into manageable subsets, each handled by a separate RAG agent. Second, RIRS is currently limited to text-only corpora and does not yet support multimodal inputs, such as images or audio. Extending the framework to support larger-scale knowledge bases and multimodal data would further broaden its applicability.

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*.

Parker Addison, Minh-Tuan H Nguyen, Tomislav Medan, Mohammad T Manzari, Brendan McElrone, Laksh Lalwani, Aboli More, Smita Sharma, Holger R Roth, Isaac Yang, et al. 2024. C-fedrag: A confidential federated retrieval-augmented generation system. *arXiv preprint arXiv:2412.13163*.

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023a. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023b. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2023a. Autoagents: A framework for automatic agent generation. *arXiv preprint arXiv:2309.17288*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T Kwok, and Yu Zhang. 2024a. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *arXiv preprint arXiv:2409.19886*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. 2023b. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Kai Chen, and Feng Zhao. 2024b. Mindsearch: Mimicking human minds elicits deep ai searcher. *arXiv preprint arXiv:2407.20183*.

Vincent Cohen-Addad, Benjamin Guedj, Varun Kanade, and Guy Rom. 2021. Online k-means clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 1126–1134. PMLR.

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

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, et al. 2024. Agentscope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. *arXiv preprint arXiv:2207.06300*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

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. *arXiv preprint arXiv:2011.01060*.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Isaac Johnson, Martin Gerlach, and Diego Sáez-Trumper. 2021. Language-agnostic topic classification for wikipedia. *WWW '21: Companion Proceedings of the Web Conference 2021*, page 594–601.

Ashutosh Joshi, Sheikh Muhammad Sarwar, Samarth Varshney, Sreyashi Nag, Shrivats Agrawal, and Juhi Naik. 2024. Reaper: Reasoning based retrieval planning for complex rag systems. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4621–4628.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, Kentaro Inui, et al. 2024. Realtime qa: what's the answer right now? *Advances in Neural Information Processing Systems*, 36.

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

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

Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. 2022. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*.

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.

Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. 2024. Agent-oriented planning in multi-agent systems. *arXiv preprint arXiv:2410.02189*.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Xiaonan Li, Changtai Zhu, Linyang Li, Zhangyue Yin, Tianxiang Sun, and Xipeng Qiu. 2023b. Llatrieval: Llm-verified retrieval for verifiable generation. *arXiv preprint arXiv:2311.07838*.

Jerry Liu. 2022. LlamaIndex.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2024. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems*, 36.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Nicholas Monath, Manzil Zaheer, and Andrew McCallum. 2023. Online level-wise hierarchical clustering. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1733–1745.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

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.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49.

Aymeric Roucher. 2024. Agentic RAG: turbocharge your RAG with query reformulation and self-query. https://huggingface.co/learn/cookbook/en/agent_rag. Accessed: 2025-02-15.

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*.

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*.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*.

EuiYul Song, Sangryul Kim, Haeju Lee, Joonkee Kim, and James Thorne. 2024. Re3val: Reinforced and reranked generative retrieval. *arXiv preprint arXiv:2401.16979*.

KV Srivatsa, Kaushal Kumar Maurya, and Ekaterina Kochmar. 2024. Harnessing the power of multiple minds: Lessons learned from llm routing. *arXiv preprint arXiv:2405.00467*.

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. *arXiv preprint arXiv:2403.10081*.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

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.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. 2024. Plan $\times$ rag: Planning-guided retrieval augmented generation. *arXiv preprint arXiv:2410.20753*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Lilian Weng. 2023. LLM powered autonomous agents. https://lilianweng.github.io/posts/2023-06-23-agent/. Accessed: 2025-01-09.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Shangyu Wu, Ying Xiong, Yufei Cui, Haolun Wu, Can Chen, Ye Yuan, Lianming Huang, Xue Liu, Tei-Wei Kuo, Nan Guan, et al. 2024. Retrieval-augmented generation for natural language processing: A survey. *arXiv preprint arXiv:2407.13193*.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. *arXiv preprint arXiv:2402.13178*.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

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

Diji Yang, Jinmeng Rao, Kezhen Chen, Xiaoyuan Guo, Yawen Zhang, Jie Yang, and Yi Zhang. 2024b. Im-rag: Multi-round retrieval-augmented generation through learning inner monologues. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 730–740.

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*.

Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. 2025. Mas-gpt: Training llms to build llm-based multi-agent systems. *arXiv preprint arXiv:2503.03686*.

Wenhao Yu. 2022. Retrieval-augmented generation across heterogeneous knowledge. In *Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies: student research workshop*, pages 52–58.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *arXiv preprint arXiv:2407.02485*.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, et al. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *arXiv preprint arXiv:2407.19669*.

