NGM-RAG: Neural Graph Matching based Retrieval-Augmented Generation

Guo Chen[1,†], Ziwen Li[1,†], Maolin Zheng[1], Hao Gao[2], Junjie Huang[1,*], Tao Jia[1]

[1]College of Computer and Information Science, Southwest University, China
[2]Beijing Institute of Control Engineering, China
{cg1281838223, pique0202, zml778922461}@email.swu.edu.cn
goleey@163.com, junjiehuang@swu.edu.cn, tjia@swu.edu.cn
[†]Equal contribution. [*]Corresponding author.

Abstract

Retrieval-Augmented Generation (RAG) significantly enhances the ability of Large Language Models (LLMs) to provide accurate and contextually relevant answers by dynamically integrating external databases. However, traditional RAG methods are primarily constrained by their reliance on text-based retrieval strategies, which often struggle with complex questions requiring multi-hop reasoning. To address this limitation, we introduce Neural Graph Matching based Retrieval-Augmented Generation (NGM-RAG), a novel framework that leverages graph structures to effectively capture and utilize relational knowledge for improved retrieval and answer generation. NGM-RAG explicitly incorporates graph construction, graph matching, and answer generation into a unified process. Within this framework, we propose a neural graph matching approach that combines text-based matching with Graph Neural Networks (GNNs). By employing an adaptive weighting strategy, NGM-RAG efficiently integrates multiple matching methods to select the most relevant contextual node information for answer generation. Experimental results on multi-hop question answering and long-context summarization tasks demonstrate that our NGM-RAG model achieves superior performance compared to both traditional NaiveRAG methods and state-of-the-art graph-enhanced approaches such as GraphRAG and LightRAG.

1 Introduction

Retrieval-Augmented Generation (RAG) is an innovative approach that enables Large Language Models (LLMs) to generate more accurate and contextual responses, significantly improving their practicality in real-world applications (Es et al., 2024; Salemi and Zamani, 2024). Compared to traditional large model Supervised Fine-Tuning (SFT) and In Context Learning (ICL) (Taori et al., 2023; Zhang et al., 2023) methods, RAG can adapt to specific domain knowledge to ensure that the information provided is not only relevant but also tailored to the user's needs. In addition, RAG can dynamically update private data, so it has received widespread attention from academia and industry (Gao et al., 2023).

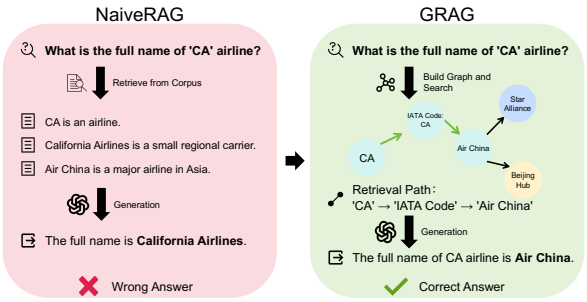

Figure 1: An example question that NaiveRAG may answer incorrectly due to inaccurate retrieval, whereas GRAG can provide the correct answer through knowledge graph relationships.

Although RAG effectively leverages external and up-to-date databases, its performance heavily depends on the accuracy of the retrieved context (Zhao et al., 2024). However, in real-world scenarios, many questions require multi-hop reasoning to answer correctly (Yang et al., 2024). Traditional text retrieval methods, such as sparse retrieval (Robertson et al., 2009) or dense retrieval (Karpukhin et al., 2020), cannot effectively support LLMs in answering complex questions.

For example, as shown in Figure 1, given a question like `What is the full name of the 'CA' airline?`, traditional RAG retrieves contexts related to 'CA', but these contexts lack semantic connections. As a result, the LLM generates an incorrect answer. To better organize and manage knowledge, graph structures (with their intrinsic nature of "nodes connected by edges", encoding rich, heterogeneous, and relational information) are integrated into the RAG framework (Han et al., 2024). By modeling knowledge graphs and performing

graph search, “CA" can be associated with “Air China" via its “IATA code", enabling the retrieval of the correct answer.

