HiGraAgent: Dual-Agent Adaptive Reasoning over Hierarchical Knowledge Graph for Open Domain Multi-hop Question Answering

Hung Luu[1,2], **Long S. T. Nguyen**[1,2,3], **Trung Pham**[1,2], **Hieu Pham**[1,2], **Tho Quan**[1,2*]
[1]URA Research Group, Ho Chi Minh City University of Technology (HCMUT), Vietnam
[2]Vietnam National University Ho Chi Minh City, Vietnam
[3]Center for AI Research (CAIR), VinUniversity, Vietnam
[*]**Correspondence:** qttho@hcmut.edu.vn

Abstract

Open Domain Multi-hop Question Answering faces a dual compositionality challenge: reasoning over complex query structures and integrating evidence scattered across contexts. Despite recent advancements in Graph-based Retrieval-Augmented Generation (GraphRAG), persistent limitations in complex reasoning and retrieval inaccuracies continue to constrain the efficacy of multi-hop QA systems. We introduce HiGraAgent, a framework that unifies graph-based retrieval with adaptive reasoning. It constructs a Hierarchical Knowledge Graph (HiGra) with entity alignment, reducing redundancy by 34.5% while preserving expressiveness; employs HiGraRetriever, a hybrid graph–semantic retriever that consistently outperforms the strongest graph-based method across benchmarks; and integrates a dual-agent adaptive reasoning protocol where a Seeker and a Librarian dynamically coordinate retrieval and reasoning. Together, these innovations enable HiGraAgent to achieve 85.3% average accuracy on HotpotQA, 2WikiMultihopQA, and MuSiQue, surpassing the strongest prior system by 11.7%. Our results highlight the importance of reframing multi-hop QA as a problem of adaptive reasoning, offering a more robust and flexible paradigm for complex information seeking.[1]

1 Introduction

Multi-hop Question Answering (QA) over open domain corpora extends traditional distractor-based QA by requiring reasoning over multiple documents, thereby increasing both retrieval and answer-generation complexity (Mavi et al., 2024). Classical Information Retrieval (IR) methods falter when evidence is scattered across passages, especially in the presence of high-similarity noise or implicit keywords. Recent advances address this challenge through Graph-based Retrieval-Augmented Generation (GraphRAG), which represents textual information via relational structures such as Knowledge Graphs (KG), yielding notable improvements in retrieval quality (Gutiérrez et al., 2025a; Wang et al., 2023). However, prior work (Press et al., 2023) has revealed that the *compositionality gap* persists even for Large Language Models, underscoring the necessity of advances in both retrieval and reasoning for multi-hop QA.

While retrieval determines whether relevant evidence can be identified, reasoning dictates how such evidence is composed into an answer. On the reasoning side, decomposing complex questions into smaller sub-questions and solving them iteratively has emerged as a promising direction (Trivedi et al., 2023; Cheng et al., 2025). Yet, the very notion of multi-hopness remains underexplored. We argue that *multi-hopness* is not inherent to the question itself, but emerges from its interaction with context: a question may appear multi-hop only when its supporting evidence is fragmented. As demonstrated in Table 1, the necessity for multi-hop reasoning shifts with the structure of available evidence. However, prevailing iterative reasoning frameworks are predominantly constrained by a *sequential reasoning flow*. This rigidity makes them ill-suited for problems requiring branching reasoning paths or parallel exploration, thereby limiting the capacity for adaptive, context-dependent query decomposition.

These limitations motivate a reframing of multi-hop QA as an *adaptive reasoning* task rather than a fixed pipeline. To this end, we introduce **HiGraAgent** (**Hi**erarchical Knowledge **Gra**ph Dual-**Agent** Question Answering), a framework that fuses a hierarchical KG with a dual-agent architecture. Drawing inspiration from the Hydra, HiGraAgent can pursue multiple reasoning paths simultaneously, making it well-suited for fragmented evidence across heterogeneous contexts. At its

[1]Code and data are available at: https://github.com/headinthecloud6453/higra_agent

Findings of the Association for Computational Linguistics: EACL 2026, pages 1193–1217
March 24-29, 2026

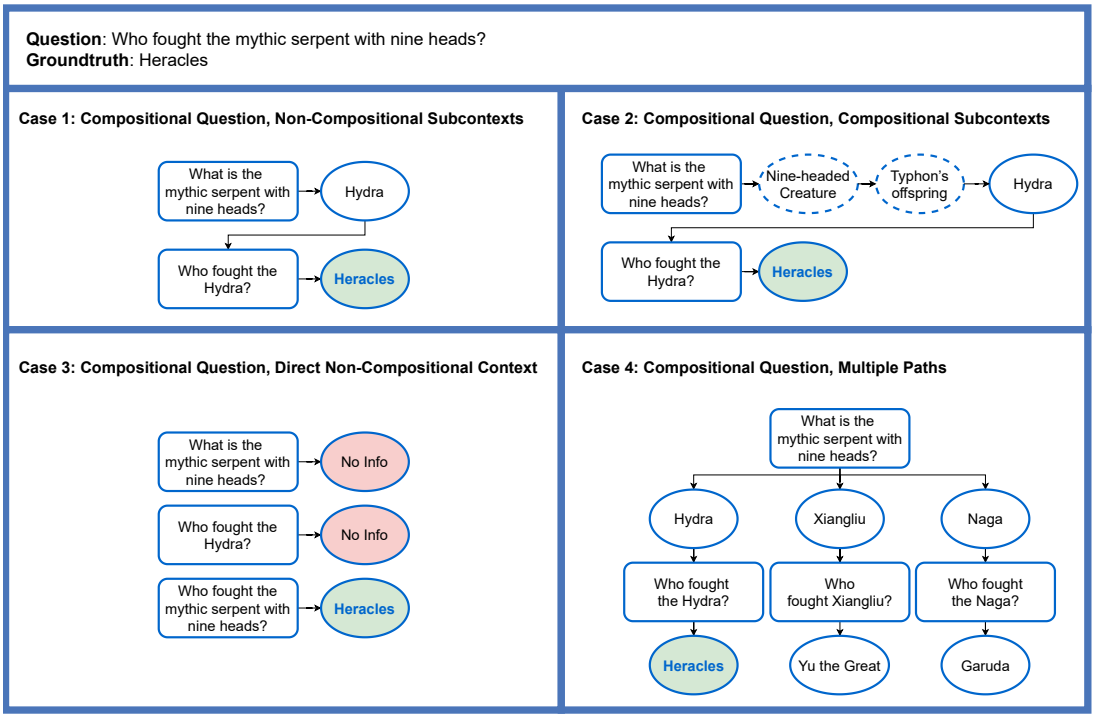

Figure 1: Examples illustrating that multi-hop reasoning emerges from evidence fragmentation and multiple reasoning paths.

core, HiGraAgent is designed to overcome two central limitations in multi-hop QA: *retrieving coherent context* and *composing queries adaptively*. To achieve this, it unifies KG construction, hybrid retrieval, and adaptive reasoning into a single framework. Our contributions are four-fold:

1. We propose a **Hierarchical Knowledge Graph (HiGra)** construction pipeline with an entity alignment step that reduces redundant entities by 34.5%, yielding more compact graphs for retrieval.

2. We design **HiGraRetriever**, a hybrid retriever that integrates semantic and structural features, surpassing the strongest graph-based single-reasoning baseline on all benchmarks with a 3.7% average accuracy gain, and even exceeding the average accuracy of all iterative reasoning baselines.

3. We introduce **Dual-Agent Adaptive Reasoning**, a Seeker–Librarian iterative reasoning scheme that achieves 83.0% average accuracy with simple retrieval, improves over strongest baseline by 9.4%.

4. We develop the full system, **HiGraAgent**, which achieves the best results across all benchmarks with 85.3% average accuracy, a substantial 11.7% improvement over the second-best method.

2 Related Work

2.1 GraphRAG

GraphRAG systems involve two main steps: KG construction and retrieval.

2.1.1 KG Construction

KG construction primarily relies on *entity–relation extraction*. Systems such as **HippoRAG** (Gutiérrez et al., 2025a,b) use OpenIE-style methods, while more recent frameworks—including **GraphRAG** (Edge et al., 2025), **NodeRAG** (Xu et al., 2025), **LeanRAG** (Zhang et al., 2025), and **HiRAG** (Huang et al., 2025)—employ large language models (LLMs) to generate structured graphs. Beyond extraction, *summarization-based construction* is common: GraphRAG, NodeRAG, LeanRAG, **RAPTOR** (Sarthi et al., 2024), and HiRAG build hierarchical KGs by clustering passages or applying community detection. In contrast, **HopRAG** (Liu et al., 2025) structures graphs around pseudo-questions to guide multi-hop reasoning. Despite these advances, *entity redundancy* remains a challenge due to weak entity alignment, and summarization-based abstraction risks *incompleteness* by omitting non-entity-based answers.

2.1.2 KG Retrieval

Retrieval approaches fall into two categories: similarity-based and graph-based. Summarization-driven systems primarily rely on semantic similarity, while graph-based methods such as HippoRAG and NodeRAG exploit graph structure using Personalized PageRank (PPR). However, purely structural algorithms like PPR fail to capture semantic cues, particularly struggling with high-degree nodes. This highlights the need for hybrid strategies that combine semantic similarity with graph relationships. Other variants such as **T2RAG** (Gong et al., 2025), which focus exclusively on triplet similarity, can reduce context noise but risk omitting relevant evidence. Our framework addresses these issues by retrieving both triplets and passages from hierarchical knowledge levels, ensuring a better balance between completeness and precision.

2.2 Iterative Reasoning for Multi-hop QA

Multi-hop QA, particularly inference-oriented questions, often requires an *iterative discovery process*. **IRCoT** (Trivedi et al., 2023) first demonstrated the effectiveness of iterative fact gathering, and subsequent extensions such as **DualRAG** (Cheng et al., 2025) and **HANRAG** (Sun et al., 2025) formalized this idea into sequential reasoning pipelines. However, sequential flows are inherently rigid: they cannot easily accommodate *branching evidence paths*, *parallel exploration*, or *context-dependent query decomposition*. Our framework instead models reasoning as a dynamic graph, enabling parallel exploration across multiple paths and better capturing the adaptive nature of multi-hop reasoning.

2.3 Planning for Multi-hop QA

Prior multi-hop QA systems follow two paradigms: iterative retrieval–reasoning without explicit planning (e.g., IRCoT, HopRAG, DualRAG), and explicit question decomposition before retrieval (e.g., T2RAG, HANRAG). Beyond QA, decomposition has also been explored in general LLM reasoning (McDonald and Emami, 2024; Zhou et al., 2023). Our approach combines lightweight, grammar-aware decomposition with a dual-agent reasoning protocol that supports dynamic branching and recovery.