Qinlin Zhao, Jindong Wang, Yixuan Zhang, Yiqiao Jin, Kaijie Zhu, Hao Chen, and Xing Xie. 2023. Competeai: Understanding the competition behaviors in large language model-based agents. *arXiv preprint arXiv:2310.17512*.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. 2024. Loraretriever: Input-aware lora retrieval and composition for mixed tasks in the wild. *arXiv preprint arXiv:2402.09997*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

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 Related Works

Retrieval Augmented Generation (RAG). RAG has gained substantial interest in academic research as a robust framework that integrates external knowledge sources into large language models to enhance the quality and reliability of generated responses (Lewis et al., 2020; Jiang et al., 2023; Chen et al., 2017; Guu et al., 2020; Karpukhin et al., 2020; Izacard and Grave, 2020; Borgeaud et al., 2022; Yu, 2022; Shi et al., 2023; Yan et al., 2024; Asai et al., 2023b; Li et al., 2023b; Press et al., 2022; Chan et al., 2024; Su et al., 2024; Trivedi et al., 2022a; Ma et al., 2023; Shao et al., 2023). Notable recent contributions in this domain include RankRAG (Yu et al., 2024), which reranks the selected knowledge pieces and generates a response with genuinely important ones; EfficientRAG (Zhuang et al., 2024), an approach that iteratively generates new queries by sorting out the portion addressed by retrieved knowledge until a multi-hop question can be well-addressed; PlanRAG (Verma et al., 2024), which decomposes complex queries into interrelated atomic sub-queries by formulating a reasoning plan as a directed acyclic graph (DAG). While these methods primarily focus on single-agent retrieval frameworks, our work differentiates itself by addressing the challenge of coordinating knowledge retrieval across multiple agents, thereby enabling a more comprehensive handling of diverse and cross-domain queries.

Routing Mechanism in Multi-agent System. LLM-based multi-agent systems leverage the collective intelligence and specialized capabilities of multiple expert agents to collaboratively tackle complex problems, a research direction that has garnered significant interests (Hong et al., 2023; Li et al., 2023a; Wu et al., 2023; Chen et al., 2023a,b; Zhao et al., 2023; Guo et al., 2024; Chen et al., 2024b). In such systems, a robust routing mechanism is essential to direct each query to the most appropriate agent based on its unique expertise (Shnitzer et al., 2023; Lu et al., 2023; Zhao et al., 2024; Srivatsa et al., 2024; Li et al., 2024; Lu et al., 2024; Chen et al., 2024a; Addison et al., 2024; Ye et al., 2025).

Recent works in this area include Chameleon (Lu et al., 2024), which requires a comprehensive textual description of each agent's capabilities such that a well-trained LLM can select the most appropriate agents to address an input query; RouterDC (Chen et al., 2024a), which trains a lightweight model to dynamically route queries to the most suitable agent within a predefined set; and C-FedRAG (Addison et al., 2024), which forwards queries to randomly selected agents to retrieve relevant documents such that a central entity can collect these documents to generate a final answer.

In contrast, our approach accurately characterizes each agent's knowledge capacity based on their own data, thereby eliminating the need for exhaustive textual descriptions and labor-intensive dataset labeling. Moreover, our effective routing mechanism ensures that queries are directed only to agents capable of providing comprehensive answers, with each agent summarizing its response based solely on local knowledge, thereby preserving data privacy and protecting knowledge sovereignty.

B Key Modules on the Server

The proposed RIRS is comprised of two key components: multiple RAG-based agents, each specialized in a domain of expertise, and a central server that coordinates their collaborative efforts. The server serves as the nexus of our framework by orchestrating the interaction between agents through a series of specialized roles designed to ensure that user queries are addressed with both logical rigor and relevant supporting evidence. Below, we detail the primary agents managed by the server and their corresponding responsibilities.

B.1 Response Evaluator

The Response Evaluator is the first checkpoint in the server's processing pipeline. Given that the server itself lacks domain-specific background knowledge, the evaluator assesses agent responses using a common-sense, logic-based approach. Its primary tasks are:

- **Logical Assessment:** Evaluating whether the provided response is coherent and free from logical fallacies.
- **Evidence Verification:** Confirming that necessary citations and supporting evidence are included in the response.
- **Response Classification:** Determining if the response is "addressed" or "not addressed." A response is marked "not addressed" if it fails to provide sufficient evidence, contains logical deficiencies, or is entirely irrelevant to the user query.
- **Granular Evaluation for Multi-hop Questions:** For multistep queries, even a response classified as "addressed" is further divided into "fully addressed" or "partially addressed." When a response is rated as "partially addressed," the unaddressed part is expected to be answered by other agents and/or in the further reasoning steps.