Although graph structures have already been explored in RAG, such as GraphRAG (Edge et al., 2024) and LightRAG (Guo et al., 2024), effectively leveraging the relational information of knowledge graphs—namely, graph structures—still lacks a clear solution. In this paper, we first provide a formal definition of Graph Retrieval-Augmented Generation (GRAG), and then propose a novel Neural Graph Matching RAG (NGM-RAG) framework comprising graph construction, graph matching, and answer generation. For the graph matching, in addition to identifying relevant nodes via text similarity, we further propose a Neural Graph Matching method based on Graph Neural Networks (GNNs). Our Neural Graph Matching method can be applied to parameter-free homogeneous graph neural networks, such as LightGCN (He et al., 2020). It can also be adapted to heterogeneous graph neural networks, such as GINE (Hu et al., 2019), by introducing supervision signals. By applying an adaptive weighting strategy, different matching algorithms can be effectively integrated. We then incorporate the most relevant information into the context to perform retrieval-augmented generation. Empirical results on multiple multi-hop question answering and long-context summarization tasks demonstrate that our proposed model outperforms both NaiveRAG and state-of-the-art GRAG methods, such as GraphRAG and LightRAG.

We summarize our contributions as follows:

- We present a formal definition of GRAG, and based on this formalization, propose a Neural Graph Matching-based RAG framework, named NGM-RAG.
- NGM-RAG consists of three components: graph construction, graph matching, and answer generation. For graph matching, we integrate direct matching, text similarity, and neural graph matching algorithms into an adaptive framework that combines different matching functions.
- NGM-RAG achieves superior performance compared to several baselines on multiple multi-hop question answering and long-context summarization tasks.

2 Related Work

2.1 Graph Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) enhances language models by integrating external knowledge during generation, typically using semantic or lexical similarity for retrieval. However, traditional RAG methods are limited in effectively utilizing structured, relational data (Han et al., 2024). Graph Retrieval Augmented Generation (GRAG) addresses this limitation by incorporating graph-structured data into the retrieval process. Graphs naturally encode rich relational information, making them ideal for enhancing retrieval (Han et al., 2024). GraphRAG utilizes specialized techniques such as Graph Neural Networks (GNNs) (Kipf and Welling, 2017) and graph traversal algorithms (Wang et al., 2024), allowing retrieval based on relational paths rather than just embeddings (Mavromatis and Karypis, 2024). Overall, GraphRAG effectively captures structural nuances, improving the quality and accuracy of language models, especially in complex reasoning and domain-specific tasks (Mavromatis and Karypis, 2024; Hu et al., 2024).

2.2 Neural Graph Matching

As an efficient neural-based method for modeling graph-structured data, Graph Neural Networks (GNNs) have been widely used in fields including knowledge graphs (Zhang et al., 2024), and recommendation systems (He et al., 2020).

GNNs were first employed to model homogeneous graphs, in which all nodes belong to the same entity type (Kipf and Welling, 2017; Velickovic et al., 2017; Hamilton et al., 2017). When extended to knowledge graphs (i.e., heterogeneous graphs), researchers have adopted modeling strategies typified by the Relational Graph Convolutional Network (R-GCN) (Schlichtkrull et al., 2017). Unlike a plain GCN (and its simplified variant SGCN (Wu et al., 2019; He et al., 2020), which can forego graph-level parameters and rely solely on node embeddings), heterogeneous settings must capture complex relation patterns and mappings across diverse node types; consequently, such models generally incorporate additional trainable parameters (Phan et al., 2022). Due to their powerful capability to model graph structures, Graph Neural Networks (GNNs) have also been widely applied to graph matching problems, enabling neural graph matching (Lou et al., 2020; Roy et al., 2022).

3 Problem Definition

Retrieval-Augmented Generation (RAG) system (Guo et al., 2024) can be defined as the following framework, denoted as $\mathcal{F}$:

$$\begin{aligned} \mathcal{F} &= (\mathcal{M}_{\text{Gen}}, \mathcal{M}_{\text{Ret}}), \\ \mathcal{F}(q; D) &= \mathcal{M}_{\text{Gen}}\big(q, \mathcal{M}_{\text{Ret}}(q, D)\big), \end{aligned} \tag{1}$$

where $\mathcal{M}_{\text{Gen}}$ is the Generation Modules, $\mathcal{M}_{\text{Ret}}$ is the Retriever Modules, q is the user input query, D is the external Database for query (a.k.a Corpus).