3 Hierarchical Knowledge Graph Construction

We propose HiGra, a bi-layer Hierarchical Knowledge Graph consisting of an Entity Layer and a Passage Layer. Instead of using summarization to build hierarchies, HiGra focuses on entity disambiguation, which is more crucial for retrieval performance. This design enables us to balance coverage with precision while exploiting entity relationships for improved knowledge access.

3.1 Phase 1: Entity and Relation Extraction

Coreference Resolution. Given a corpus of passages P, we first apply *coreference resolution* to form P'. For each passage $p_i \in P'$, we perform entity and relation extraction using a Large Language Model (LLM), yielding a tuple (E_i, R_i), where E_i is the set of entities and R_i is the set of relations found in p_i. The sets are then aggregated across all passages to obtain the global sets of entities and relations:

$$\mathcal{E} = \bigcup_{p_i \in P'} E_i \quad \text{and} \quad \mathcal{R} = \bigcup_{p_i \in P'} R_i$$

Entity-Relation Schema. Our approach to entity and relation extraction focuses on creating a richer, more descriptive schema than triplet-based methods. Each entity e in the set of all entities, $\mathcal{E}$, is represented by a tuple $e = (\mathrm{id}_e, N_e, A_e, D_e, T_e)$, where id_e is a unique identifier, N_e is the **Name**, A_e is the set of **Aliases**, D_e is a short **Description**, and T_e is a set of **Types**. This comprehensive schema, particularly the inclusion of aliases and types, is crucial for the downstream **Alignment** and **Retrieval** phases, which relies on these attributes for robust entity disambiguation. Similarly, each relation r in the set of all relations, $\mathcal{R}$, is defined by the tuple $r = (\mathrm{id}_s, \mathrm{N}_r, \mathrm{id}_t)$, which links a source entity (id_s) to a target entity (id_t) with a specific relation label (N_r).

3.2 Phase 2: Entity Merging

This phase performs entity alignment to merge duplicate entities in $\mathcal{E}$ that refer to the same real-world referent.

Exact-Match Merging. We first perform exact-match merging. Two entities $e_i, e_j \in \mathcal{E}$ are merged if they share a common name or alias and at least one common type. Let $S_e = \{N_e\} \cup A_e$ be the set of all names and aliases for an entity e. The

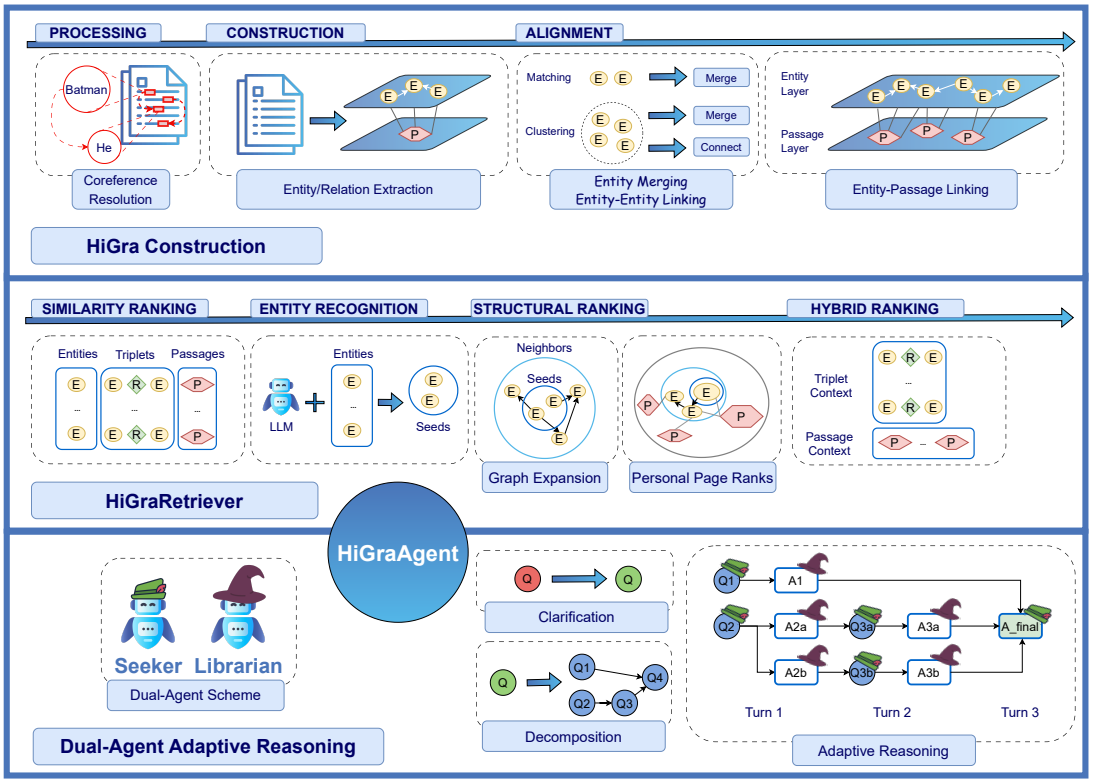

Figure 2: HiGraAgent overall architecture

merging condition is formally expressed as:

$$(S_{e_i} \cap S_{e_j} \neq \varnothing) \wedge (T_{e_i} \cap T_{e_j} \neq \varnothing)$$

Semantic Merging. We conduct semantic merging to address entities with non-identical names or aliases, such as "Stephen Curry" and "Steph Curry," and to resolve ambiguous names that require context, like "New York" (city vs. state). We first cluster entities using an embedding-based approach: $C = \text{Cluster}(\mathcal{E}, \phi, \tau)$, where ϕ is an embedding function and τ is the cluster radius. For each cluster $c \in C$, an LLM is prompted to identify merges within the cluster. The LLM's decision is based on a prompt that includes the entities' names, aliases, types, and descriptions.

3.3 Phase 3: Entity-Entity Linking

The relations extracted in the previous phase are often contextually-constrained, meaning they only exist if a passage explicitly mentions them. This can lead to a sparse knowledge graph, especially between entities that are semantically similar but never co-occur in the same document. For example, while separate passages might mention "Apple Inc." and "Apple Store," a relation between them may not be captured. This phase addresses that limitation by inferring missing relations, using a process similar to semantic merging where we first cluster the refined entities and then use an LLM to predict new, high-confidence relations.

3.4 Phase 4: Entity-Passage Linking

While initial entity-passage links are established during extraction, we create a more comprehensive mapping by finding additional links. This is because entity identification is often tied to the discovery of relations between two entities. Consequently, entities that are not part of any explicit relationship may not be identified during the initial pass, even if they are mentioned in a passage. This phase addresses that gap by searching for occurrences of each refined entity's name and aliases in all passages, subject to a set of constraints. We only create a link $L(e, p)$ if the number of occurrences of N_e in the entire corpus is below a threshold θ and N_e is not in the set of top K most common words $\mathcal{W}_{\text{topK}}$. This ensures a more precise mapping by preventing the creation of noisy links for overly common concepts like "Country" or "Of".

4 HiGra Retrieval

The HiGraRetriever algorithm operates in a sequential pipeline to identify and rank the most relevant graph relations and passages for a given query q.

The process uses a hybrid scoring mechanism that combines graph structure and semantic relevance.

4.1 Phase 1: Similarity Ranking

Prior to graph traversal, we obtain initial semantic ranks for relevant components using a hybrid retrieval (e.g., combining sparse and dense retrievers).

- **Entity Ranks ($\mathbf{Rank}_{\mathbf{E}}^{\mathbf{sim}}$)**: Ranks all entities $\mathcal{E}$ based on the entity's Name N_e, Aliases A_e, and Types T_e.
- **Relation Ranks ($\mathbf{Rank}_{\mathbf{R}}^{\mathbf{sim}}$)**: Ranks all relations $\mathcal{R}$ based on the triplet components (Source Entity Name N_{e_s}, Relation Label N_r, Target Entity Name N_{e_t}).
- **Passage Ranks ($\mathbf{Rank}_{\mathbf{P}}^{\mathbf{sim}}$)**: Ranks all passages $\mathcal{P}$ based on the passage text P.

4.2 Phase 2: Context-Aware Entity Recognition

We identify a refined set of **seed entities** $\mathcal{S}$ that serve as precise entry points for graph traversal. Unlike traditional methods that rely solely on entities explicitly mentioned in the query, our approach is *context-aware*, supporting a wider range of entity forms, including novel or abbreviated names. This process first filters the top-k candidates based on the $\mathbf{Rank}_{\mathbf{E}}^{\text{sim}}$, then use an LLM to select a precise set $\mathcal{S}$ for graph-based retrieval.

4.3 Phase 3: Subgraph Extraction

For a fixed radius r (e.g., $r = 1$), we extract a local subgraph $G_S = (E_S, R_S)$ centered around the seed entities $\mathcal{S}$. This subgraph includes all entities within a distance r from any seed node, where the distance $d_G(e, s)$ is the shortest path length on the Entity Layer.

$$E_S = \{e \in \mathcal{E} \mid \exists s \in \mathcal{S},\ d_G(e, s) \leq r\}$$

$$R_S = \{r \in \mathcal{R} \mid \mathrm{id}_s \in E_S, \mathrm{id}_t \in E_S\}$$

4.4 Phase 4: Structural Ranking

We use **Personalized PageRank (PPR)** to determine the structural importance of entities, relations, and passages within the relevant subgraph G_S. This phase calculates a structural score and rank ($\mathbf{Rank}^{\text{ppr}}$) for every component based on its connectivity to the initial $\mathcal{S}$.

The PPR calculation begins with a **personalization vector** π, which biases the random walk towards the seed nodes: $\pi(e) = \frac{1}{|\mathcal{S}|}$ if $e \in \mathcal{S}$, and 0 otherwise.

The stationary probability vector p (representing the structural scores for each node) is calculated iteratively at step t as:

$$p_t = \alpha M^\top p_{t-1} + (1 - \alpha)\pi$$

where M is the row-normalized adjacency matrix of G_S, and α is the damping factor.

Relation Structural Rank. The PPR scores are directly applied to the entities. To score relations, we average the entity scores. Each relation $r = (\mathrm{id}_s, \mathrm{id}_t) \in R_S$ is scored by averaging the PPR probabilities of its connected source and target entities:

$$p(r) = \frac{p(\mathrm{id}_s) + p(\mathrm{id}_t)}{2}$$

This score is then used to derive the **PPR-based rank, $\mathbf{Rank}_{\mathbf{R}}^{\mathbf{ppr}}(r)$**, for relations.