This preliminary evaluation ensures that only logically sound and evidence-backed responses are propagated in the subsequent stages.

B.2 Response Summarizer

Following the evaluation stage, the Response Summarizer plays a crucial role in consolidating agent responses. Depending on the evaluator's results, two variants of the summarizer are deployed:

- **Fully Addressed Summarizer:** This variant consolidates one or more responses that have been deemed to fully address the query. It integrates the responses into a comprehensive answer which is then forwarded directly to the user.
- **Partially Addressed Summarizer:** When none of the responses can fully address the query, the partially addressed summarizer steps in to compile a more complete solution based on the available partial responses. Once consolidated, it re-evaluates the answer. If the unified response is deemed fully addressed, it is returned to the user; otherwise, further action is initiated.

Notably, although the partially addressed summarizer can directly consolidate the information from all responses, the overall performance of response summarizing benefits from the initial filtering conducted by the Response Evaluator and fine-granularity of the standard collaboration flow.

B.3 Question Simplifier

When the query remains insufficiently addressed, as rated by the partially addressed summarizer, the Question Simplifier intervenes to decompose the problem into more manageable sub-questions. This agent performs the following tasks:

- **Identification of Addressed Components:** It examines the existing responses to isolate the aspects of the query that have already been effectively addressed.
- **Generation of a New Question:** By removing the resolved parts, the simplifier formulates a new query targeting the unresolved components.
- **On-the-Fly Decomposition:** This dynamic simplification enables the server to continue resolving the query in an iterative, step-by-step manner without the need for prior, rigid question planning.

This adaptive approach leverages already acquired knowledge, ensuring that subsequent agent interactions are focused on the remaining aspects of the problem, thereby reducing the need for redundant multistep reasoning in later stages.

B.4 Discussion: Unhandleable Queries

In some instances, the server may fail to obtain any reliable ("addressed") responses. This scenario may arise due to several factors:

- **Query Outside the System's Domain:** The user may pose a question that falls outside the scope of all RAG-based agents' expertise. For instance, in a medical suggestion QA system, a travel recommendation query would not align with the agents' specialized knowledge, resulting in no suitable answer.
- **Ambiguous or Incomplete Queries:** A query that is vague, under-specified, or contains numerous typographical errors can hinder the server's ability to correctly map the question to the appropriate agents. For example, in a medical context, ambiguous terminology or poorly structured queries may impede the identification of a clear problem statement, leading to an inability to retrieve a fully addressed response.
- **Rapidly Evolving Information Domains:** In areas where information is rapidly changing, some RAG-based agents may not have the most current data or guidelines. This lag can result in responses that are either outdated or insufficient, prompting the system to classify the query as out-of-scope.

In such cases, the system will inform the user that the question cannot be answered based on the current scope of the available agents. However, if the server integrates a knowledgeable agent with broader capabilities, this agent may be employed to attempt an answer. For multi-hop questions, the knowledgeable agent is provided with a simplified version of the query, since evidence suggests that large language models perform better when fewer reasoning steps are required (Zhuang et al., 2024).

C Implementations and Baselines

C.1 Implementation Details

Our implementation builds upon the open-source AgentScope project (Gao et al., 2024), and each agent's knowledge base is processed by LLaMAIndex (Liu, 2022) and stored in ElasticSearch. We leverage ElasticSearch's Mixed Retrieval for knowledge retrieval, where each agent extracts 20 chunks and selects the best five for answer generation. Without special annotations, five agents are chosen per query round. The entire system is deployed within a pseudo-distributed environment that utilizes multi-threading to enhance scalability and efficiency. As for models, we call Qwen-Plus (Yang et al., 2024a; Bai et al., 2023) via API and deploy the LLaMA-3.1-8B model locally using the vLLM service (Kwon et al., 2023) with four Nvidia A100 GPUs. Our code and the setup of multi-agent systems (including the corpus) will be released upon acceptance.

C.2 Datasets

As each of the Wikipedia-related datasets (i.e., NQ, HotpotQA, 2WikiMultiQA, and MusiQue) consists of thousands of questions, we sample a subset of questions from each dataset to speed up our evaluation progress, while ensuring at most 3% error within a 95% confidence interval.