In general, the Generation Module $\mathcal{M}_{\text{Gen}}$ is an LLM that produces high-quality answers based on the user input query and the most relevant information obtained by Retriever Modules $\mathcal{M}_{\text{Ret}}$:

$$\mathcal{M}_{\text{Ret}}\big(q, D\big) = \mathcal{R}(q, \mathcal{I}(D)\big), \tag{2}$$

where Retrieval Module $\mathcal{M}_{\text{Ret}}$ usually consists of two parts: an Indexer $\mathcal{I}$ for external database and a Retriever $\mathcal{R}$ for matching query and indexed data.

For Retriever Module $\mathcal{M}_{\text{R}}$, a common approach is to embed text into vector representations and do semantic retrieval (Gao et al., 2023).

In our study, we focus on Graph Retrieval-Augmented Generation (GRAG), which introduces graph structure (e.g., knowledge graph) into Retrieval Module $\mathcal{M}_{\text{R}}^{G}$, including GRAG Retriever $\mathcal{R}^{G}$ and Indexer $\mathcal{I}^{G}$, as follows:

$$\begin{aligned} \mathcal{M}_{\text{Ret}}^{G} &= \mathcal{R}^{G}(G_{\text{query}}, G_{\text{target}}), \\ G_{\text{query}} &= \mathcal{I}^{G}(q), G_{\text{target}} = \mathcal{I}^{G}(D), \end{aligned} \tag{3}$$

where G_{target} and G_{query} are the target graph and query graph produced by Graph Indexer $\mathcal{I}^{G}$ respectivelly; $\mathcal{R}^{G}$ is a graph matching function to match the query graph and target graph. More specifically, given a query, $\mathcal{I}^{G}$ transforms text into a knowledge graph $G = (V, E)$, where V is the node set and E is the edge set. $\mathcal{R}^{G}$ performs matching between two graphs, to find the most relevent nodes V_{rel} and edges E_{rel} for Generation Modules $\mathcal{M}_{\text{Gen}}$, as follows:

$$\mathcal{M}_{\text{Ret}}^{G}(q, D) = (V_{\text{rel}}, E_{\text{rel}}). \tag{4}$$

The goal of GRAG is to design an efficient Graph Retriever Module $\mathcal{M}_{\text{Ret}}^{G}$ that has a stronger ability to solve complex and comprehensive problems than traditional RAG. Common graph-based RAG (e.g., GraphRAG (Edge et al., 2024) and LightRAG (Guo et al., 2024)) adopt the above paradigm with different designs on Graph Retriever Module.

4 Methodology

Based on the definition of GRAG, we proposed a new Retrieval-Augmented Generation with Neural Graph Matching, named NGM-RAG in Figure 2.

4.1 Graph Construction

Given a dataset corpus, we first leverage LLMs to build a target knowledge graph $G_{\text{target}} = (V_{\text{target}}, E_{\text{target}})$, a representation widely adopted in GRAG (Guo et al., 2024), where V_{target} is the node set and E_{target} is the edge set, respectively. For a given the query, we construct its graph G_{query} in the same manner. Each node and edge is annotated with its corresponding textual information, including name, type and description for a node and description for an edge shown in Figure 2. More detail about prompts can be found in Section A.3.1. It is worth noting that the Graph Construction process serves for data distillation and indexing. Therefore, the Graph Indexer Module in GRAG can be formed as:

$$\mathcal{I}^{G} = \text{LLM}_{\text{prompt}}(q, D) \rightarrow G_{\text{target}}, G_{\text{query}}, \tag{5}$$

where $\text{LLM}_{\text{prmopt}}$ is a powerful LLM with specific prompts for knowledge graph extraction[1].

4.2 Graph Matching

After completing the graph construction in Section 4.1, We aim to identify the most relevant nodes in G_{target} with respect to G_{query}. This is a graph matching problem. For each node $v \in G_{\text{query}}$, we aim to find the top K most similar nodes in G_{target} based on the matching score $f(v, u)$. Formally,

$$\mathcal{N}_{K}(v) = \text{TopK}_{u \in V_{\text{target}}} f(v, u), \tag{6}$$

where $f(v, u)$ denotes the matching score between node v and node u, $\mathcal{N}_{K}(v)$ denotes the set of K nodes in G_{target} that have the highest matching scores with v. We design three different approaches to compute the matching score: *Direct Matching*, *Text Similarity*, and *Graph Neural Matching*.