Passage Structural Rank. To score passages, we augment the subgraph G_S into a G_{aug}. This involves extending the graph to include all passage nodes $\mathcal{P}$ and the links $\mathcal{L}$ connecting them to entities in E_S. We then run PPR again on this augmented graph G_{aug} using the same personalization vector π. The resulting passage node scores $p(p)$ are used to derive the **PPR-based rank, $\mathbf{Rank}_{\mathbf{P}}^{\mathbf{ppr}}(p)$**, for each passage p.

4.5 Phase 5: Hybrid Ranking

The final ranking is achieved by fusing the structural ranks ($\mathbf{Rank}^{\text{ppr}}$) from Phase 4 with the initial semantic ranks ($\mathbf{Rank}^{\text{sim}}$) from Phase 1 using **Reciprocal Rank Fusion (RRF)**.

The RRF score for any component c (relation r or passage p) is calculated as:

$$\text{rrf}(c) = \frac{1}{k + \mathbf{Rank}^{\text{ppr}}(c)} + \frac{1}{k + \mathbf{Rank}^{\text{sim}}(c)}$$

For relations ($c = r$), $\mathbf{Rank}_{\mathbf{R}}^{\text{ppr}}$ and $\mathbf{Rank}_{\mathbf{R}}^{\text{sim}}$ are fused to yield R_{final}. For passages ($c = p$), $\mathbf{Rank}_{\mathbf{P}}^{\text{ppr}}$ and $\mathbf{Rank}_{\mathbf{P}}^{\text{sim}}$ are fused, resulting in $\mathbf{Rank}_{\mathbf{P}}^{\text{rrf}}$. The constant k stabilizes the fusion process.

4.6 Phase 6: Final Retrieved Context

The final output of the retrieval process consists of the top-ranked graph relations and passages.

We define the final set of ranked passages, $\mathbf{Rank}_{\mathbf{P}}^{\text{final}}$, by combining the top-$K$ passages from the RRF ranking and the top-K passages from the initial semantic ranking ($\mathbf{Rank}_{\mathbf{P}}^{\text{sim}}$).

$$\mathbf{Rank}_{\mathbf{P}}^{\text{final}} = \text{TopK}(\mathbf{Rank}_{\mathbf{P}}^{\text{rrf}}) \cup \text{TopK}(\mathbf{Rank}_{\mathbf{P}}^{\text{sim}})$$

The final retrieved context is then defined as the tuple of the top-K ranked relations from RRF and this combined set of passages:

$$\text{Context} = \left(\text{TopK}(\mathbf{Rank}_{\mathbf{R}}^{\text{rrf}}), \mathbf{Rank}_{\mathbf{P}}^{\text{final}}\right)$$

The design separates Phase 5 (Hybrid Ranking) from Phase 6 (Final Retrieved Context) to address complementary retrieval scenarios: Phase 5 prioritizes passages connected to seed nodes or their neighbors, while Phase 6 recovers highly similar yet unconnected passages, resulting in more stable performance than relying on seed-node connections alone.

5 Dual-Agent Adaptive Reasoning

Our approach, Dual-Agent Adaptive Reasoning, is a collaborative and highly transparent framework designed for complex, multi-hop QA. It operates through the dynamic interaction of two distinct LLM-based agents: the **Seeker** and the **Librarian**. The architecture specifically aims to address the compositional challenge of complex queries: the **Seeker** manages the composition of the query (decomposition and orchestration), while the Librarian manages the composition of the context (retrieval and local synthesis). This dynamic collaboration ensures a robust reasoning process, as the Seeker handles the global reasoning flow, the Librarian manages local reasoning contexts.

5.1 Agent Roles and Responsibilities

The Seeker. The Seeker is the primary LLM-based reasoning agent responsible for orchestrating the overall plan, breaking down the complex query, and executing actions based on the information it gathers.

The Librarian. A specialized LLM agent for retrieval and local synthesis. Given sub-questions, it extracts relevant contexts and drafts targeted answers. Operating statelessly, it relies on the Seeker to supply context, ensuring outputs are grounded in provided evidence and converting raw text into actionable knowledge.

Question Clarification. If a question does not begin with a question word, we apply a rewrite procedure using an LLM to explicitly clarify its objective and intent.

Question Decomposition. The clarified question is then decomposed into an initial reasoning plan by an LLM, enriched with dependency information between sub-questions. Further implementation details are provided in Appendix P.

5.2 The Adaptive Reasoning Protocol

The core of our methodology is the Adaptive Reasoning Protocol, an iterative loop that adaptively refines the problem decomposition and gathers information until a final, evidence-backed answer is formulated. The entire process is summarized in Algorithm 1.

Algorithm 1 Dynamic Reasoning Protocol

```
1:  procedure INVOKE(question)
2:      history ← []
3:      turn ← 0; done ← False
4:
5:      clarified_question ← CLARIFY(question)
6:      sub_q ← DECOMPOSE(clarified_question)
7:
8:      init_result ← LIBRARIAN(sub_q)
9:      history ← UPDATE(init_result)
10:
11:     while not done and turn < max_turns do
12:         context ← history, sub_q, question
13:         action ← SEEKER(context)
14:         result ← EXECUTE(action)
15:         done ← (action is "answer")
16:         history ← UPDATE(result)
17:         turn ← turn + 1
18:     end while
19:     return result, history
20: end procedure
```

This Adaptive Reasoning Protocol offers three key advantages over sequential methods. First, the **Librarian** component handles independent sub-questions in parallel (Line 8), optimizing initial evidence gathering time. Second, separating planning and execution allows the **Seeker** to dynamically decide the next action (Line 13), enabling handling of complex scenarios like branching reasoning paths or multiple valid answers. Finally, this dynamic loop provides robustness against failure: if an action yields no result, the Seeker can

adapt the strategy (e.g., rephrase or try a new path) instead of terminating.

6 HiGraAgent

While HiGraRetriever and Dual-Agent can each serve as stand-alone retrieval and reasoning modules, we propose HiGraAgent, which integrates the two into a unified adaptive reasoning RAG framework over structured knowledge. The overall architecture and workflow of all HiGraAgent modules are illustrated in Figure 2.

7 Experimental Setup

7.1 Datasets

We evaluate HiGraRetriever, Dual-Agent, and HiGraAgent in an open domain setting on three multi-hop QA benchmarks: **HotpotQA** (Yang et al., 2018), **2WikiMultihopQA** (Ho et al., 2020), and **MuSiQue** (answerable subset) (Trivedi et al., 2022). Following HippoRAG, we sample 1,000 questions from the *dev* split and merge all paragraphs (including distractors) into a single retrieval corpus per dataset. This setup simulates a realistic open domain retrieval scenario.

7.2 Baselines

We compare our method against the following baselines:

- **Simple Methods**: Parametric, which uses only the base LLM without retrieval, and NaiveRAG, which applies BM25, Dense Retriever, or Hybrid Retriever as basic approaches.
- **Single-Step Reasoning with Graph-based Retrieval**: HippoRAG2 (Gutiérrez et al., 2025b), NodeRAG (Xu et al., 2025), and HiRAG (Huang et al., 2025), representing recent advancements in graph-based retrieval methods.
- **Iterative Reasoning Methods**: IRCoT (Trivedi et al., 2023), HopRAG, T2RAG (Gong et al., 2025), and DualRAG (Cheng et al., 2025), representing recent methods for iterative reasoning.

7.3 Metrics

While F1 and EM are standard QA metrics, they often misrepresent performance by *overestimating* correctness (surface overlap) or *underestimating* it (format mismatches, multiple valid answers) (Schuff et al., 2020). We therefore adopt an **LLM-as-a-Judge** protocol (Gu et al., 2025), which more robustly evaluates both correctness and reasoning quality. F1 and EM scores are reported in Appendix I for completeness.

7.4 Implementation Details

We use **GPT-4o-mini** as the only language model for all tasks that require LLM-based reasoning. For dense retrieval, we adopt the widely used model **sentence-transformers/all-MiniLM-L6-v2** from HuggingFace, applying it uniformly across all evaluated methods to ensure fairness. For HiGraRetriever, we employ only the Librarian agent to conduct single-step reasoning. In contrast, for Dual-Agent methods, we rely on the Hybrid Retriever to provide the retrieval backbone.

8 Results and Analysis

Overall Performance. Table 1 summarizes the accuracy of our models and baselines across three datasets. All three of our proposed methods—HiGraRetriever, Dual-Agent, and HiGraAgent—achieve the highest average accuracy compared to baselines, with mean scores of 77.3%, 83.0%, and 85.3% respectively. The closest baseline, HiRAG, reaches 73.6% on average, resulting in a clear performance gap of 3.7%–11.7%.

HiGraRetriever. Our graph-based single-step reasoning module, HiGraRetriever, achieves 93.5% on HotpotQA, 81.5% on 2Wiki, and 57.0% on MuSiQue. It outperforms all other methods with single-step reasoning and demonstrates competitive performance compared to iterative methods. Further ablation results on retrieval item effects and ranking parameter robustness are provided in Appendix L and M.

Dual-Agent. Dual-Agent, which combines dual-agent iterative reasoning with simple unstructured retrieval, shows substantial gains on HotpotQA (92.0%) and MuSiQue (77.2%), and remains competitive on 2Wiki (79.8%), comparable to the best iterative baseline, DualRAG.

HiGraAgent. The full framework, HiGraAgent, consistently outperforms all baselines on each dataset, achieving 93.1% on HotpotQA, 85.1% on 2Wiki, and 77.8% on MuSiQue, resulting in the highest average accuracy of 85.3%. Compared to the strongest baseline on each dataset, HiGraAgent

Method	Reasoning	Retrieval	HotpotQA	2Wiki	MuSiQue	Avg
Parametric	Single	None	54.7	39.0	22.5	38.7
NaiveRAG (BM25)	Single	Unstructured	85.6	60.5	30.1	58.7
NaiveRAG (Dense)	Single	Unstructured	76.1	29.6	32.9	46.2
NaiveRAG (Hybrid)	Single	Unstructured	87.3	49.8	34.4	57.2
HippoRAG2	Single	Graph-based	84.5	69.4	48.8	67.6
NodeRAG	Single	Graph-based	84.4	43.8	46.0	58.1
HiRAG	Single	Graph-based	90.2	74.5	56.0	73.6
IRCoT	Iterative	Unstructured	90.2	66.4	34.3	63.6
HopRAG	Iterative	Graph-based	86.2	59.0	41.7	62.3
T2RAG	Iterative	Graph-based	47.6	38.5	32.8	39.6
DualRAG	Iterative	Unstructured	59.4	81.8	*72.6*	71.3
HiGraRetriever (Ours)	Single	Graph-based	**93.5**	*81.5*	57.0	*77.3*
Dual-Agent (Ours)	Iterative	Unstructured	*92.0*	79.8	77.2	83.0
HiGraAgent (Ours)	Iterative	Graph-based	93.1	**85.1**	**77.8**	**85.3**

Table 1: Accuracy (%) comparison across datasets. Best (1st) highlighted in bold, 2nd underlined, 3rd italic. Avg is the mean of HotpotQA, 2WikiMultihopQA, and MuSiQue.