C.3 Multi-Agent Settings

Chameleon, RouterDC, and RIRS are in a multi-agent setting. To simulate the practical knowledge domain segmentation, we construct two groups of RAG agents for different datasets:

- **WikiAgents for NQ, HotpotQA, 2WikiMQA and MusiQue:** WikiAgents group is built upon a corpus of over 121K Wikipedia pages, dumped as of November 1, 2023, and made publicly available via the HuggingFace dataset. The system comprises exactly 64 RAG agents. This specific number is derived from the inherent limitation of the ORES legacy service, which can only classify a Wikipedia page into 64 predefined categories (Johnson et al., 2021). Consequently, each RAG agent is designated to handle one of these 64 categories, ensuring that the categorization of pages is consistent and aligned with the predefined taxonomy established by the ORES service.

However, these agents cannot cover all required documents, and once the question cannot be answered, a knowledgeable agent will be called to answer the question because the existing LLMs have been pretrained with the Wikipedia corpus.

- **NewsAgents for MHR:** NewsAgents group is constructed on a corpus of 609 news articles from 49 distinct news media and spanning six domains (Tang and Yang, 2024). To analyze the effects of source and domain characteristics, we further split NewsAgent into two variants: *NewsAgent-Source* and *NewsAgent-Domain*, corresponding to partitions based on news media and domain categories, respectively. Each agent uses LLaMAIndex to split news articles into 256-token chunks, with 20 overlapping tokens between consecutive chunks.

C.4 Baselines

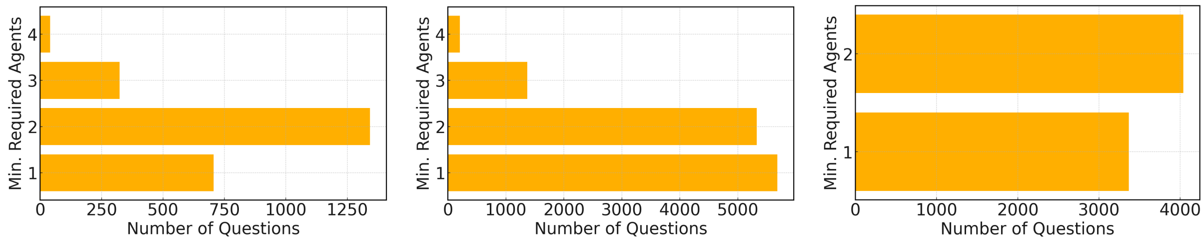
In order to comprehensively evaluate our proposed system, we have reproduced several baselines inspired by existing works. These baselines are implemented manually to fit within our experimental framework. Notably, our study focuses on an off-the-shelf scenario, which does not fine-tune or train any models and instead leverages the capabilities of existing LLMs to achieve desired effects. Toward the goal, the implementation details of the baselines are given as follows:

Single-agent Scenarios. In the single-agent scenario, the knowledge contained within all RAG-based agents is merged into a single agent. This is a unified setup for conventional RAG-based methods. In this setting, baseline methods can review all documents and retrieve the most relevant ones within the system. Although the single-agent setup does not fully align with the scenario we aim to examine, we include it to demonstrate the challenges associated with managing a large knowledge base, i.e., longer retrieval times and potential distractions from plausible yet irrelevant information. Ideally, if the retrieved knowledge were perfectly clean, the single-agent setup could serve as an upper bound of the multi-agent setup in terms of accuracy when comparing the generated results against the ground truth.

- **RankRAG:** This method retrieves some documents from the knowledge base using both dense and sparse retrievers. Next, a pretrained model is introduced to evaluate the helpfulness of each retrieved document and select the most appropriate document(s) to answer the given query. Based on the retrieved document(s), the LLM is asked to generate a response. If the provided document(s) are irrelevant to the question, the LLM is supposed to generate the answer on its own ability. Therefore, this method maintains a single query round for all types of questions. This baseline method covers a number of the existing works (Yu et al., 2024; Glass et al., 2022; Song et al., 2024; Ram et al., 2023; Ma et al., 2023; Nogueira et al., 2020), which focuses on using reranking to enhance LLM content generation, while they use different ways to train the ranking model.
- **EfficientRAG:** This approach iteratively simplifies the query by retrieving query-related documents from the knowledge base until it can be fully addressed. The simplification process depends entirely on the LLM's ability to remove portions of the query that have already been resolved. In cases where the remaining question cannot be further simplified or adequately answered using the provided context, a knowledgeable agent is used to generate the final answer. This baseline follows the EfficientRAG framework (Zhuang et al., 2024), which generates training data from an LLM to train a compact model for question simplification, and is further inspired by related works such as Self-ASK (Press et al., 2022), SelfRAG (Asai et al., 2023b), and IM-RAG (Yang et al., 2024b). This approach has proven effective for multi-hop questions due to its adoption of multiple reasoning steps to converge on a final answer.

