

HyperGraphRAG: Retrieval-Augmented Generation with Hypergraph-Structured Knowledge Representation

Haoran Luo^{1,2}, Haihong E^{1*}, Guanting Chen¹, Yandan Zheng², Xiaobao Wu², Yikai Guo³, Qika Lin⁴, Yu Feng¹, Zemin Kuang⁵, Meina Song¹, Yifan Zhu¹, Luu Anh Tuan²

¹Beijing University of Posts and Telecommunications ²Nanyang Technological University

³Beijing Institute of Computer Technology and Application ⁴National University of Singapore

⁵Beijing Anzhen Hospital, Capital Medical University

{luohaoran, ehaihong, yifan_zhu}@bupt.edu.cn, anhtuan.luu@ntu.edu.sg

Abstract

While standard Retrieval-Augmented Generation (RAG) based on chunks, GraphRAG structures knowledge as graphs to leverage the relations among entities. However, previous GraphRAG methods are limited by binary relations: one edge in the graph only connects two entities, which cannot well model the n-ary relations among more than two entities that widely exist in reality. To address this limitation, we propose HyperGraphRAG, a novel hypergraph-based RAG method that represents n-ary relational facts via hyperedges, modeling the complicated n-ary relations in the real world. To retrieve and generate over hypergraphs, we introduce a complete pipeline with a hypergraph construction method, a hypergraph retrieval strategy, and a hypergraph-guided generation mechanism. Experiments across medicine, agriculture, computer science, and law demonstrate that HyperGraphRAG outperforms standard RAG and GraphRAG in accuracy and generation quality. Our code is publicly available¹.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Gao et al., 2024) has achieved remarkable progress in knowledge-intensive tasks by integrating information retrieval with large language models (LLMs) (OpenAI et al., 2024; Zhao et al., 2024), thereby enhancing factual awareness and generation accuracy. However, standard RAG frameworks predominantly rely on chunk-based retrieval mechanisms, where text is segmented into fixed-length passages and retrieved via dense vector matching. It fails to capture complex relationships between entities, resulting in poor knowledge modeling capabilities and low retrieval efficiency, constraining its applicability in fine-grained knowledge-driven applications (Pan et al., 2024).

* Corresponding author.

¹<https://github.com/LHRLAB/HyperGraphRAG>

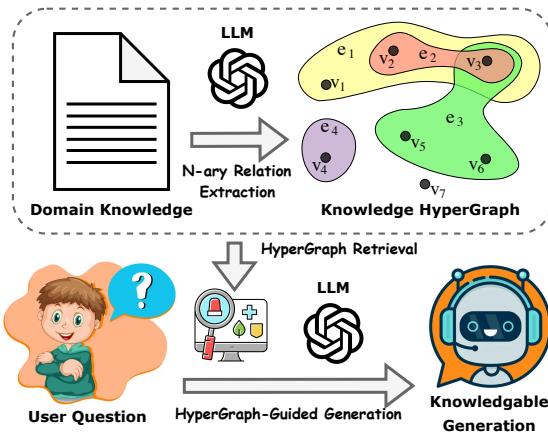


Figure 1: An illustration of graph-based RAG framework. Specifically, we introduce HyperGraphRAG with hypergraph-structured knowledge representation.

Recently, GraphRAG (Edge et al., 2024) has been introduced as an enhancement to conventional RAG by representing knowledge as a graph-based structure, improving retrieval precision. However, previous GraphRAG is restricted to binary relations, making it inadequate for modeling the prevalent n-ary relations among more than two entities found in real-world knowledge (Wen et al., 2016). For instance, in the medical domain, as illustrated in Figure 2, when modeling the fact that “*Male hypertensive patients with serum creatinine levels between 115–133 μmol/L are diagnosed with mild serum creatinine elevation*”, GraphRAG must decompose it into multiple isolated binary relations, such as *Gender:(Hypertensive patient, Male)* and *Diagnosed_with:(Hypertensive patient, Mild serum creatinine elevation)*. This decomposition leads to representation loss when transforming natural language facts into a structured knowledge graph.

To address these limitations, we propose HyperGraphRAG, as shown in Figure 1, a novel graph-based RAG method with hypergraph-structured knowledge representation. Unlike GraphRAG, which is restricted to binary relations, Hyper-

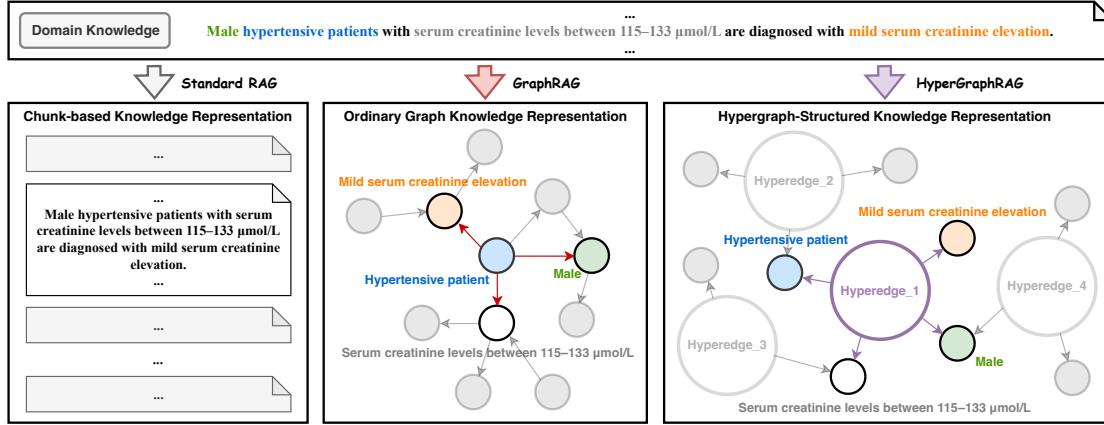


Figure 2: Comparison of knowledge representation of different RAG methods: standard RAG uses chunk as unit, GraphRAG captures binary relations with graphs, and HyperGraphRAG models n-ary relations with hyperedges.

GraphRAG employs hyperedges to represent n-ary relational facts, where each hyperedge connects n entities ($n \geq 2$), e.g. *Hyperedge:(Hypertensive patient, Male, Serum creatinine levels between 115–133 $\mu\text{mol/L}$, Mild serum creatinine elevation)*, and expresses each hyperedge in a natural language description. This approach improves knowledge completeness, structural expressiveness, and inferential capability, offering more comprehensive support for knowledge-intensive applications.