Method	HotpotQA	2Wiki	MuSiQue	Avg. Relative Size (HiGra/Method)
HippoRAG2	94,516	51,655	98,584	69.1%
NodeRAG	103,131	55,734	111,338	62.9%
HiRAG	92,022	40,208	74,793	83.7%
T2RAG	96,944	50,296	101,782	68.4%
HiGra (Before Alignment)	98,790	53,719	106,627	65.5%
HiGra	63,659	36,779	67,741	100.0%

Table 2: Total number of entity nodes across datasets. We report the relative size of the HiGra with respect to each baseline (smaller is better).

surpasses HiRAG by 2.9% on HotpotQA (93.1% vs. 90.2%), DualRAG by 3.3% on 2Wiki (85.1% vs. 81.8%), and DualRAG by 5.2% on MuSiQue (77.8% vs. 72.6%), demonstrating a substantial performance gap across all benchmarks.

Entities Size Reduction. As shown in Table 2, our alignment step yields a substantial 34.5% reduction in graph size compared to HiGra before alignment. Moreover, HiGra (After Alignment) consistently produces more compact graphs than all baseline methods, with relative sizes ranging from 62.9% (vs. NodeRAG) to 83.7% (vs. HiRAG). Further ablation studies on the impact of reduced graph size on performance are provided in Appendix N.

9 Discussion

Dataset-specific challenges. The three benchmarks exhibit distinct characteristics. HotpotQA requires limited multi-hop reasoning (Min et al., 2019), allowing even parametric and simple retrieval-based methods (e.g., NaiveRAG) to achieve high accuracy. 2Wiki is entity-centric (Gutiérrez et al., 2025a), favoring graph-based single-step reasoning methods such as HiGraRetriever and HiRAG, though these methods struggle on MuSiQue. MuSiQue contains complex reasoning with multiple paths and answers, where iterative methods like HiGraAgent, Dual-Agent, and DualRAG surpass 72% accuracy. These results suggest that many baselines lack generalizability, which HiGraAgent successfully addresses. Moreover, we observe that iterative reasoning is not the most effective approach for HotpotQA, as evidenced by the poor performance of T2RAG and Du-

alRAG, while HiGraRetriever achieves the highest accuracy. These dataset-specific patterns motivate a closer examination of how graph-based retrieval contributes to robust reasoning across benchmarks.

Graph-based Retrieval Role. HiGraAgent is designed to address a dual compositionality challenge: query composition, which methods such as Dual-Agent handle effectively, and context composition, where supporting evidence is distributed across multiple, weakly related facts that cannot be reliably retrieved through similarity alone. In such settings, effective reasoning requires explicitly modeling relationships among retrieved entities rather than treating retrieval steps independently. Empirically, entity-centric benchmarks such as 2WikiMultiHopQA benefit substantially from graph-based context composition, where HiGraAgent achieves a 5.3% absolute improvement over the non-graph Dual-Agent baseline. Moreover, the single-step graph-based retriever (HiGraRetriever) can outperform the iterative retrieval strategy used in Dual-Agent, further reinforcing the importance of structured graph retrieval in entity-heavy reasoning scenarios.

Performance–Efficiency trade-off. Our framework exposes a flexible trade-off between accuracy and inference cost. HiGraRetriever achieves strong performance with low overhead, using only 4,107 tokens per query with substantially lower inference time, while outperforming all iterative reasoning baselines on entity-centric datasets. Dual-Agent improves robustness on compositionally challenging queries without requiring full graph construction, allowing slower inference only when needed. HiGraAgent attains the highest overall accuracy (85.3% average accuracy) among all methods, with its closest competitors being HiRAG and DualRAG. While HiGraAgent incurs higher inference time (26.1s/query) and token usage (25,119 tokens/query), it remains more token-efficient than HiRAG (26,399 tokens/query) and DualRAG (32,342 tokens/query). Notably, this efficiency comes alongside a substantial performance improvement over both baselines, demonstrating that HiGraAgent achieves a large accuracy gain while maintaining more effective token utilization. Rather than optimizing a single operating point, our design provides a spectrum of retrieval and reasoning strategies, enabling practitioners to balance performance and efficiency based on dataset characteristics. Detailed cost and latency breakdowns are provided in Appendix J and Appendix K.

10 Conclusion

We present HiGraAgent, a modular framework for multi-hop QA that integrates hierarchical knowledge graphs and adaptive dual-agent reasoning. Experiments demonstrate that HiGraRetriever, Dual-Agent, and the full HiGraAgent system achieve state-of-the-art performance across diverse benchmarks. HiGraAgent consistently outperforms strong baselines while producing more compact knowledge graphs, validating both its efficiency and robustness. The modular design allows flexible deployment: single-step retrieval for entity-centric tasks and iterative reasoning for complex queries. Our work illustrates the benefits of combining structured knowledge and adaptive reasoning in open domain multi-hop QA.

Limitations

Despite the promising results, several limitations remain in our current approach:

Backbone Sensitivity. A systematic study of different LLM and embedding backbones is an important next step, which we leave for future work due to current resource constraints.

Time and Token Efficiency. Agentic methods, including ours, inherently require more reasoning steps and LLM calls, leading to increased inference time and token usage. While this trade-off enables more sophisticated reasoning, optimizing for efficiency is an important direction for future work.

Generalization Across Tasks and Domains. Our evaluation is limited to the multi-hop QA setting. Validating the versatility of our approach requires further experiments across different reasoning tasks and knowledge-intensive domains (e.g., biomedical, legal).

Ethical Considerations

Our method only works with public datasets and does not involve training or data collection. Therefore, it poses no potential risks related to data privacy and others

Acknowledgements

This research is funded by Ho Chi Minh City University of Technology, Vietnam National University Ho Chi Minh City, under its research program, via the project "Explainable Artificial Intelligence System for Diagnosing the Severity of Knee Osteoarthritis Using Visual Prompting on MRI Images" (DS2025-20-05).

We also thank the anonymous reviewers for their thoughtful feedback and constructive suggestions, which helped improve the clarity and quality of this work.

References

Rong Cheng, Jinyi Liu, Yan Zheng, Fei Ni, Jiazhen Du, Hangyu Mao, Fuzheng Zhang, Bo Wang, and Jianye Hao. 2025. Dualrag: A dual-process approach to integrate reasoning and retrieval for multi-hop question answering. *Preprint*, arXiv:2504.18243.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *Preprint*, arXiv:2404.16130.

Shengbo Gong, Xianfeng Tang, Carl Yang, and Wei jin. 2025. Beyond chunks and graphs: Retrieval-augmented generation through triplet-driven thinking. *Preprint*, arXiv:2508.02435.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A survey on llm-as-a-judge. *Preprint*, arXiv:2411.15594.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2025a. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. *Preprint*, arXiv:2405.14831.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025b. From RAG to Memory: Non-Parametric Continual Learning for Large Language Models. *Preprint*, arXiv:2502.14802.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Haoyu Huang, Yongfeng Huang, Junjie Yang, Zhenyu Pan, Yongqiang Chen, Kaili Ma, Hongzhi Chen, and James Cheng. 2025. Hirag: Retrieval-augmented generation with hierarchical knowledge. *Preprint*, arXiv:2503.10150.

Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. 2025. HopRAG: Multi-Hop Reasoning for Logic-Aware Retrieval-Augmented Generation. *Preprint*, arXiv:2502.12442.

Vaibhav Mavi, Anubhav Jangra, and Adam Jatowt. 2024. Multi-hop question answering. *Preprint*, arXiv:2204.09140.

Tyler McDonald and Ali Emami. 2024. Trace-of-thought prompting: Investigating prompt-based knowledge distillation through question decomposition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 293–306, Bangkok, Thailand. Association for Computational Linguistics.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4249–4257, Florence, Italy. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *Preprint*, arXiv:2401.18059.

Hendrik Schuff, Heike Adel, and Ngoc Thang Vu. 2020. F1 is Not Enough! Models and Evaluation Towards User-Centered Explainable Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7076–7095, Online. Association for Computational Linguistics.

Duolin Sun, Dan Yang, Yue Shen, Yihan Jiao, Zhehao Tan, Jie Feng, Lianzhen Zhong, Jian Wang, Peng Wei, and Jinjie Gu. 2025. Hanrag: Heuristic accurate noise-resistant retrieval-augmented generation for multi-hop question answering. *Preprint*, arXiv:2509.09713.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2023. Knowledge graph prompting for multi-document question answering. *Preprint*, arXiv:2308.11730.

Tianyang Xu, Haojie Zheng, Chengze Li, Haoxiang Chen, Yixin Liu, Ruoxi Chen, and Lichao Sun. 2025. NodeRAG: Structuring Graph-based RAG with Heterogeneous Nodes. *Preprint*, arXiv:2504.11544.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Yaoze Zhang, Rong Wu, Pinlong Cai, Xiaoman Wang, Guohang Yan, Song Mao, Ding Wang, and Botian Shi. 2025. Leanrag: Knowledge-graph-based generation with semantic aggregation and hierarchical retrieval. *Preprint*, arXiv:2508.10391.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. *Preprint*, arXiv:2205.10625.

A Baseline Methods

For completeness, we briefly summarize the baseline methods considered in our experiments. These approaches cover both single-step and iterative reasoning, and include unstructured as well as graph-based retrieval.

HippoRAG2. HippoRAG2 (Gutiérrez et al., 2025b) enhances the original HippoRAG framework by extracting triplets and connecting synonyms through semantic similarity. Relevant passages and triplets are used to seed Personalized PageRank (PPR), enabling more effective multi-hop passage retrieval.

NodeRAG. NodeRAG (Xu et al., 2025) constructs a heterogeneous graph with semantic units, entities, relationships, attributes, and chunks. Retrieval combines exact matching on title nodes with vector search on rich information nodes, further refined by HNSW and shallow PPR.