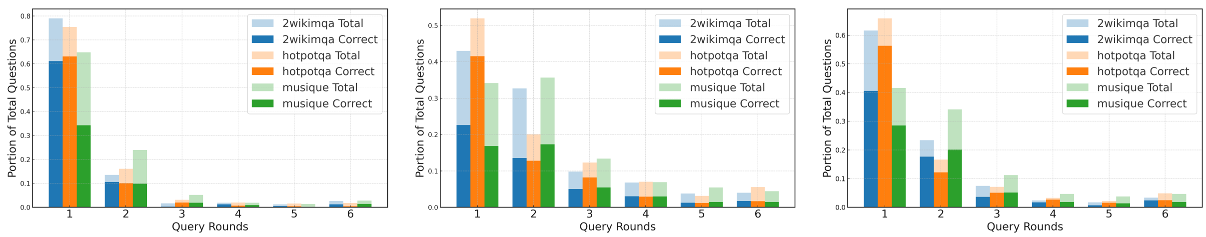
Multi-agent Scenarios. In this scenario, we implement two routing strategies for comparison with our proposed routing mechanism, RIRS, while keeping the other server modules unchanged and still employing iterative routing. These two methods are derived from settings that involve multiple LLMs and use a router to identify the best LLM for handling a given task. Inspired by these approaches, we extend their ideas to our scenario to construct an effective routing mechanism. In our experiments, these two routing strategy are used to handle Wikipedia-related QA tasks, i.e., Natural Questions, HotpotQA, 2WikiMultiHopQA, and MuSiQue.

- **Chameleon:** This method (Lu et al., 2024) leverages a collection of tools, including LLMs and off-the-shelf vision models, to accomplish complex reasoning tasks step by step, selecting the best tool for each step. The router, which is based on a well-trained LLM, decomposes a com-



(a) MuSiQue (b) 2WikiMultiHopQA (c) HotpotQA

Figure 4: Data distribution v.s. Minimum required agents under different datasets



(a) Qwen-Plus (b) LLaMA-3.1-8B (c) Mixed

Figure 5: Evaluation for Wikipedia-related QA (2WikiMQA, HotpotQA, and MuSiQue)

plex task into multiple steps and identifies the most suitable tool for each. In our adaptation, each WikiAgent is dedicated to a specific topic from Wikipedia. By providing the router with a description of each agent's specialized topic, it can select up to five agents whose expertise best aligns with the given query.

- **RouterDC:** This method (Chen et al., 2024a) utilizes several LLMs by training a representation vector for each model based on a collection of questions and the optimal candidate from a pool of models. During inference, the router determines the best agents by computing the similarity between the query embedding and the stored representation vectors.

  Inspired by this approach, we calculate the similarity between the query embedding and the representation vectors of our RAG-based agents. However, obtaining these vectors typically requires additional training, which is not compatible with our training-free setting. Instead, we leverage insights from recent works (Lampinen et al., 2022; Mishra et al., 2021) and cache 100 historical questions for each agent to represent their knowledge capacity. For a new question, the router computes the average similarity between the query and the cached questions for each agent, then selects the five agents with the highest similarity scores to generate an answer.

D More Experimental Results for Multi-hop Wikipedia-related QA

Number of Required Agents. Figure 4 illustrates the distribution of questions by the minimum number of required agents across three multi-hop QA datasets: MuSiQue, 2WikiMultiHopQA, and HotpotQA. Although all three datasets consist of multi-hop questions, the necessary knowledge pieces may sometimes reside within a single agent. In MuSiQue and 2WikiMultiHopQA, most questions can be answered using two RAG agents, and all questions require no more than four agents. For HotpotQA, all questions can be answered with just two agents. It is important to note that the number of required agents does not directly correspond to the number of query rounds. Multiple agents can be invoked simultaneously within a single round, and conversely, a single agent may be queried multiple times to perform complex reasoning before reaching the final answer.

Query Rounds and Accuracy. Figure 5 presents the proportion of questions that can be answered within different numbers of query rounds, along with the corresponding answer correctness. Across all LLMs, most questions are resolved within the first two query rounds. Specifically, stronger models (e.g., Qwen-Plus) can answer more questions compared to weaker ones (e.g., LLaMA-3.1-8B) within fewer query rounds. Interestingly, the figure also shows that increasing the number of query rounds does not necessarily lead to improved answer accuracy. In fact, unless a question inherently requires multi-step reasoning, additional query rounds may result in even poorer performance.

Case Study. Figures 12 and 13 illustrate how the proposed RIRS answers a single-hop question and a multi-hop question, respectively.