To implement HyperGraphRAG, we introduce several key innovations. First, we propose a knowledge hypergraph construction method, leveraging LLM-based n-ary relation extraction to extract and structure multi-entity relationships. The resulting hypergraph is stored in a bipartite graph database, with separate vector databases for entities and hyperedges to facilitate efficient retrieval. Second, we develop a hypergraph retrieval strategy that employs vector similarity search to retrieve relevant entities and hyperedges, ensuring that the knowledge retrieved is both precise and contextually relevant. Lastly, we introduce a hypergraph-guided generation mechanism, which combines retrieved n-ary facts with traditional chunk-based RAG passages, thereby improving response quality.

To validate the effectiveness of HyperGraphRAG, we conduct experiments in multiple knowledge-intensive domains (Guo et al., 2024), including medicine, agriculture, computer science, and law. The experimental results demonstrate that HyperGraphRAG significantly outperforms traditional RAG and GraphRAG in terms of factuality, comprehensiveness, diversity, and empowerment. Furthermore, HyperGraphRAG lays the foundation for a more comprehensive and

automated approach to knowledge representation, retrieval, and generation, showcasing its strong potential for real-world applications.

2 Related Work

Graph-based RAG. RAG (Lewis et al., 2020) enhances generative AI in knowledge-intensive tasks by integrating external knowledge retrieval. GraphRAG (Edge et al., 2024) improves LLM generation via graph-based retrieval, but incurs high construction costs. MedGraphRAG (Wu et al., 2024) and OG-RAG (Sharma et al., 2024) explore graph-based RAG in the medical domain and ontology-driven approaches, respectively. LightRAG (Guo et al., 2024) and MiniRAG (Fan et al., 2025) optimize retrieval efficiency with graph indexing and incremental updates, but remain constrained by binary relational representation, causing knowledge sparsity. In this work, we propose HyperGraphRAG, the first RAG method with hypergraph-based knowledge representation.

Hypergraph Representation. Hypergraph-structured knowledge representation aims to overcome the limitations of traditional knowledge graphs in modeling n-ary relations (Luo et al., 2024). The early methods (Wen et al., 2016; Zhang et al., 2018; Liu et al., 2020; Rosso et al., 2020) employed various embedding techniques to represent n-ary relational entities and relations. Later approaches (Galkin et al., 2020; Wang et al., 2021; Luo et al., 2023) utilized GNN or attention mechanisms to model n-ary relations. However, existing methods focus mainly on link prediction, which is promising in enhancing knowledge representation of graph-based RAG.

3 Preliminaries

We define three basic concepts of this work: RAG, Graph-based RAG, and Hypergraph, as follows:

Definition 1: RAG. Given a question q and a knowledge base K , standard RAG first selects relevant document fragments d from K based on q , and then produces an answer y based on q and d . The probability model is formulated as follows:

$$P(y|q) = \sum_{d \in K} P(y|q, d)P(d|q, K). \quad (1)$$

Definition 2: Graph-based RAG. Graph-based RAG optimizes retrieval by representing knowledge as a graph structure $G = (V, E)$, where V is the set of entities and E is the set of relationships between entities. G consists of facts represented as $F = (e, V_e) \in G$, where e is the relation and V_e is the entity set connected to e . Given a question q , the retrieval process is defined as:

$$P(y|q) = \sum_{F \in G} P(y|q, F)P(F|q, G). \quad (2)$$

Definition 3: Hypergraph. A hypergraph $G_H = (V, E_H)$ (Zhou et al., 2006) is a generalized form of a graph, where V is the set of entities, and E_H is the set of hyperedges. Each hyperedge $e_H \in E_H$ connects two or more entities:

$$V_{e_H} = (v_1, v_2, \dots, v_n), \quad n \geq 2. \quad (3)$$

Unlike ordinary graphs, where relationships are binary $V_e = (v_h, v_t)$, hypergraphs model n-ary relational facts $F_n = (e_H, V_{e_H}) \in G_H$.

4 Methodology

In this section, we introduce HyperGraphRAG, as shown in Figure 3, including knowledge hypergraph construction, hypergraph retrieval strategy, and hypergraph-guided generation.

4.1 Knowledge Hypergraph Construction

To represent and store knowledge, we propose a knowledge hypergraph construction method with n-ary relational extraction, bipartite hypergraph storage, and vector representation storage.

4.1.1 N-ary Relation Extraction

To construct the knowledge hypergraph G_H , we first need to extract multiple n-ary relational facts F_n from natural language documents $d \in K$. We introduce the proposed representation structure and LLM-based extraction process as follows:

Representation Structure. Unlike traditional hyper-relations (Rosso et al., 2020), events (Lu et al., 2021), or other n-ary relations (Luo et al., 2024), in the era of LLMs, we utilize natural language description to represent relationships among multiple entities. This approach preserves the complexity and diversity of n-ary relationships while ensuring the completeness and flexibility of knowledge representation. Based on our designed n-ary relation representation $F_n = (e_H, V_{e_H})$, each fact consists of a hyperedge e_H and multiple entities V_{e_H} , with the following structure:

- **Hyperedge:** Given an input text d , the text is parsed and segmented into several independent knowledge fragments. Each knowledge fragment is considered as a hyperedge: $E_H^d = \{e_1, e_2, \dots, e_k\}$, where each hyperedge $e_i = (e_i^{\text{text}}, e_i^{\text{score}})$ consists of two parts: the natural language description e_i^{text} , and the confidence score $e_i^{\text{score}} \in (0, 10]$ indicating the association strength of this hyperedge e_i with the original text d .
- **Entity:** For each hyperedge e_i , entity recognition is performed to extract all contained entities: $V_{e_i} = \{v_1, v_2, \dots, v_n\}$, where V_{e_i} is the set of entities associated with e_i . Each entity $v_j = (v_j^{\text{name}}, v_j^{\text{type}}, v_j^{\text{explain}}, v_j^{\text{score}})$ consists of four parts: entity name $v_j^{\text{name}} \subseteq e_i^{\text{text}}$, entity type v_j^{type} , entity explanation v_j^{explain} , and confidence score $v_j^{\text{score}} \in (0, 100]$ indicating the certainty of extraction from e_i^{text} .