HiRAG. HiRAG (Huang et al., 2025) introduces hierarchical indexing (HiIndex), where higher-level summary entities generated via clustering and LLMs connect semantically related entities across layers. Its retrieval (HiRetrieval) integrates local descriptions, global community reports, and shortest-path reasoning to improve evidence aggregation.

IRCoT. IRCoT (Trivedi et al., 2023) alternates between reasoning and retrieval: given an initial set of documents, the model iteratively generates chain-of-thought (CoT) steps and issues new retrieval queries. The process stops once an answer is generated or a maximum step limit is reached.

HopRAG. HopRAG (Liu et al., 2025) constructs a passage-level graph by simulating pseudo-queries and merging edges. Its retrieval–reason–prune pipeline combines similarity-based retrieval, reasoning-augmented traversal, and a novel "Helpfulness" metric for pruning irrelevant passages.

T2RAG. T2RAG (Gong et al., 2025) transforms the corpus into a knowledge base of canonical triplets and indexes their verbalized forms. Iterative

retrieval is driven by structured triplet queries with placeholders, allowing adaptive and fine-grained multi-hop evidence acquisition.

DualRAG. DualRAG (Cheng et al., 2025) integrates Reasoning-augmented Querying (RaQ) with Progressive Knowledge Aggregation (pKA), maintaining a dynamic "Knowledge Outline." This closed-loop process enables iterative refinement of reasoning and retrieval for multi-hop QA.

B Artifact Documentation

We use three publicly available datasets in our work: HotpotQA, 2WikiMultihopQA, and MuSiQue.

B.1 Citation of Artifact Creators

- **HotpotQA** (Yang et al., 2018) – accessed from https://hotpotqa.github.io/, licensed under CC BY-SA 4.0.
- **2WikiMultihopQA** (Ho et al., 2020) – accessed from https://github.com/Alab-NII/2wikimultihop, licensed under Apache License 2.0.
- **MuSiQue** (Trivedi et al., 2022) – accessed via Hugging Face at https://huggingface.co/datasets/dgslibisey/MuSiQue, licensed under CC BY 4.0.

B.2 Dataset Characteristic

We summarize basic characteristics of each dataset:

- **HotpotQA**: English language, Wikipedia domain, multi-hop questions with supporting sentences.
- **2WikiMultihopQA**: English language, based on reasoning over Wikidata triples.
- **MuSiQue**: English language, derived from diverse web sources with fine-grained multi-hop QA construction.

Further documentation and dataset cards are available via the original dataset repositories.

B.3 Dataset Sampling

Dataset	2-hop	3-hop	4-hop	5+ hop
HotpotQA	550	300	121	29
2WikiMultihop	500	2	496	2
MuSiQue	550	350	100	0

Table 3: Distribution of sampled questions by hop count for each dataset.

We strategically select evaluation samples from each dataset's development set to construct a maximally diverse test set. Our sampling procedure follows two key criteria: (1) we exclude questions with repeated answers (excluding binary *yes/no* cases) to avoid redundancy and better evaluate cross-model consistency; and (2) we stratify samples by *hop count*, defined as the number of gold passages required to answer a question in the original dataset.

Due to distributional imbalances introduced by criterion (1), the number of selected samples at each hop count is constrained by the remaining available questions. Table 3 summarizes the resulting hop count distribution across datasets. Note that MuSiQue includes up to 4-hop questions, with no instances beyond that.

B.4 Intended Use

Our use of each dataset adheres to the original intended research purposes. The datasets are used solely for academic research. No models trained on these datasets are deployed in production or used commercially. Any derived models or outputs remain within a research scope and comply with the access terms of the original datasets.

B.5 Privacy and Anonymization

The datasets are publicly released for research and do not contain personally identifiable information. We reviewed a representative sample from each dataset and did not find offensive or sensitive content requiring redaction. No additional anonymization steps were necessary. No new data was collected.

C Computational Experiments Report

C.1 Computational Budget

All methods, including ours, were executed without GPU computation. This is because the primary language model used across all methods is

GPT-4o-mini, which relies on API calls rather than local computation. The total estimated cost for development, baseline runs, and final experiments is approximately $990.

C.2 Hyperparameter Tuning

We adopted an iterative approach to optimize development costs using progressively larger subsets of the training data from each benchmark. The final evaluation on the complete open domain test corpus was conducted only once, without any hyperparameter tuning.

Table 4: Hyperparameter configuration for HiGraAgent

HiGraRetriever	**Value**
Ontology top-k	20
Entity–passage top-k	10
Similarity–passage top-k	5
Prune top-k	10
PPR α	0.85
RRF semantic constant	10
RRF PPR constant	10
Passage top-k (Hybrid Retrieval)	20
LLM Client	**Value**
Model name	gpt-4o-mini
Batch size (parallel limit)	200

D Implementation Details

This section outlines the software dependencies and libraries utilized in our implementation.

Coreference resolution. Coreference resolution is implemented using fastcoref[2] library.

Semantic clustering. We leverage scikit-learn DBSCAN[3] implementations to cluster word embeddings into candidates pool.

Question Decomposition. For linguistic extraction, we use the spaCy[4] model for dependency parsing.

BM25 Retrieval. The rank_bm25[5] library is used as the main implementation for the sparse retrieval module.

E Evaluation Fairness Discussion

Comparing multi-hop retrieval-augmented generation (RAG) systems developed under different experimental settings poses challenges for fair evaluation. We take several steps to ensure that our comparisons are consistent and faithful to the original methods.

For each baseline, we verify our implementations against the originally reported results. The reproduced performance is generally close to the original numbers, indicating faithful reproduction. In some cases, higher performance is observed, which we attribute to improvements in the underlying language models rather than changes to the retrieval or reasoning logic.

Some baselines rely on proprietary language models that are no longer publicly available. In particular, IRCoT was originally built on GPT-3; we therefore re-implement it using GPT-4o-mini to maintain a unified backbone across all methods. For all baselines, we use the authors' released code without modifying their core algorithmic components. These methods are largely model-agnostic and implemented using standard libraries, allowing the language model to be substituted without affecting overall methodology.

While exact replication of historical environments is not always possible, standardizing the backbone model and preserving the original algorithmic structure ensures a fair and meaningful comparison across all evaluated baselines.

F Retrieval Metrics Discussion

While retrieval-specific metrics such as recall@k are often informative, applying them consistently across prior work is challenging due to substantial differences in retrieval granularity. Existing baselines retrieve heterogeneous units, including passages, entities, summaries, community- or path-level descriptions, and structured triplets, for which no shared gold-standard retrieval target exists. Our method similarly retrieves mixed units (passages and triplets), further complicating direct comparison. Consequently, we focus on end-to-end QA performance, which provides a more faithful and comparable evaluation across structurally diverse retrieval paradigms.

G Qualitative Analysis

To demonstrate the effectiveness of the HiGraAgent framework, we analyze several real examples. Figures 3, 4 and 5 illustrate complex reasoning paths from the 2WikiMultihopQA and MuSiQue benchmarks. By enabling parallel question pro-

[2] https://github.com/shon-otmazgin/fastcoref

[3] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

[4] https://spacy.io

[5] https://github.com/dorianbrown/rank_bm25

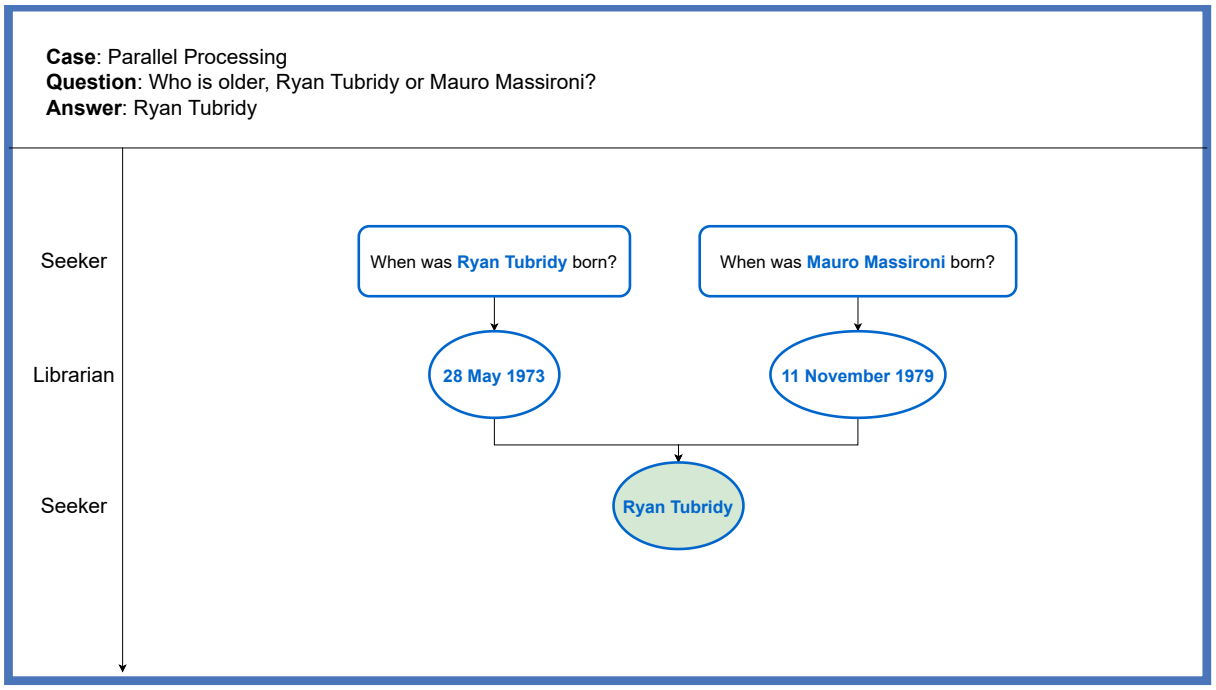

Figure 3: Dual-Agent Adaptive Reasoning Example 1: Parallel Processing

cessing and maintaining separate dialogue histories for the Seeker and Librarian, our framework can effectively manage intricate reasoning graphs.

Figure 3 shows a parallel processing case: "Who is older, Ryan Tubridy or Mauro Massironi?" HiGraAgent decomposes this into independent birthdate queries, resolves them concurrently via the Librarian, and integrates the results at the Seeker to derive the correct answer ("Ryan Tubridy"). This highlights the framework's ability to exploit parallelism for efficiency while maintaining reasoning accuracy.