E Prompts

System Prompts for Chain-of-Thought (CoT) Agent

You're a knowledgeable assistant. You are provided with a question, and you should answer the question in the following two steps. FIRST, you should utilize your knowledge and analyze the question step by step. SECOND, you should finalize an answer based on your analysis with no more than 30 words.

Your output should be in the json format:

~~~
```json
{
    "analysis": "<a paragraph with no more than six sentences>",
    "answer": "<a response within 30 words>"
}
```
~~~

Figure 6: Prompt Templates for the CoT Agents.

System Prompts of Wikipedia Agents for A Single Question

You are provided with one question and a collection of knowledge. Based on the given knowledge, you should try to analyze and tackle the question as thoroughly as possible, even if you cannot fully answer the question. Unless the given context is unrelated to the question, you must directly quote the evidence (i.e., sentences) without being altered to support your analysis, enclosing it in double asterisks (**). You should not state any arguments that are not explicitly mentioned or implied from the pieces of evidence or without quoting them. The analysis should be in one paragraph with no more than ten sentences. Moreover, the analysis should start with "I" and not mention that the analysis is generated based on the given knowledge, documents, or information.

Your output should be in the json format:

~~~
```json
{
    "analysis": "<one paragraph of up to ten sentences, directly quoting supporting evidence from
        the provided knowledge>"
}
```
~~~

Figure 7: Prompt Templates for the Wikipedia Agents.

System Prompts of News Agents for A Single Question

You are given one question and a collection of news articles. Each article contains content along with its source information, including the title, news source, author, and published time. Your task is to analyze and address the question as thoroughly as possible based on the provided news articles, even if you cannot fully answer the question. Unless the given context is unrelated to the question, you must directly quote the evidence (i.e., sentences) without being altered to support your analysis. You should enclose quoted evidence (sentences) in double asterisks (**), followed by the source in brackets, including the title, news source, author, and published time, separated by semicolons (;). You should not state any arguments that are not explicitly mentioned or implied from the pieces of evidence or without quoting them. Your analysis to each question should be concise, limited to one paragraph per question, with no more than ten sentences. The analysis must begin with "I" and should not mention that the analysis is based on provided news articles, knowledge, or information.

Your output should be in the json format:

````
```json
{
    "analysis": "<one paragraph of up to ten sentences, directly quoting supporting evidence from
        the provided knowledge>"
}
```
````

Figure 8: Prompt Templates for the News Agents.

System Prompts of Evaluator for "Fully/Partially/Not Addressed" for A Question and A Response

You are provided with a question and a response. Your task is to evaluate the response according to the following steps:
1. Assess the response against three criteria:
- Relevance: Does the response help answer the question, even if the response does not fully resolve it?
- Evidence-Based Support: Are statements supported by explicitly mentioned evidence enclosed in double asterisks (**)?
- Logical Coherence: Is it well-structured, logically reasoned, and free from logical fallacies or contradictions?
2. Assign one of the following ratings:
- Fully addressed: The response meets all criteria and completely answer the question.
- Partially addressed: The response meets all criteria but not fully resolve the question.
- Not addressed: The response fails to meet one or more of the criteria.

Your output should be in the json format:

````
```json
{
    "evaluation": {
        "relevance": <a sentence within 30 words>,
        "evidence_support": <a sentence within 30 words>,
        "logical_coherence": <a sentence within 30 words>
    },
    "rating": <"Fully addressed" or "Partially addressed" or "Not addressed">
}
```
````

Figure 9: Prompt Templates for the evaluator.

System Prompts of Summarizing Fully-addressed Responses

You are given a question, and one or more responses that fully resolve the question. Your task is to produce a final answer by following these steps:
1. Analysis: Incorporate all relevant information from the given responses, quoting any supporting evidence word-for-word in double asterisks (**).
2. Answer: Provide a concise conclusion in no more than 30 words that summarizes the analysis.

Your output should be in the json format:

````
```json
{
    "analysis": "<a paragraph that directly quotes relevant evidence in **double asterisks**>",
    "answer": "<a concise final answer within 30 words>"
}
```
````

System Prompts of Summarizing Partially-addressed Responses

You are given a question and several partially addressed responses. Your task is to combine these responses to create a comprehensive solution, then evaluate its completeness. Follow these steps:
1. Solution Synthesis:
- Incorporate all relevant information from the provided responses to form a solution that addresses the question as thoroughly as possible.
- Directly quote supporting evidence (word-for-word) using double asterisks (**).
2. Evaluation and Justification:
- Determine whether this combined solution fully addresses the question.
- Provide a clear explanation of why it does or does not fully address the question.
3. Answerability Determination:
- Based on your evaluation, decide if the solution makes the question answerable.
- Respond with either "yes" (if the solution fully addresses the question) or "no" (if it does not).
- Most importantly, if the solution explicitly states that the question cannot be fully addressed or identifies missing aspects or necessary additional information, you must answer "no".
4. Final Answer:
- If the solution is deemed answerable ("yes"), provide a concise conclusion in no more than 30 words that summarizes the solution.
- If the solution is not answerable ("no"), return "None" as the final answer.