LLM-based Extraction. Following this knowledge representation, we design an n-ary relation extraction prompt p_{ext} to enable the LLM π to perform end-to-end knowledge fragment segmentation and entity recognition, thereby forming the n-ary relational fact set F_n^d :

$$F_n^d = \{f_1, f_2, \dots, f_k\} \sim \pi(F_n|p_{\text{ext}}, d), \quad (4)$$

where each extracted n-ary relational fact $f_i = (e_i, V_{e_i})$ contains information about the corresponding hyperedge e_i and its associated entity set V_{e_i} .

We convert all texts in K into hyperedges and entities using n-ary relation extraction, forming a complete knowledge hypergraph G_H :

$$G_H = \bigcup_{d \in K} \text{ExtractNaryFacts}(d), \quad (5)$$

where $\text{ExtractNaryFacts}(d)$ represents the process of extracting n-ary relational facts with hyperedges and entities from a document fragment d .

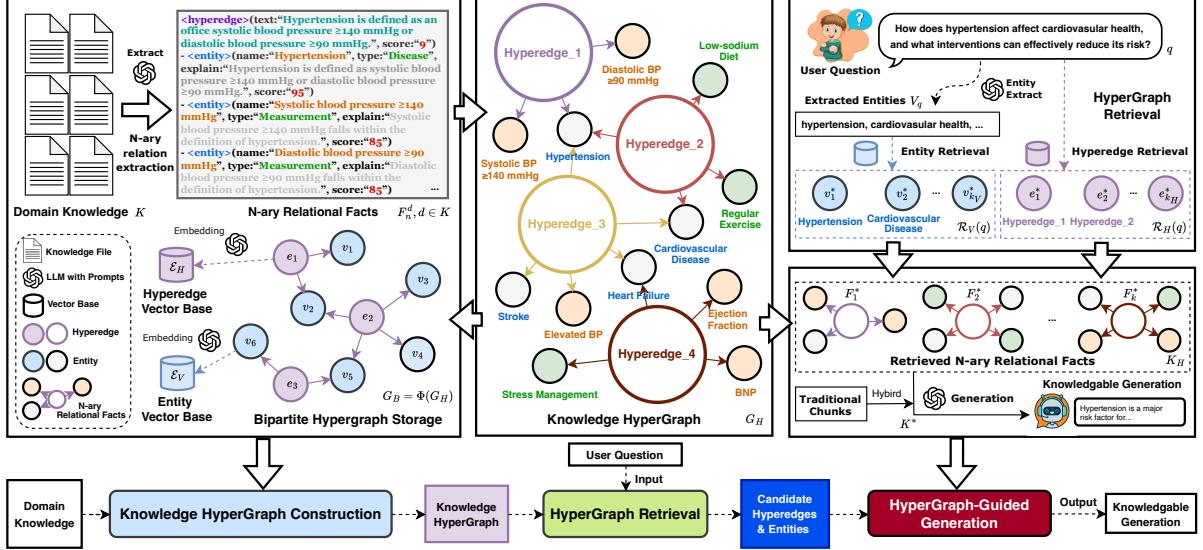


Figure 3: An overview of HyperGraphRAG, which constructs a knowledge hypergraph from domain knowledge, retrieves relevant hyperedges and entities based on user queries, and generates knowledgeable responses.

4.1.2 Bipartite Hypergraph Storage

After n-ary relation extraction, we need to store the constructed knowledge hypergraph G_H in a graph database to support an efficient query. We adopt an ordinary graph database in a bipartite graph structure G_B :

$$G_B = (V_B, E_B) = \Phi(G_H), \quad (6)$$

to store the knowledge hypergraph $G_H = (V, E_H)$, where Φ is a transformation function satisfying:

$$V_B = V \cup E_H, \quad (7)$$

$$E_B = \{(e_H, v) \mid e_H \in E_H, v \in V_{e_H}\}, \quad (8)$$

where V_B is the set of nodes in G_B , formed by merging the entity set V and the hyperedge set E_H from G_H . The edge set E_B consists of connections between each hyperedge $e_H \in E_H$ and its associated entities $v \in V_{e_H}$.

Based on G_B , we can efficiently query all entities associated with a hyperedge e_H or query all hyperedges linked to a specific entity v , benefiting from the optimized query efficiency of ordinary graph database, as well as reserving the complete hypergraph-structured knowledge representation.

Moreover, G_B allows incremental updates for dynamicaly expansion:

$$G_B \leftarrow G_B \cup \Phi(G'_H), \quad (9)$$

where G'_H represents newly added hypergraph information. The transformation of hyperedges and entities into the bipartite graph storage format enables seamless updates to the graph database.

4.1.3 Vector Representation Storage

To support efficient semantic retrieval, we embed hyperedges $e_H \in E_H$ and entities $v \in V$ using the same embedding model f , ensuring that the vector representation of hyperedges and entities are in the same vector space as questions. Let Ψ be the vector function, then the vector representation storage for the knowledge hypergraph G_H is defined as:

$$\Psi(G_H) = (\mathcal{E}_H, \mathcal{E}_V), \quad (10)$$

where \mathcal{E}_H is the vector base of hyperedges and \mathcal{E}_V is the vector base of entities:

$$\mathcal{E}_H = \{\mathbf{h}_{e_H} \mid e_H \in E_H\}, \quad (11)$$

$$\mathcal{E}_V = \{\mathbf{h}_v \mid v \in V\}, \quad (12)$$

where each hyperedge e_H and entity v in G_H is embedded into their vector representations: $\mathbf{h}_{e_H} = f(e_H)$, and $\mathbf{h}_v = f(v)$, respectively.

4.2 Hypergraph Retrieval Strategy

After constructing and storing the hypergraph G_H , we need to design an efficient retrieval strategy to match user questions with relevant hyperedges and entities. Hypergraph retrieval consists of entity extraction, entity retrieval, and hyperedge retrieval.

4.2.1 Entity Extraction

Before entity retrieval, we extract key entities from the user question q to facilitate subsequent matching. We design an entity extraction prompt p_{q_ext} along with the LLM π to perform entity extraction:

$$V_q \sim \pi(V|p_{q_ext}, q), \quad (13)$$

where V_q represents the entity set extracted from q .