4.2.1 Direct Matching

Direct Matching selects target nodes whose names match those in the query graph. Since LLM-extracted node names may vary across graphs, we use the Levenshtein distance (Yujian and Bo, 2007) to compute name-level similarity:

$$S_{\text{levenshtein}} = \text{Levenshtein}(v.\text{name}, u.\text{name}), \tag{7}$$

[1] In this work, we used GPT-4o-mini for our method and baselines.

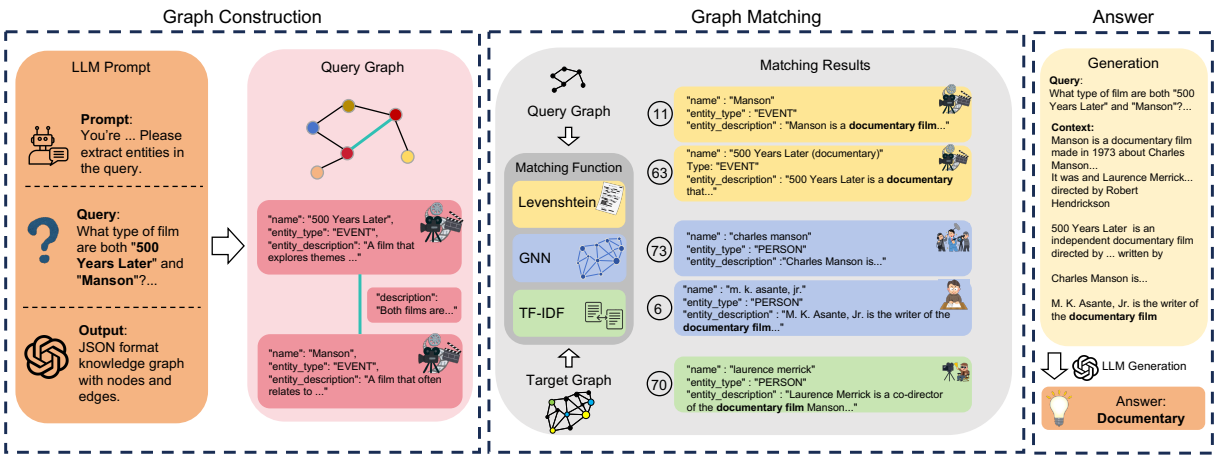

Figure 2: Overall workflow of our NGM-RAG, which consists of multiple components, including Graph Construction, Graph Matching, and Context and Answer Generation.

where v.name and u.name denote the names of query node v and target node u, respectively.

4.2.2 Text Similarity

To capture semantic relevance beyond exact name matching, we measure the textual similarity between node descriptions. We first use TF-IDF (Schütze et al., 2008) as a basic formulation:

$$S_{\text{tfidf}} = \text{TFIDF}(v.\text{description}, u.\text{description}). \tag{8}$$

In practice, we adopt BM25 (Robertson et al., 2009), which further considers term saturation and document length normalization:

$$S_{\text{BM25}} = \sum_{i=1}^{n} W_i \cdot R(q_i, d), \tag{9}$$

where q_i is the i-th query term, W_i is its weight, and $R(q_i, d)$ measures its relevance to document d.

4.2.3 Neural Graph Matching

Direct Matching and Text Similarity mainly rely on textual signals and do not fully exploit graph topology. To incorporate structural dependencies, we use GNNs to encode nodes and compute graph-aware matching scores:

$$S_{\text{gnn}} = \text{Cosine}(\text{GNN}(v), \text{GNN}(u)), \tag{10}$$

where $\text{GNN}(v)$ and $\text{GNN}(u)$ are the graph-enhanced representations of nodes v and u, respectively. We initialize node features with BGE embeddings (Xiao et al., 2023) and adopt LightGCN (He et al., 2020) and GINE (Hu et al., 2019) as representative parameter-free homogeneous and parameterized heterogeneous GNN backbones.

For LightGCN, node representations are updated by parameter-free neighborhood aggregation:

$$x_v^l = \sum_{j \in \mathcal{N}_v} \frac{1}{\sqrt{|\mathcal{N}_v|}\sqrt{|\mathcal{N}_j|}} x_j^{l-1}, \tag{11}$$

where $\mathcal{N}_v$ and $\mathcal{N}_j$ denote the neighbors of nodes v and j, respectively.