Figure 4 shows a question where a single sub-question leads to multiple possible reasoning branches. Instead of committing early, HiGraAgent explores all candidate paths concurrently, ensuring that no potentially relevant evidence is overlooked before converging on the correct answer ("Dathan").

Figure 5 demonstrates adaptiveness at the answer stage: the query admits multiple valid final answers. HiGraAgent systematically explores and verifies these alternatives, ultimately producing a more complete answer set ("Lani Hall"), which better aligns with user expectations.

Together, these examples highlight HiGraAgent's ability to adapt its reasoning strategy to the structure of the problem—branching when intermediate ambiguity arises and expanding when multiple answers are plausible—leading to more reliable and user-centered outcomes.

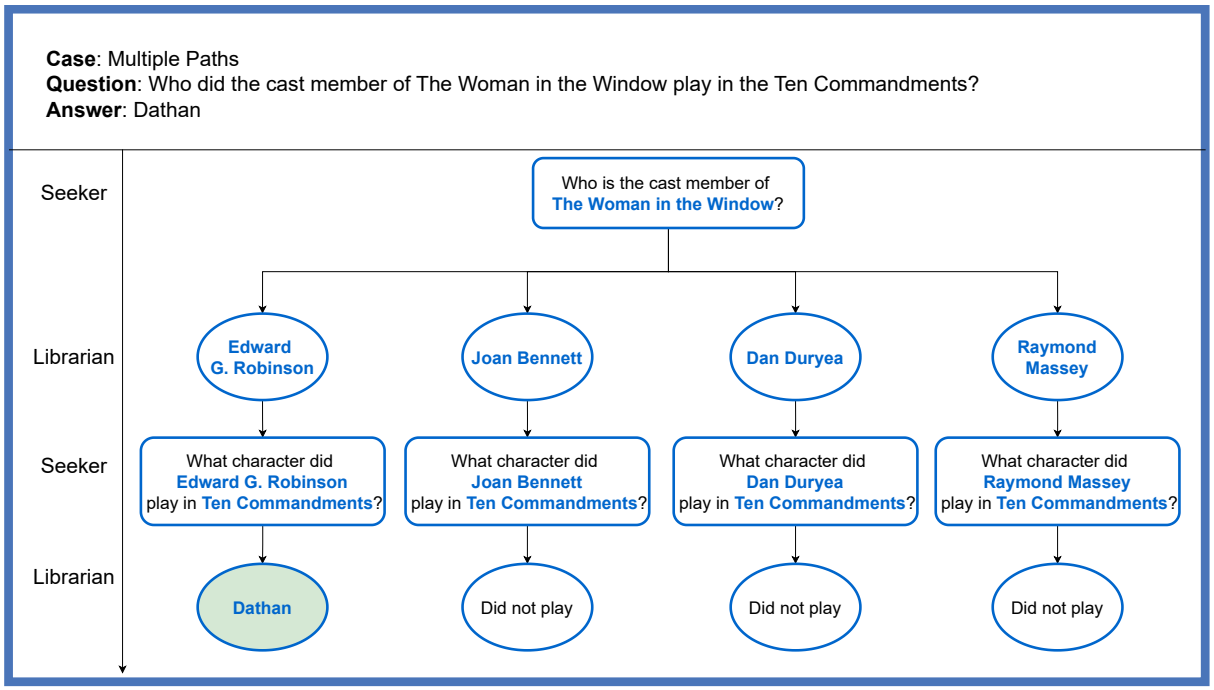

Figure 4: Dual-Agent Adaptive Reasoning Example 2: Multiple Reasoning Paths

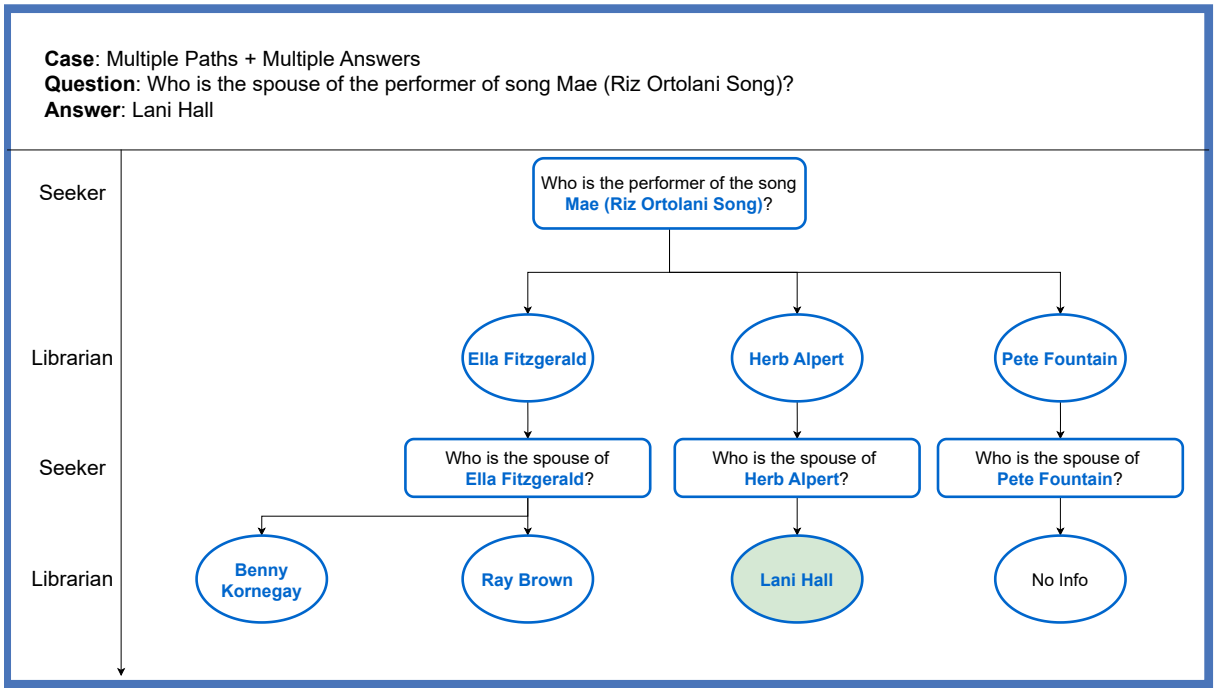

Figure 5: Dual-Agent Adaptive Reasoning Example 3: Multiple Reasoning Paths + Multiple Possible Answers

H Performance by Hop-counts Analysis

We further analyze model performance across different hop counts. Tables 5, 6, and 7 report accuracies by hop level for HotpotQA, 2WikiMultihopQA, and MuSiQue, respectively. Following the sample distribution in Table 3, we only consider hop counts with more than 50 instances to ensure a reliable evaluation basis.

Our methods deliver strong performance across different hop counts. On HotpotQA, HiGraAgent achieves 94.5% on 2-hop and sustains 90.1% on 4-hop, clearly ahead of NodeRAG (81.0%) and competitive against HopRAG (90.9%). On 2WikiMultihopQA, HiGraAgent reaches 88.2% (2-hop) and 82.5% (4-hop), outperforming HiRAG (73.6% / 75.8%). On MuSiQue, where baselines often collapse below 40% on higher hops, Dual-Agent and HiGraAgent maintain around 70% on 3-hop and above 50% on 4-hop, surpassing the strongest baseline DualRAG (64.9% / 52.0%).

These results show that our methods not only excel on easier 2-hop cases but also sustain high accuracy on challenging 3–4 hop reasoning, proving their robustness and effectiveness for multi-hop QA.

Method	**2-hop**	**3-hop**	**4-hop**
Parametric	57.8	51.7	51.2
NaiveRAG(BM25)	85.1	85.7	88.4
NaiveRAG(Dense)	78.0	74.0	77.7
NaiveRAG(HyBrid)	88.0	87.0	87.6
HippoRAG2	85.6	82.7	82.6
NodeRAG	85.6	80.3	81.0
HiRAG	91.3	90.0	89.3
IRCoT	84.7	87.3	86.8
HopRAG	86.0	85.0	90.9
T2RAG	50.4	47.0	38.8
DualRAG	64.2	52.7	55.4
HiGraRetriever	94.0	88.3	94.2
Dual-Agent	92.2	93.0	93.3
HiGraAgent	94.5	91.7	90.1

Table 5: HotpotQA accuracy by hop count.

Method	**2-hop**	**4-hop**
Parametric	35.2	42.9
NaiveRAG(BM25)	58.4	63.1
NaiveRAG(Dense)	42.0	17.3
NaiveRAG(HyBrid)	59.2	40.7
HippoRAG2	72.4	66.7
NodeRAG	52.2	35.7
HiRAG	73.6	75.8
IRCoT	64.6	66.9
HopRAG	59.2	59.1
T2RAG	30.4	46.8
DualRAG	79.8	84.3
HiGraRetriever	78.2	76.6
Dual-Agent	87.8	72.5
HiGraAgent	88.2	82.5

Table 6: 2WikiMultihopQA accuracy by hop count.

Method	**2-hop**	**3-hop**	**4-hop**
Parametric	24.5	22.0	10.0
NaiveRAG(BM25)	37.1	22.6	18.0
NaiveRAG(Dense)	39.1	25.4	25.0
NaiveRAG(HyBrid)	43.3	24.0	22.0
HippoRAG2	58.4	39.7	28.0
NodeRAG	52.5	35.4	35.0
HiRAG	66.7	45.7	33.0
IRCoT	37.8	24.9	22.0
HopRAG	46.7	35.7	35.0
T2RAG	35.8	30.3	25.0
DualRAG	81.3	64.9	52.0
HiGraRetriever	68.5	46.3	30.0
Dual-Agent	86.4	70.6	52.1
HiGraAgent	87.1	70.6	52.0

Table 7: MuSiQue accuracy by hop count.

I F1 and EM Evaluation

For completeness, we additionally report F1 and EM metrics (Tables 8, 9). Following HiRAG (Huang et al., 2025), we extract the short-form answer from the final predicted answer across all methods, which provides a consistent basis for F1 and EM evaluation.

The results show trends broadly consistent with overall accuracy: our proposed HiGraRetriever, Dual-Agent, and HiGraAgent achieve the strongest average performance across datasets. While absolute values are sometimes lower than accuracy, this mainly arises from cases with multiple valid answer expressions. In such cases, predictions are considered correct in accuracy but may not receive full credit under F1 or EM, which rely on exact or token-level overlap.

J Speed

We further analyze inference time, measured in seconds per question (Table 10).