4.2.2 Entity Retrieval

After extracting entities, we need to retrieve the most relevant entities from the entity set V of the knowledge hypergraph G_H . We define the entity retrieval function \mathcal{R}_V , which retrieves the most relevant entities from \mathcal{E}_V based on cosine similarity:

$$\mathcal{R}_V(q) = \operatorname{argmax}_{v \in V}^k (\operatorname{sim}(\mathbf{h}_{V_q}, \mathbf{h}_v) \odot v^{\text{score}})_{>\tau_V}, \quad (14)$$

where $\mathbf{h}_{V_q} = f(V_q)$ is the concatenated text vector representation of the extracted entity set V_q , $\mathbf{h}_v \in \mathcal{E}_V$ is the vector representation of entity v , $\operatorname{sim}(\cdot, \cdot)$ denotes the similarity function, \odot represents element-wise multiplication between similarity and entity relevance score v^{score} determining the final ranking score, τ_V is the threshold for the entity retrieval score, and k_V is the limit on the number of retrieved entities.

4.2.3 Hyperedge Retrieval

Moreover, to expand the retrieval scope by searching for complete n-ary relations within the hyperedge set E_H of the knowledge hypergraph G_H , we define the hyperedge retrieval function \mathcal{R}_H , which retrieves a set of hyperedges related to q :

$$\mathcal{R}_H(q) = \operatorname{argmax}_{e_H \in E_B}^k (\operatorname{sim}(\mathbf{h}_q, \mathbf{h}_{e_H}) \odot e_H^{\text{score}})_{>\tau_H}, \quad (15)$$

where $\mathbf{h}_q = f(q)$ is the text vector representation of q , $\mathbf{h}_{e_H} \in \mathcal{E}_H$ is the vector representation of the hyperedge e_H , \odot represents element-wise multiplication between similarity and hyperedge relevance score e_H^{score} determining the final ranking score, τ_H is the threshold for the hyperedge retrieval score, and k_H limits the number of retrieved hyperedges.

4.3 Hypergraph-Guided Generation

To fully utilize the structured knowledge in the hypergraph, we propose the Hypergraph-Guided Generation mechanism, which consists of hypergraph knowledge fusion and generation augmentation.

4.3.1 Hypergraph Knowledge Fusion

The core objective of hypergraph knowledge fusion is to expand and reorganize the retrieved n-ary relational knowledge to form a complete knowledge input. Since q may only match partial entities or hyperedges, we need to further expand the retrieval scope. To obtain the complete set of

n-ary relational facts, we design a bidirectional expansion strategy, including expanding hyperedges from retrieved entities and expanding entities from retrieved hyperedges.

First, given the entity set retrieved from q , denoted as $\mathcal{R}_V(q) = \{v_1, v_2, \dots, v_{k_V}\}$, we retrieve all hyperedges in the knowledge hypergraph G_H that connect these entities:

$$\mathcal{F}_V^* = \bigcup_{v_i \in \mathcal{R}_V(q)} \{(e_H, V_{e_H}) \mid v_i \in V_{e_H}, e_H \in E_H\}. \quad (16)$$

Next, we expand the set of entities connected to the retrieved hyperedges $\mathcal{R}_H(q) = \{e_1, e_2, \dots, e_{k_H}\}$:

$$\mathcal{F}_H^* = \bigcup_{e_i \in \mathcal{R}_H(q)} \{(e_i, V_{e_i}) \mid V_{e_i} \subseteq V\} \quad (17)$$

Finally, we merge the expanded hyperedge set \mathcal{F}_V^* and the expanded entity set \mathcal{F}_H^* to form a complete retrieved n-ary relational fact set K_H :

$$K_H = \mathcal{F}_V^* \cup \mathcal{F}_H^*. \quad (18)$$

This set contains all n-ary relational knowledge necessary for reasoning and generation, ensuring that the input to the LLM is as complete as possible.

4.3.2 Generation Augmentation

After completion of hypergraph knowledge fusion, we augment the generation strategy to improve the accuracy and readability of the responses. We also adopt a hybrid RAG fusion mechanism, combining hypergraph knowledge K_H with retrieved text fragments based on chunks K_{chunk} to form the final input of knowledge. We define the final knowledge input K^* as:

$$K^* = K_H \cup K_{\text{chunk}}, \quad (19)$$

where K_{chunk} consists of chunk-based document fragments retrieved using traditional RAG:

$$K_{\text{chunk}} = \operatorname{argmax}_{d \in K}^{k_C} \operatorname{sim}(\mathbf{h}_q, \mathbf{h}_d)_{>\tau_C}, \quad (20)$$

where \mathbf{h}_q represents the vector representation of q , \mathbf{h}_d represents the vector representation of chunk-based document fragment d , τ_C is the similarity threshold for chunk retrieval, and k_C limits the number of retrieved chunks.

Finally, we use a retrieval-augmented generation prompt p_{gen} that combines hypergraph knowledge K^* and the user question q as input to the LLM π to generate the final answer:

$$y^* \sim \pi(y|p_{\text{gen}}, K^*, q), \quad (21)$$

where y^* is the generated response.

Table 1: Performance comparison across different domains. **Bold** numbers indicate the best performance.