Your output should be in the json format:

````
```json
{
    "solution": "<a paragraph that integrates responses with direct quotes in **double asterisks
        **>",
    "evaluation and justification": "<a paragraph explaining whether and why the solution fully
        addresses the question>",
    "answerable": <"yes" or "no">,
    "answer": "<a final answer within 30 words if ’answerable’ is ’yes’, or ’None’ if ’answerable’
         is ’no’>"
}
```
````

Figure 10: Prompt Templates for the Summarizer.

System Prompts of Question Simplifier

You are given a question and a piece of knowledge that partially addresses the question. Your task is to simplify or refactor the original question so that answering the simplified question will yield the same final answer as answering the original one. Follow the steps below:

1. Map Known Information:
- Identify Answered Parts: Examine the original question and determine which parts have been explicitly answered by the provided solution. Extract the relevant words or phrases from the question.
- Map to Solution: For each identified part, find the corresponding words or phrases in the provided solution and include supporting evidence by directly quoting the exact sentences, enclosed in double asterisks.
- Note: If the provided knowledge does not address any aspect of the original question (i.e., you cannot identify the answered parts from the question), return "None" for the mapping.

2. Identify Unresolved Aspects:
- Determine Gaps: Identify the parts of the question that remain unanswered or unclear after considering the provided solution.
- List Unresolved Points: Clearly list these unresolved aspects. They must be retained in the simplified question to ensure that no essential detail is lost.

3. Generate a Simplified/Refactored Question:
- Remove Redundancies: Remove from the original question any words or phrases that have been fully addressed by the solution (as determined in Step 1).
- Incorporate Mapped Knowledge: Replace the removed parts with their corresponding mapped words or phrases from the solution, ensuring to preserve the supporting evidence.
- Retain Unresolved Aspects: Ensure that any unresolved aspects identified in Step 2 are still present in the final simplified question.
- Ensure Consistency: The new question must be clear, concise, and structured so that answering it will yield the same final answer as if the original question were fully addressed.
- Note: If the mapping is "None" (as indicated in Step 1), keep the simplified/refactored question identical to the original question.

Your output should be in the json format:

````
```json
{
    "question_knowledge_mapping": <a list of tuples in the format (words from given question,
        words from provided knowledge, supporting evidence) or "None">,
    "unresolved_aspects": <a list of aspects that remain unanswered>,
    "new_question": <the simplified/refactored question; remove all "words from given question"
        found in the mapping and include the corresponding "words from provided knowledge" to
        replace them, ensuring all unresolved aspects are retained. If mapping is "None", use the
        original question>
}
```
````

Figure 11: System Prompt for Question Simplifier.

User Input

who is darrell brother in the walking dead

Expected Answer

Merle Dixon

Assistant: Culture.Media.Media*

I learned that there is no direct mention of a character named 'darrell brother' in the provided context related to 'The Walking Dead'. The content does not provide any information about characters from this series. Therefore, I cannot answer who darrell's brother is based on the given information.

Assistant: Culture.Literature

I learned that **Daryl Dixon is a fictional character from AMC's horror drama series The Walking Dead, and the protagonist of its last three seasons.** It also mentions that **the character was introduced in the first season as a southerner, expert tracker, living in the shadow of his older brother, Merle.** From another source, it states that **Merle Dixon is a fictional character from the horror drama television series The Walking Dead, which aired on AMC in the United States. He was created by series developer Frank Darabont and was portrayed by Michael Rooker.** These sentences explicitly indicate that Merle is Daryl's older brother in The Walking Dead.

Server: Evaluation


```
{
    "response_1": {
        "relevance": "Misinterprets the question by focusing on a character named 'darrell brother
            ', which does not exist.",
        "evidence_support": "No relevant evidence provided regarding Darrell or his brother.",
        "logical_coherence": "The response is logically coherent but irrelevant to the asked
            question.",
        "rating": "Not addressed"
    },
    "response_2": {
        "relevance": "Identifies Merle as Daryl's older brother, accurately addressing the query.",
        "evidence_support": "Supports claims with multiple pieces of evidence from the series.",
        "logical_coherence": "Well-structured with logical flow and no contradictions.",
        "rating": "Fully addressed"
    }
}
```


Server: FullySummarizer


```
{
  "analysis": "The responses collectively indicate that **Merle Dixon is Daryl Dixon's older
      brother** in The Walking Dead. They provide consistent information from multiple sources
      confirming Merle as the elder sibling, portrayed by Michael Rooker, and known for his
      volatile character.",
  "answer": "In The Walking Dead, Daryl's older brother is Merle Dixon, a volatile character
      portrayed by Michael Rooker."
}
```


Response

In The Walking Dead, Daryl's older brother is Merle Dixon, a volatile character portrayed by Michael Rooker.