For single-step reasoning methods, our HiGraRetriever requires on average 14.0s, which is slower than HiRAG (8.7s), NodeRAG (5.5s) and HippoRAG2 (2.6s). This trade-off arises because HiGraRetriever uses two LLM calls (entity recognition + final answering), whereas HiRAG relies on a heavier graph expansion process with a single answering step. Importantly, HiGraRetriever still achieves exceptional accuracy while maintaining a moderate runtime.

For iterative reasoning methods, our Dual-Agent (22.8s) and HiGraAgent (26.1s) are substantially faster than other strong baselines such as DualRAG (36.3s) and HopRAG (36.4s), while also delivering better overall performance (see Tables 1, 8 and 9). Although IRCoT (2.4s) and T2RAG (10.7s) run faster, their performance is significantly weaker across all benchmarks, reflecting the limited reasoning depth of their design.

K Tokens Consumption

We also compare the total prompt tokens consumed per question (Table 11).

For single-step reasoning methods, our HiGraRetriever uses on average 4,107.0 tokens, which is slightly higher than HippoRAG2 (1,356.7) and NodeRAG (2,367.7), due to the combination of graph structure and additional passage context. However, it is still an order of magnitude lower than HiRAG (26,399.5), showing that HiGraRetriever achieves strong accuracy with a far more efficient token budget.

For iterative reasoning methods, the trend mirrors the speed analysis. Our Dual-Agent (19,953.0) and HiGraAgent (25,119.1) consume substantially fewer tokens than DualRAG (32,342.8) and HopRAG (31,260.8), while being higher than lightweight but less effective approaches such as IRCoT (4,776.2) and T2RAG (2,620.8).

These results suggest that our designs balance reasoning strength with token efficiency, avoiding the prohibitive costs of heavy graph expansion (HiRAG, HopRAG, DualRAG) while still outperforming fast but weak baselines (IRCoT, T2RAG).

Method	Reasoning	Retrieval	HotpotQA	2Wiki	MuSiQue	Avg
Parametric	Single	None	42.6	37.6	12.9	31.0
NaiveRAG (BM25)	Single	Unstructured	73.0	59.6	24.5	52.4
NaiveRAG (Dense)	Single	Unstructured	65.2	31.6	26.1	40.9
NaiveRAG (Hybrid)	Single	Unstructured	73.7	50.4	28.0	50.7
HippoRAG2	Single	Graph-based	70.9	66.4	41.8	59.7
NodeRAG	Single	Graph-based	69.3	45.6	35.4	50.1
HiRAG	Single	Graph-based	73.6	73.7	44.0	63.8
IRCoT	Iterative	Unstructured	74.1	66.4	26.5	55.6
HopRAG	Iterative	Graph-based	74.7	58.6	38.0	57.1
T2RAG	Iterative	Graph-based	43.0	39.5	29.6	37.4
DualRAG	Iterative	Unstructured	48.7	78.1	56.8	61.2
HiGraRetriever (Ours)	Single	Graph-based	74.4	71.4	40.3	62.0
Dual-Agent (Ours)	Iterative	Unstructured	75.1	75.4	56.1	68.9
HiGraAgent (Ours)	Iterative	Graph-based	74.8	80.4	58.0	71.1

Table 8: F1 comparison across datasets. Average column is the mean of HotpotQA, 2WikiMultihopQA, and MuSiQue.

Method	Reasoning	Retrieval	HotpotQA	2Wiki	MuSiQue	Avg
Parametric	Single	None	27.9	31.7	2.7	20.8
NaiveRAG (BM25)	Single	Unstructured	56.6	55.4	14.8	42.3
NaiveRAG (Dense)	Single	Unstructured	50.7	28.5	18.2	32.5
NaiveRAG (Hybrid)	Single	Unstructured	57.5	47.6	18.6	41.2
HippoRAG2	Single	Graph-based	52.4	59.4	24.8	45.5
NodeRAG	Single	Graph-based	52.0	40.8	19.9	37.6
HiRAG	Single	Graph-based	55.3	65.7	26.8	49.3
IRCoT	Iterative	Unstructured	50.1	62.6	16.7	43.1
HopRAG	Iterative	Graph-based	58.6	55.0	27.3	47.0
T2RAG	Iterative	Graph-based	31.0	36.4	18.6	28.7
DualRAG	Iterative	Unstructured	34.6	71.7	39.8	48.7
HiGraRetriever (Ours)	Single	Graph-based	58.4	64.2	26.3	49.6
Dual-Agent (Ours)	Iterative	Unstructured	58.8	69.0	38.9	55.6
HiGraAgent (Ours)	Iterative	Graph-based	57.4	73.6	39.0	56.7

Table 9: EM comparison across datasets. Average column is the mean of HotpotQA, 2WikiMultihopQA, and MuSiQue.

Method	Reasoning	Retrieval	HotpotQA	2Wiki	MuSiQue	Avg
Parametric	Single	None	0.9	0.9	0.9	0.9
NaiveRAG (BM25)	Single	Unstructured	1.6	1.4	1.4	1.5
NaiveRAG (Dense)	Single	Unstructured	1.6	1.3	1.1	1.3
NaiveRAG (Hybrid)	Single	Unstructured	2.5	1.3	1.1	1.6
HippoRAG2	Single	Graph-based	2.4	2.8	2.7	2.6
NodeRAG	Single	Graph-based	5.4	5.3	5.7	5.5
HiRAG	Single	Graph-based	7.6	6.4	12.0	8.7
IRCoT	Iterative	Unstructured	2.2	2.5	2.4	2.4
HopRAG	Iterative	Graph-based	28.5	35.6	45.2	36.4
T2RAG	Iterative	Graph-based	10.1	12.4	9.6	10.7
DualRAG	Iterative	Unstructured	38.6	29.1	41.3	36.3
HiGraRetriever (Ours)	Single	Graph-based	15.7	11.6	14.6	14.0
Dual-Agent (Ours)	Iterative	Unstructured	23.4	19.9	25.0	22.8
HiGraAgent (Ours)	Iterative	Graph-based	24.7	22.7	31.0	26.1

Table 10: Speed comparison across datasets (seconds per question).

Method	Reasoning	Retrieval	HotpotQA	2Wiki	MuSiQue	Avg
Parametric	Single	None	56.9	52.2	55.0	54.7
NaiveRAG (BM25)	Single	Unstructured	2,909.7	3,586.2	2,909.7	3,135.2
NaiveRAG (Dense)	Single	Unstructured	3,409.6	3,101.5	3,046.4	3,185.8
NaiveRAG (Hybrid)	Single	Unstructured	3,499.2	3,538.3	3,213.7	3,417.1
HippoRAG2	Single	Graph-based	1,336.2	1,360.2	1,373.7	1,356.7
NodeRAG	Single	Graph-based	2,402.4	2,399.9	2,300.7	2,367.7
HiRAG	Single	Graph-based	26,793.4	26,972.1	25,433.1	26,399.5
IRCoT	Iterative	Unstructured	3,838.1	6,121.5	4,369.1	4,776.2
HopRAG	Iterative	Graph-based	25,857.7	31,743.1	36,181.6	31,260.8
T2RAG	Iterative	Graph-based	2,564.3	2,861.2	2,436.9	2,620.8
DualRAG	Iterative	Unstructured	21,528.8	31,445.3	44,054.2	32,342.8
HiGraRetriever (Ours)	Single	Graph-based	3,873.9	4,491.5	3,955.5	4,107.0
Dual-Agent (Ours)	Iterative	Unstructured	18,394.0	21,202.4	20,262.7	19,953.0
HiGraAgent (Ours)	Iterative	Graph-based	22,338.9	27,447.7	25,570.8	25,119.1

Table 11: Total prompt tokens per question comparison across datasets. Average column is the mean of HotpotQA, 2WikiMultihopQA, and MuSiQue.

L Ablation Study: Effect of Retrieval Items

Method / Setting	HotpotQA	2Wiki	MuSiQue
HiGraRetriever			
Graph + Passage	93.5	81.5	57.0
Passage Only	92.2	77.1	56.9
HiGraAgent			
Graph + Passage	93.1	85.1	77.8
Passage Only	91.2	85.0	77.5

Table 12: Ablation study comparing Graph + Passage vs Passage Only retrieval.

To assess the effectiveness of combining both graph and passage context, we conduct an ablation study on HiGraRetriever and HiGraAgent (Table 12).

For HiGraRetriever, removing graph information and relying only on passages leads to a notable performance drop (e.g., 81.5% $\rightarrow$ 77.1% on 2Wiki), highlighting the importance of structural signals in retrieval.

For HiGraAgent, the performance gap between the two settings is narrower. The main difference appears in HotpotQA (93.1% vs. 91.2%), while results on 2Wiki and MuSiQue remain comparable. This can be attributed to the adaptive reasoning design of HiGraAgent, where the Seeker can flexibly explore alternative reasoning paths even when graph context is absent.

Overall, these results confirm that graph information is especially beneficial for retrieval-focused settings, while adaptive multi-step reasoning can mitigate its absence to some extent.

M Ablation Study: Ranking Combination Robustness

Our ranking combination integrates multiple graph-derived signals, including PageRank-based connectivity (PPR) and relevance scores from semantic and structural components (RFF). The design goal is to balance these factors such that neither graph connectivity nor semantic relevance dominates, enabling robust retrieval across datasets.

To evaluate the sensitivity of this design, we conduct an ablation study varying three key hyperparameters: (1) the Personalized PageRank restart probability α, which controls random walk persistence; (2) the weight assigned to the semantic relevance component in RFF; and (3) the weight assigned to graph-based signals. We report accuracy under different configurations in Table 13.

Setting	Hotpot	2Wiki	MuSiQue
HiGraRetriever			
Default	93.5	81.5	57.0
PPR α=0.7	91.8	82.1	58.4
PPR α=0.5	94.5	82.7	58.3
RFF semantic=1	93.2	79.5	57.2
RFF graph =1	93.9	84.3	56.6

Table 13: Ablation study on ranking combination hyperparameters for HiGraRetriever.

As shown in Table 13, performance remains stable under moderate variations of all hyperparameters. Lowering the PPR continuation probability does not substantially affect retrieval accuracy, and adjusting the balance between semantic and graph-based relevance results in only minor fluctuations across datasets. No configuration leads to a significant performance degradation relative to the default setting.

These results support the robustness of our ranking combination design, demonstrating that it effectively balances multiple retrieval signals and maintains stable performance across different graph and semantic weighting strategies.

N Ablation Study: Effect of Graph Refinement

This ablation study examines the role of graph refinement in our graph-based retrieval pipeline. The refinement procedure is not intended to reduce inference-time latency directly, but to improve the quality and usability of the constructed knowledge graph for downstream retrieval and reasoning.