Method	Medicine			Agriculture			CS			Legal			Mix		
	C-Rec	C-ERec	A-Rel												
<i>Random Source</i>															
NaiveGeneration	10.43	10.28	31.13	9.59	8.80	28.35	15.78	14.52	47.64	14.41	13.49	41.82	14.78	13.87	47.63
StandardRAG	14.16	14.17	42.69	11.63	10.88	38.80	19.33	17.53	55.98	17.37	16.43	51.35	17.25	16.15	57.28
GraphRAG	41.61	44.64	58.98	34.44	35.99	56.50	41.72	42.54	56.38	35.67	36.57	46.73	27.63	27.99	33.58
LightRAG	47.86	50.73	84.80	39.35	40.56	82.43	52.20	53.38	82.30	46.02	47.92	82.40	45.82	46.24	81.96
HyperGraphRAG (ours)	55.73	58.82	85.70	46.64	48.13	82.64	58.28	59.21	84.36	54.33	56.40	83.67	56.22	56.50	82.62
<i>Single-Entity Source</i>															
NaiveGeneration	15.57	14.18	42.46	16.28	14.08	44.18	19.75	17.09	46.34	19.92	17.65	48.67	19.22	16.45	50.65
StandardRAG	20.73	18.97	57.28	19.18	16.77	54.51	24.66	21.42	52.55	24.35	21.84	62.11	22.37	19.39	59.66
GraphRAG	42.91	45.68	51.17	43.18	43.17	48.56	42.54	42.20	45.79	38.31	37.44	37.77	30.60	30.67	33.53
LightRAG	56.10	56.24	83.22	54.52	53.65	79.49	60.65	59.71	77.82	56.08	55.15	81.80	53.28	51.99	81.22
HyperGraphRAG (ours)	65.16	65.26	82.04	60.71	60.42	79.76	66.57	65.61	79.42	64.85	64.18	81.99	65.22	63.85	83.03
<i>Multi-Entity Source</i>															
NaiveGeneration	12.85	12.51	34.76	14.56	13.38	40.45	16.59	14.68	40.83	16.81	15.34	42.60	7.70	6.51	12.76
StandardRAG	18.62	18.20	51.53	18.70	17.53	52.04	23.62	21.19	58.80	23.33	21.66	58.35	22.30	19.96	56.14
GraphRAG	40.02	43.33	52.60	42.67	43.11	56.56	42.93	42.67	49.37	38.17	37.81	42.88	32.32	32.33	36.78
LightRAG	53.96	55.57	87.19	52.53	52.13	86.74	60.31	59.68	86.22	55.67	55.24	86.42	56.24	54.53	85.61
HyperGraphRAG (ours)	60.12	61.79	87.71	58.99	59.16	88.03	66.47	65.95	87.48	61.57	61.48	87.48	63.13	61.62	85.72
<i>Overall</i>															
NaiveGeneration	12.95	12.33	36.12	13.48	12.09	37.66	17.37	15.43	44.94	17.05	15.49	44.36	13.90	12.28	37.02
StandardRAG	17.84	17.11	50.50	16.50	15.06	48.45	22.55	20.05	55.78	21.68	19.98	57.27	20.64	18.50	57.69
GraphRAG	41.51	44.55	54.25	40.10	40.75	53.87	42.40	42.50	50.51	37.39	37.27	42.46	30.18	30.33	34.63
LightRAG	52.64	54.18	85.07	48.73	48.78	82.89	57.72	57.59	82.11	52.59	52.17	83.54	51.78	50.92	82.93
HyperGraphRAG (ours)	60.34	61.95	85.15	55.44	55.90	83.78	63.78	63.59	83.75	60.25	60.69	84.38	61.52	60.69	83.79

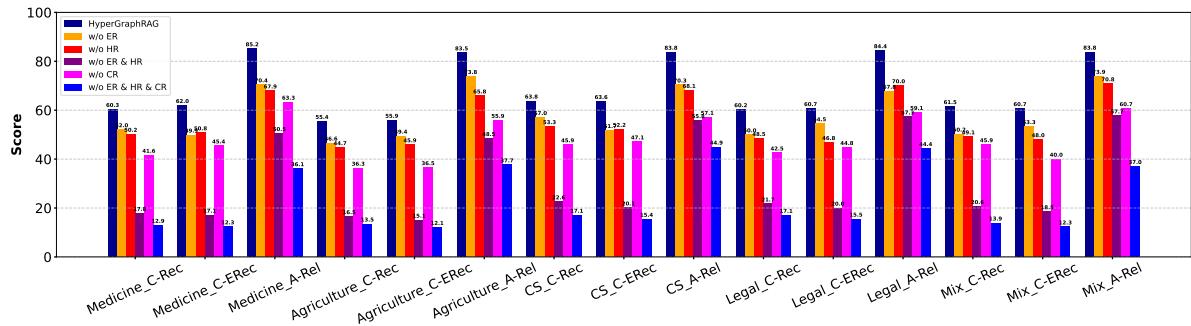


Figure 4: Comparison of knowledge representation of different RAG methods: standard RAG uses chunk as unit, GraphRAG captures binary relations with graphs, and HyperGraphRAG models n-ary relations with hyperedges.

5 Experiments

This section presents the experimental setup, main results, and analysis. We answer the following research questions (RQs): **RQ1:** Does HyperGraphRAG outperform other methods? **RQ2:** Does the main component of HyperGraphRAG work? **RQ3:** How is the quality of the knowledge hypergraph constructed by HyperGraphRAG in different domains? **RQ4:** What are the efficiency and cost of HyperGraphRAG?

5.1 Experimental Setup

Datasets. To evaluate the performance of HyperGraphRAG across multiple domains, we select four knowledge contexts from UltraDomain (Qian et al., 2024), as used in LightRAG (Guo et al., 2024): **Agriculture**, Computer Science (**CS**), **Legal**, and a mixed domain (**Mix**). In addition,

we include the latest international hypertension guidelines (McEvoy et al., 2024) as the foundational knowledge for the **Medicine** domain. For each of these five domains, we apply different sampling methods, **Random**, **Single-Entity**, and **Multi-Entity**, to generate questions, followed by human verification of ground-truth answers. More details can be found in Appendix A.

Baselines. We compared HyperGraphRAG against four publicly available baseline methods: **NaiveGeneration** (OpenAI et al., 2024), which directly generates responses using LLM; **StandardRAG** (Gao et al., 2024), a traditional chunk-based RAG approach; **GraphRAG** (Edge et al., 2024), a graph-based RAG model introduced by Microsoft; and **LightRAG** (Guo et al., 2024), a lightweight optimized RAG model. To ensure fairness, we use the same generation prompt as LightRAG.