Figure 12: A example of single-hop QA task

User Input

Where was the director of film Rough Going born?

Expected Answer

New Hyde Park, New York

Assistant: Culture.Media.Films

I found that **Rough Going is a 1925 American silent Western film directed by Wally Van**. However, the provided information does not specify where the director, Wally Van, was born.

Assistant: Culture.Media.Media*

I learned that **Richard Rush (April 15, 1929 – April 8, 2021) was an American film director, scriptwriter, and producer.** This indicates that Richard Rush, the director of Rough Going, was born in the United States. However, the specific place of his birth is not mentioned in the provided context.

Server: Evaluation


```
{
    'response_1': {
        'relevance': 'Identifies the director of Rough Going but does not provide his place of
            birth.',
        'evidence_support': 'Provides relevant information about the film and its director, Wally
            Van, with evidence.',
        'logical_coherence': 'The response is logically structured and coherent, though it
            acknowledges incomplete information.',
        'rating': 'Partially addressed'
    },
    'response_2': {
        'relevance': 'Incorrectly identifies Richard Rush as the director of Rough Going, making it
             irrelevant to the question asked.',
        'evidence_support': "Offers evidence about Richard Rush's lifespan and profession, but this
             does not relate to the query.",
        'logical_coherence': 'Logically coherent for statements about Richard Rush, but fails to
            address the actual question.',
        'rating': 'Not addressed'
    }
}
```


Server: PartiallySummarizer


```
{
    'solution': 'The film **Rough Going is a 1925 American silent Western film directed by Wally
        Van**. However, the provided information does not specify where the director, Wally Van,
        was born. This indicates a gap in the available data regarding the birthplace of Wally Van
        .',
    'evaluation and justification': 'The solution identifies the director of the film Rough Going
        as Wally Van but acknowledges that there is no information provided about his place of
        birth. Since the question specifically asks for the birthplace of the director and this
        detail is missing, the solution does not fully address the question.',
    'answerable': 'no',
    'answer': 'None'
}
```

Server: QuestionSimplifier


```
{
    'question_knowledge_mapping': [
        ['the director of film Rough Going', 'Wally Van', '**Rough Going is a 1925 American silent
            Western film directed by Wally Van**.']
    ],
    'unresolved_aspects': ['The birthplace of Wally Van'],
    'new_question': "Where was Wally Van born?"
}
```


Assistant: Culture.Biography.Biography*

I learned that **Van was born Charles Wallace Van Nostrand on September 27, 1880, in New Hyde Park, New York.** This explicitly answers the question about where Wally Van was born, which is New Hyde Park, New York.

Assistant: Culture.Linguistics

I found that **Wally Williams (water polo) (1921–2009), New Zealand water polo player** is listed, indicating a Wally Williams who was born in 1921 and was from New Zealand. However, there is no specific mention of where exactly he was born within New Zealand.

Server: Evaluation


```
{
    'response_1': {
        'relevance': "Directly answers the question by specifying Wally Van's birthplace as New
            Hyde Park, New York.",
        'evidence_support': 'Provides explicit evidence enclosed in double asterisks about Wally
            Van's birth details.',
        'logical_coherence': 'The response is logically coherent and clearly structured without
            contradictions.',
        'rating': 'Fully addressed'
    },
    'response_2': {
        'relevance': 'Partially relevant as it discusses a person named Wally but not the specific
            Wally Van in question.',
        'evidence_support': 'Provides evidence about Wally Williams but does not link this
            information to Wally Van.',
        'logical_coherence': "Coherent statement but fails to address the specific query about
            Wally Van's birthplace.",
        'rating': 'Not addressed'
    }
}
```


Server: FullySummarizer


```
{
    'analysis': "The response provides explicit information about Wally Van's birthplace: **Van
        was born Charles Wallace Van Nostrand on September 27, 1880, in New Hyde Park, New York
        .**",
    'answer': 'Wally Van was born in New Hyde Park, New York.'
}
```


Response

Wally Van was born in New Hyde Park, New York.

Figure 13: A example of a multihop QA task using the proposed RIRS