Our work differs from prior GraphRAG approaches that rely on retrieval-based seed selection and intermediate summarization. Instead, we investigate whether a detection-based seed selection strategy combined with a purely graph-structured context (without summarization) can be effective. In this setting, raw graphs exhibit severe entity redundancy, which substantially degrades both seed detection and graph traversal. Graph refinement mitigates this issue by merging redundant entities and producing a more coherent structure.

To isolate the effect of refinement, we evaluate HiGraRetriever with and without graph refinement. As shown in Table 14, removing refinement leads

to consistent performance degradation across all datasets, with particularly large drops on 2Wiki and MuSiQue. This indicates that the primary benefit of refinement lies in improving graph quality rather than computational efficiency.

Dataset	Original	With Refinement
HotpotQA	92.1	93.5
2Wiki	66.3	81.5
MuSiQue	46.3	57.0

Table 14: Accuracy of HiGraRetriever with and without graph refinement.

In addition to performance gains, refinement substantially reduces graph fragmentation. Table 15 reports the number of connected components before and after refinement, showing an order-of-magnitude reduction across datasets. This produces denser and more coherent graphs, which are more suitable for graph-based retrieval.

Dataset	Original	With Refinement
HotpotQA	11,595	1,123
2Wiki	7,478	1,146
MuSiQue	14,737	1,818

Table 15: Number of connected components with and without graph refinement.

While graph size reduction alone does not directly translate to faster inference, it significantly improves retrieval effectiveness and graph coherence. Compared to summarization-heavy GraphRAG pipelines, our refinement process is also computationally lightweight, as it operates on short entity representations. These results highlight the importance of entity refinement for effective graph-based retrieval and suggest potential benefits for broader graph reasoning and knowledge construction tasks.

O HiGra Statistics

Dataset	**Entities**	**Relations**	**Passages**
HotpotQA	63,659	90,062	9,773
2Wiki	36,779	47,584	6,595
MuSiQue	67,741	96,208	11,185

Table 16: HiGra graph size statistics.

Dataset	**Entitiy Degree (min / avg / max)**
HotpotQA	0 / 2.8 / 514
2Wiki	0 / 2.6 / 180
MuSiQue	0 / 2.8 / 905

Table 17: Entity degree distribution in HiGra graphs.

Dataset	**Passages per Entity (min / avg / max)**
HotpotQA	1 / 2.11 / 542
2Wiki	1 / 1.8 / 189
MuSiQue	1 / 2.1 / 824

Table 18: Mapping of passages to entities in HiGra graphs.

To better understand the structural properties of HiGra across datasets, we report statistics of entity graphs in Tables 16–18. In terms of scale, MuSiQue yields the largest graph with 67K entities and 96K relations, while 2Wiki is the smallest with 37K entities and 48K relations.

Regarding connectivity, the average entity degree remains consistent across datasets (∼2.6–2.8), but maximum degrees vary significantly: MuSiQue reaches 905 due to highly central nodes such as *United States*, while HotpotQA peaks at 514 and 2Wiki at 180.

Finally, the passage mapping shows each entity is grounded in multiple contexts. On average, entities appear in 1.8–2.1 passages, with extreme cases such as United States linked to 542 passages in HotpotQA and 824 passages in MuSiQue. These statistics highlight the heterogeneous nature of multi-hop benchmarks and motivate the need for robust retrieval strategies in HiGra.

P Additional Algorithm Details

P.1 Question Clarification.

In open domain QA, natural language queries are often underspecified or expressed in ambiguous forms that complicate downstream reasoning. To address this, we employ a clarification protocol (Algorithm 2) that ensures questions are cast into a canonical form. Specifically, the procedure first verifies whether the input begins with a wh-question word; if not, it invokes a language model to rephrase the input into an explicit interrogative. This step improves the reliability of decomposition and reasoning modules. Example is illustrated below:

Algorithm 2 Clarify Question Protocol

```
procedure CLARIFY(q)
    q' ← q
    if not STARTWITHQUESTIONWORD(q) then
        q' ← LLM(q)
    end if
    return q'
end procedure
```

Original: *Brendon Urie wrote new perspective after firing which bassist?*

Clarified: *Which bassist did Brendon Urie fire after writing New Perspective?*

P.2 Question Decomposition

To enable multi-step reasoning, complex queries are decomposed into smaller sub-questions. As shown in Figure 6, the process first extracts linguistic features (e.g., relative clauses, "of"-structures, possessive constructions), which guide the language model in generating a structured decomposition plan. This allows the reasoning agent to resolve intermediate entities step by step.

Algorithm 3 Question Decomposition Protocol

```
procedure DECOMPOSE(q)
    features ← GETFEATURES(q)
    decomposed_q ← LLM(q, features)
    return decomposed_q
end procedure
```

Q Seeker Action Execution

The Seeker serves as the controller that executes reasoning actions based on the current reasoning state. As shown in Algorithm 4, each action is either directed to the Librarian (for retrieval and synthesis) or finalized as an answer with supporting evidence.

R Librarian Parallel Processing

The Librarian handles retrieval and synthesis in parallel. As described in Algorithm 5, given a list of sub-questions, the system first retrieves raw contexts from the knowledge base, then formats them into prompts. These prompts are processed in batched LLM calls, allowing efficient parallel execution. This design reduces latency while ensuring that each answer is grounded in its corresponding retrieved evidence.

Algorithm 4 Execute Action Protocol

```
procedure EXECUTE(action)
    if action.name is "ask_librarian" then
        q_list ← action.question_list
        return LIBRARIAN(q_list)
    else
        res ← action.answer
        evidence ← action.evidence
        return ANSWER(res, evidence)
    end if
end procedure
```

Algorithm 5 Librarian Executions

```
procedure LIBRARIAN(question_list)
    contexts ← RETRIEVE(question_list)
    prompts ← []
    for q, c in question_list, contexts do
        prompt ← FORMATPROMPT(q, c)
        Append prompt to prompts
    end for
    responses ← CALLBATCHED(prompts)
    return responses
end procedure
```

S Prompts

All prompts used in our framework are provided for transparency and reproducibility. Specifically, the Seeker and Librarian prompts are illustrated in Figures 7 and 8, respectively. Prompts for HiGra Construction, HiGra Alignment, Entity Recognition, and LLM-as-a-Judge are also included in https://github.com/headinthecloud6453/higra_agent.

Question: What is the name of the castle in the city where the performer of I'm Alive and on Fire was formed?

Question Linguistic Features


```
{
  "full_relative_clauses": [
    "the city where the performer of I'm Alive and on
Fire was formed"
  ],
  "reduced_relative_clauses": [],
  "of_structures": [
    "the name of the castle",
    "the performer of I"
  ],
  "possessive_s_structures": []
}
```


Question Decomposition


```
{
  "question_subject": "castle",
  "question_decomposition": [
    {
      "id": 1,
      "subquestion": "Who is the performer of I'm Alive and on Fire?",
      "depends_on": null
    },
    {
      "id": 2,
      "subquestion": "In which city was the performer from #1 formed?",
      "depends_on": 1
    },
    {
      "id": 3,
      "subquestion": "What is the name of the castle in the city from #2?",
      "depends_on": 2
    }
  ]
}
```


Figure 6: Question Decomposition Example

Seeker Prompt

Overview
- You are in a **Dual-Agent Question-Answering system** (Seeker + Librarian) built to solve complex, multi-hop questions

Task
Use dynamic, multi-step reasoning to navigate toward a correct, well-supported answer for the user's original question. Handle ambiguity, multiple possible answers, abstractions, and composition of knowledge across sources.

Tools
You must follow the exact syntax of these functions
1. `ask_librarian(question_list: List[str]) -> List[Dict]
  - The input requires parameter 'question_list', which is a list of questions.
  - Fetch relevant passages from the knowledge base for the (sub-)question.

 - **Preprocess** the question before calling: remove numbered decomposition tags (e.g., `#1`, `#2`) and avoid embedding your own assumptions. This means that your question in 'question_list' must strictly do not contain '#'
    Example:
      + Avoid: What is the father of #1?
      + Should: What is the father of A? (Assuming A was discovered in the previous question)
  - Always ensure the query provides sufficient context — `retrieve_knowledge` does not retain memory between calls.

 - If `retrieve_knowledge` returns **multiple candidate results**, record all of them and treat each as a distinct possibility (see *Multiple results* below).
  - Use both the **answer** and its **evidence** when reasoning. The answer alone may be too short or incomplete.
  - All factual claims must be verified with `ask_librarian`. Do not rely on internal knowledge.
  - For the final result, you must identify **all possible answers**. Explore exhaustively across all reasoning paths.

 - Continue asking follow-up questions until you are certain no additional answers exist (e.g., awards, record labels, people, etc.).
  - Each follow-up query must include the **full details** from the previous answers. Do not shorten or omit context.

 - Prioritize following the **question decomposition** steps by following the 'id', also follow the exact wording of the decomposed-questions.

2. `answer(answer, evidence) -> dict`
  - Use this only to **return the final answer** to the user.
  - The evidence must include all information to answer the original question.
  - Your answer must preserve exact wording with the evidences (numbers, names,...).
  - Your answer must contain a short rephrasing of the original question.

Figure 7: Seeker's Prompt

Librarian Prompt

Guide
- **Do Not Answer from Your Own Knowledge**: Do not make assumptions or rely on your own prior knowledge, as it may be outdated.
- **Not all contexts found are relevant**: Identify what is important and ignore noise.
- **You do not need a definitive evidence to answer**: Provide the best answer possible from what you have retrieved, or anything relevant, closed to the questions.

Multiple Answer Handling
Your questions are often multiple-answer in nature, often due to nondeterministic formulation. Identify and highlight all possible answers. "Multiple answers" could also mean multiple specific and granular levels.
Reasoning Guide: Identify and explain all reasonable interpretations and why they might differ, based only on the retrieved context.

Mismatching in Specificity and Granularity
Some questions and answers may differ in specificity. If retrieved knowledge does not match the specificity, the more general (or more specific) answer is acceptable, with explanation.

Complex Relationship Handling
When answering, if the exact information is not explicitly stated in the context, you may infer it from related details. Always explain when an answer is inferred.

Style
- Reply in **JSON ONLY**
- Answer in the full form, closest to the wording of the context.

Output Format
- Your output must be a JSON object:

```
{
  "answer": <answer if found, else answer any relevant information>,
  "evidence": <exact evidences in the context, or any relevant information that we found>,
}
```

Figure 8: Librarian's Prompt