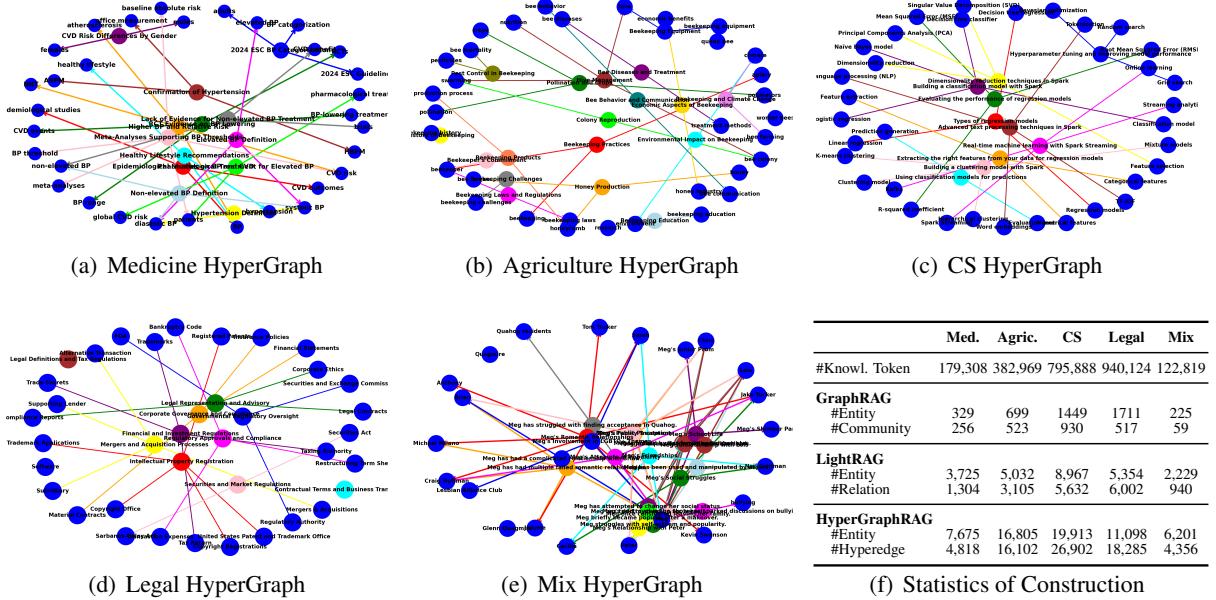


Figure 5: Hypergraph-structured knowledge representation across five domains: Visualizations of knowledge hypergraphs constructed in (a) Medicine, (b) Agriculture, (c) CS, (d) Legal, and (e) Mixed domains. (f) Statistical comparison highlights HyperGraphRAG’s richer expressiveness over GraphRAG and LightRAG.

Evaluation Metrics. Following evaluation methodologies from RAGAS (Es et al., 2024) and OG-RAG (Sharma et al., 2024), we employ the following metrics: Context Recall (C-Rec), Context Entity Recall (C-ERec), and Answer Relevance (A-Rel). Details of these metrics are provided in Appendix B.

Implementation Details. We use OpenAI’s GPT-4o-mini for extraction and generation, and text-embedding-3-small for vector. During retrieval, we set the following parameters: entity retrieval $k_V = 60$, $\tau_V = 50$; hyperedge retrieval $k_H = 60$, $\tau_H = 5$; and chunk retrieval $k_C = 3$, $\tau_C = 0.5$. All experiments were conducted on a server with an 80-core CPU and 512GB RAM.

5.2 Main Results (RQ1)

To evaluate the effectiveness of HyperGraphRAG, we compare its performance across multiple domains, as shown in Table 1.

Overall Performance. HyperGraphRAG consistently outperforms all baselines, achieving the highest C-Rec, C-ERec, and A-Rel scores. It averages C-Rec = 60.34, C-ERec = 61.95, and A-Rel = 85.15, surpassing GraphRAG and LightRAG. Compared to StandardRAG, HyperGraphRAG improves answer relevance by 28%, demonstrating the advantage of hypergraph structures in knowledge modeling. Its performance is particularly

strong in knowledge-intensive domains, achieving C-Rec = 60.34 and A-Rel = 85.15 in Medicine, and C-Rec = 63.78 and A-Rel = 83.75 in CS.

Comparison Across Different Sources. We analyze HyperGraphRAG’s performance across different sampling sources. Random Source constructs questions randomly, making retrieval the most challenging due to scattered knowledge. Single-Entity Source, based on a single entity, is the easiest due to its clear retrieval target. Multi-Entity Source, involving multiple entities, presents moderate difficulty. HyperGraphRAG excels in all three settings, maintaining high recall and answer relevance, proving its adaptability to complex knowledge scenarios.

5.3 Ablation Study (RQ2)

To assess the impact of each module in HyperGraphRAG, we conduct an ablation study by removing entity retrieval (ER), hyperedge retrieval (HR), and their combination (ER & HR). We also remove chunk retrieval fusion (CR) and evaluate its impact independently and alongside ER & HR. Figure 4 presents the results.

Impact of Entity and Hyperedge Retrieval. Removing ER (w/o ER) slightly reduces C-Rec and A-Rel, particularly in Computer Science and Medicine, indicating its role in precise knowledge localization. Removing HR (w/o HR) results in

minor performance drops, especially in complex relational reasoning tasks like Legal, highlighting its contribution to multi-entity relationship extraction. Removing both (w/o ER & HR) causes the most significant drop, aligning performance with StandardRAG, demonstrating their joint importance for effective knowledge retrieval.

Impact of Chunk Retrieval Fusion. Removing CR (w/o CR) lowers A-Rel, showing its role in enhancing generation via supplementary knowledge. When all three components are removed (w/o ER & HR & CR), performance deteriorates to NaiveGeneration levels. These results confirm the necessity of all retrieval components in HyperGraphRAG for optimal generation quality.

5.4 Analysis of HyperGraph-structured Knowledge Representation (RQ3)

To assess HyperGraphRAG’s knowledge representation, we visualize its hypergraphs in Medicine, Agriculture, CS, Legal, and Mixed domains, comparing them with GraphRAG and LightRAG. Figure 5 presents the results.

Visualization of Knowledge Structures. As shown in Figure 5(a)-5(e), HyperGraphRAG constructs more comprehensive knowledge structures. Unlike GraphRAG, which only models binary relations, HyperGraphRAG connects multiple entities via hyperedges, forming a more interconnected and expressive network.

Statistical Analysis. Figure 5(f) compares entity and relation coverage. HyperGraphRAG significantly surpasses GraphRAG and LightRAG in all domains. For instance, in CS, it constructs 19,913 entities and 26,902 hyperedges, whereas GraphRAG has 930 communities and LightRAG 5,632 relations. This highlights HyperGraphRAG’s superior ability to capture complex relationships, enhancing retrieval and generation quality.

5.5 Analysis of Efficiency and Cost (RQ4)

To evaluate the efficiency and cost of HyperGraphRAG, we compare different methods in terms of knowledge construction, retrieval, and generation. We assess time consumption per 10k tokens (TP10kT), cost per 10k tokens (CP10kT), time per query (TPQ), and cost per query (CPQ), with the experimental results presented in Table 2.

Efficiency & Cost of Construction Phase. In terms of TP10kT, the knowledge construction time for HyperGraphRAG is 366.52s, slightly higher than GraphRAG (91.25s) and LightRAG (297.83s).

Method	Construction		Generation	
	TP10kT	CP10kT	TPQ	CPQ
NaiveGeneration	0 s	0 \$	5.607 s	0.00174 \$
StandardRAG	0 s	0 \$	6.960 s	0.00207 \$
GraphRAG	91.25 s	0.2408 \$	13.318 s	0.00493 \$
LightRAG	297.83 s	0.0309 \$	8.336 s	0.00296 \$
HyperGraphRAG	366.52 s	0.0452 \$	9.546 s	0.00321 \$

Table 2: Efficiency comparison and cost comparison of different methods in terms of knowledge construction and retrieval and generation.

Additionally, in CP10kT (cost per 10k tokens), HyperGraphRAG costs \$0.0452, slightly higher than LightRAG (\$0.0309), but its hypergraph structure captures richer n-ary relations, leading to more stable and efficient retrieval and generation. Moreover, its cost remains significantly lower than GraphRAG (\$0.2408).

Efficiency & Cost of Generation Phase. For TPQ (time per query), HyperGraphRAG’s retrieval and generation time is 9.546s, slightly higher than LightRAG (8.336s), but lower than GraphRAG (13.318s). Additionally, in CPQ (cost per query), HyperGraphRAG costs \$0.00321, lower than GraphRAG (\$0.00493) and close to LightRAG (\$0.00296). These results indicate that although HyperGraphRAG slightly increases query time and cost due to its hypergraph structure, it keeps them within a reasonable range, making it more advantageous for large-scale knowledge-intensive applications.

6 Conclusion

This paper presents HyperGraphRAG, a novel hypergraph-based RAG method that leverages hypergraph-structured knowledge representation to model n-ary relational facts. Unlike StandardRAG, which relies on chunk-based retrieval, or GraphRAG, which only supports binary relations, HyperGraphRAG introduces a hypergraph retrieval strategy that effectively captures complex n-ary relations. Extensive experiments across multiple domains demonstrate that HyperGraphRAG consistently outperforms existing RAG frameworks in retrieval recall and answer relevance. Additionally, efficiency and cost analysis show that while HyperGraphRAG incurs slightly higher knowledge construction costs than LightRAG, it maintains a reasonable range. Overall, HyperGraphRAG enhances RAG through hypergraph modeling, providing a scalable and efficient solution for real-world knowledge-driven applications.

Limitations

Although HyperGraphRAG has achieved significant improvements in knowledge retrieval and generation, there is still room for enhancement.

- First, the current approach is still limited to the text modality and does not fully utilize multimodal knowledge such as images and tables, which may affect retrieval coverage in certain domains like medicine and law.
- Second, hypergraph retrieval primarily relies on single-step matching, lacking strong multi-hop reasoning capabilities, making it difficult to perform complex inference over deep knowledge chains.
- Additionally, while HyperGraphRAG optimizes knowledge structure through hypergraph representation, there is still room for improving retrieval and generation efficiency.

We will explore these aspects in future work.

Ethics Statement

This paper investigates the problem of RAG with Hypergraph-Structured Knowledge Representation. We use large language models and hypergraph-based retrieval methods to enhance knowledge representation, retrieval accuracy, and generation quality. All data used in this study are publicly available from the internet, and no personally identifiable or sensitive information is involved. Therefore, we believe it does not violate any ethical principles.

References

- Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. [From local to global: A graph rag approach to query-focused summarization](#). *Preprint*, arXiv:2404.16130.
- Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. [RAGAs: Automated evaluation of retrieval augmented generation](#). In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.
- Tianyu Fan, Jingyuan Wang, Xubin Ren, and Chao Huang. 2025. [Minirag: Towards extremely simple retrieval-augmented generation](#). *Preprint*, arXiv:2501.06713.
- Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. [Message passing for hyper-relational knowledge graphs](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7346–7359, Online. Association for Computational Linguistics.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. [Retrieval-augmented generation for large language models: A survey](#). *Preprint*, arXiv:2312.10997.
- Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. [Lightrag: Simple and fast retrieval-augmented generation](#). *Preprint*, arXiv:2410.05779.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kütter, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive nlp tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.
- Yu Liu, Quanming Yao, and Yong Li. 2020. [Generalizing tensor decomposition for n-ary relational knowledge bases](#). In *Proceedings of The Web Conference 2020*, WWW ’20, page 1104–1114, New York, NY, USA. Association for Computing Machinery.
- Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. [Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.
- Haoran Luo, Haihong E, Yuhao Yang, Yikai Guo, Mingzhi Sun, Tianyu Yao, Zichen Tang, Kaiyang Wan, Meina Song, and Wei Lin. 2023. [HAHE: Hierarchical attention for hyper-relational knowledge graphs in global and local level](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8095–8107, Toronto, Canada. Association for Computational Linguistics.
- Haoran Luo, Haihong E, Yuhao Yang, Tianyu Yao, Yikai Guo, Zichen Tang, Wentai Zhang, Kaiyang Wan, Shiyao Peng, Meina Song, Wei Lin, Yifan Zhu, and Luu Anh Tuan. 2024. [Text2nkg: Fine-grained n-ary relation extraction for n-ary relational knowledge graph construction](#). *Preprint*, arXiv:2310.05185.
- John William McEvoy, Cian P McCarthy, Rosa Maria Bruno, Sofie Brouwers, Michelle D Canavan, Claudio Ceconi, Ruxandra Maria Christodorescu, Stella S Daskalopoulou, Charles J Ferro, Eva Gerdts, et al.

2024. 2024 esc guidelines for the management of elevated blood pressure and hypertension. *Giornale italiano di cardiologia* (2006), 25(11):1e–107e.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.
- Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.
- Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. 2024. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery. *Preprint*, arXiv:2409.05591.
- Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2020. Beyond triplets: Hyper-relational knowledge graph embedding for link prediction. In *Proceedings of The Web Conference 2020*, WWW ’20, page 1885–1896, New York, NY, USA. Association for Computing Machinery.
- Kartik Sharma, Peeyush Kumar, and Yunqing Li. 2024. Og-rag: Ontology-grounded retrieval-augmented generation for large language models. *Preprint*, arXiv:2412.15235.
- Quan Wang, Haifeng Wang, Yajuan Lyu, and Yong Zhu. 2021. Link prediction on n-ary relational facts: A graph-based approach. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 396–407, Online. Association for Computational Linguistics.
- Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI’16*, page 1300–1307. AAAI Press.
- Junde Wu, Jiayuan Zhu, Yunli Qi, Jingkun Chen, Min Xu, Filippo Menolascina, and Vicente Grau. 2024. Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation. *Preprint*, arXiv:2408.04187.
- Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *Proceedings of the 2018 World Wide Web Conference*, WWW ’18, page 1185–1194, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2024. A survey of large language models. *Preprint*, arXiv:2303.18223.
- Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Appendix

A Dataset Construction

This section describes the dataset construction process for HyperGraphRAG, including the five knowledge-intensive domains and three sampling strategies.

A.1 Domains

The dataset used for HyperGraphRAG evaluation covers five domains, with data sourced as follows:

Medicine: Derived from the latest international hypertension guidelines (McEvoy et al., 2024), covering medical diagnosis, treatment plans, and clinical indicators. **Agriculture:** Extracted from the UltraDomain dataset (Qian et al., 2024), including knowledge on agricultural production, crop management, and pest control. **Computer Science (CS):** Sourced from the UltraDomain dataset, encompassing computer architecture, algorithms, and machine learning. **Legal:** Based on the UltraDomain dataset, covering legal provisions, judicial precedents, and regulatory interpretations. **Mix:** A combination of multiple domains to assess the model’s generalization ability across interdisciplinary tasks.

A.2 Sampling Strategies

To generate queries with varying complexity, we employ three sampling strategies:

- **Random Source:** This method randomly selects knowledge fragments to generate questions, making it the most challenging setting as retrieved knowledge tends to be more dispersed and difficult to match.
- **Single-Entity Source:** This method generates questions based on a single entity, providing clear retrieval targets and strong knowledge relevance, making it the easiest setting.
- **Multi-Entity Source:** This method generates questions based on multiple related entities, representing a medium difficulty level as it

requires the model to integrate relational information across entities.

All datasets undergo manual review to ensure the accuracy of annotated answers and the fairness of model evaluation.

B Evaluation Details

This section introduces the three evaluation metrics used in our experiments: Context Recall (C-Rec), Context Entity Recall (C-ERec), and Answer Relevance (A-Rel), along with their mathematical definitions.

B.1 Context Recall (C-Rec)

Context Recall measures how much of the ground truth knowledge is present in the generated response. Given a ground truth response G and a predicted response P , we define C-Rec as:

$$\text{C-Rec} = \frac{|\text{Tokenize}(G) \cap \text{Tokenize}(P)|}{|\text{Tokenize}(G)|} \quad (22)$$

where $\text{Tokenize}(\cdot)$ represents the tokenization process using the GPT-2 tokenizer. A higher C-Rec indicates that more relevant information from the ground truth is preserved in the generated response.

B.2 Context Entity Recall (C-ERec)

Context Entity Recall evaluates how well key entities from the ground truth are retained in the generated response. Let $E(G)$ be the set of entities extracted from the ground truth and $E(P)$ be the set extracted from the prediction. C-ERec is defined as:

$$\text{C-ERec} = \frac{|E(G) \cap E(P)|}{|E(G)|} \quad (23)$$

where entity extraction is performed using a rule-based method that considers capitalized words as potential entities. A higher C-ERec score indicates better entity preservation.

B.3 Answer Relevance (A-Rel)

Answer Relevance measures the overlap between the question and the generated response, evaluating whether the generated response directly addresses the given query. Given a query Q and a prediction P , A-Rel is computed as:

$$\text{A-Rel} = \frac{|\text{Tokenize}(Q) \cap \text{Tokenize}(P)|}{|\text{Tokenize}(Q)|} \quad (24)$$

where $\text{Tokenize}(\cdot)$ applies GPT-2 tokenization. A higher A-Rel score indicates that the response is more relevant to the given query.

C Prompts Used in HyperGraphRAG

C.1 N-ary relation Extraction Prompt

```

-Goal-
Given a text document that is potentially relevant to this activity and a list of entity types, identify all entities of those types from the text and all relationships among the identified entities. Use {language} as output language.

-Steps-
1. Divide the text into several complete knowledge segments. For each knowledge segment, extract the following information:
- knowledge_segment: A sentence that describes the context of the knowledge segment.
- completeness_score: A score from 0 to 10 indicating the completeness of the knowledge segment.
Format each knowledge segment as ("hyper-relation"<tuple_delimiter>knowledge_segment<tuple_delimiter>completeness_score)

2. Identify all entities in each knowledge segment. For each identified entity, extract the following information:
- entity_name: Name of the entity, use same language as input text. If English, capitalize the name.
- entity_type: Type of the entity.
- entity_description: Comprehensive description of the entity's attributes and activities.
- key_score: A score from 0 to 100 indicating the importance of the entity in the text.
Format each entity as
("entity"<tuple_delimiter>entity_name<tuple_delimiter>entity_type<tuple_delimiter>entity_des
cription<tuple_delimiter>key_score)

3. Return output in {language} as a single list of all the entities and relationships identified in steps 1 and 2. Use **{record_delimiter}** as the list delimiter.

4. When finished, output {completion_delimiter}

#####
-Examples-
#####
{examples}
#####
#####
```

C.2 Entity Extraction Prompt

```

--Role--
You are a helpful assistant tasked with identifying entities in the user's query.

--Goal--
Given the query, list all entities.

--Instructions--
- Output the keywords in JSON format.

#####
-Examples-
#####
{examples}
#####
#####
```

C.3 Retrieval-Augmented Generation Prompt

```

--Role--
You are a helpful assistant responding to questions about data in the tables provided.

--Goal--
Generate a response of the target length and format that responds to the user's question, summarizing all information in the input data tables appropriate for the response length and format, and incorporating any relevant general knowledge.
If you don't know the answer, just say so. Do not make anything up.
Do not include information where the supporting evidence for it is not provided.

--Target response length and format--
{response_type}

--Data tables--

{context_data}
Add sections and commentary to the response as appropriate for the length and format. Style the response in markdown.
```