For GINE, the propagation rule is:

$$x_v^l = h_{\mathbf{\Theta}}\left(x_v^{l-1} + \sum_{j \in \mathcal{N}_v} \text{ReLU}(x_j^{l-1} + e_{jv})\right), \tag{12}$$

where $h_{\mathbf{\Theta}}$ is a neural network and e_{jv} denotes edge attributes. Since GINE contains trainable parameters, we use node pairs retrieved by Direct Matching and Text Similarity as supervision signals, and optimize the model with contrastive loss (Chen et al., 2020). Let $s_i^+ = \text{sim}(x_i, x_i^+)$ and $s_i^- = \text{sim}(x_i, x_i^-)$. The contrastive loss is defined as:

$$\mathcal{L} = -\sum_i \log \frac{\exp(s_i^+/\tau)}{\exp(s_i^+/\tau) + \exp(s_i^-/\tau)}. \tag{13}$$

where x_i^+ and x_i^- are positive and negative samples for x_i, respectively, and $\text{sim}(\cdot)$ denotes cosine similarity.

4.2.4 Overall Ranking

We combine the three matching signals through adaptive weighting:

$$S_{\text{ranking}} = S_{\text{levenshtein}} + \lambda_1 S_{\text{BM25}} + \lambda_2 S_{\text{gnn}}, \tag{14}$$

where λ_1 and λ_2 balance textual and structural matching signals. For each query, we select the top-k nodes with the highest S_{ranking} scores as contextual evidence for answer generation.

5 Experiments

We conduct experiments and evaluations on different benchmark datasets to verify these research questions (RQ):

- **RQ1**: How does NGM-RAG perform in different question answering tasks?
- **RQ2**: How do different parameter settings in NGM-RAG affect its performance?
- **RQ3**: How do these individual components of NGM-RAG influence its performance?
- **RQ4**: How does NGM-RAG balance cost and model performance?

5.1 Experiments Setup

5.1.1 Datasets and Baselines

Datasets. We evaluate NGM-RAG on three benchmarks: **HotpotQA** (Yang et al., 2018), **MultiHop-RAG** (Tang and Yang, 2024), and the **UltraDomain benchmark** (Qian et al., 2025). HotpotQA and MultiHop-RAG are used for multi-hop question answering, while UltraDomain is used to evaluate long-context summarization.

For UltraDomain, we follow the question generation protocol of GraphRAG (Edge et al., 2024). For each of two domains, an LLM simulates five RAG users, each assigned five tasks. For each user-task pair, the LLM generates five questions based on textual summaries of the corpus, resulting in 125 questions per domain.

Baselines. We compare NGM-RAG with representative RAG and GRAG baselines, including **NaiveRAG** (Gao et al., 2023), **GraphRAG** (Edge et al., 2024), **LightRAG** (Guo et al., 2024), **PathRAG** (Chen et al., 2025), and **MiniRAG** (Fan et al., 2025).

5.1.2 Implementation Details

Question Selection. We sample 500 questions from HotpotQA and 500 from MultiHop-RAG for multi-hop QA evaluation. For HotpotQA, we use the Distractor Setting, where most documents are irrelevant, requiring models to identify supporting evidence from noisy contexts.

Evaluation Metrics. For HotpotQA and MultiHop-RAG, we report Exact Match (EM) and F1. EM measures whether the prediction exactly matches the ground-truth answer, while F1 evaluates word-level overlap between the prediction and reference.

For long-context summarization, we follow the pairwise LLM-based evaluation protocol of GraphRAG (Edge et al., 2024). Given a question, target metric, and two candidate answers, the evaluator selects the better answer for each dimension and then determines the overall winner.

Backbone LLMs. We use four types of backbone models for answer generation: GPT-4o-mini, Llama-3.1-8B, o1-mini, and DeepSeek-R1, including both the 8B model and the full-size API version. This selection covers both conversational and reasoning-oriented models across different scales.

LLM for Basic Operations. For fairness, GPT-4o-mini is used for all LLM-based auxiliary operations, including graph construction, UltraDomain task and question generation, and win-rate evaluation.

Hyperparameter Settings. Unless otherwise specified, we set the top-K matched nodes to 5 and use $\lambda_1 = \lambda_2 = 1.0$ for BM25 and GNN matching weights.

5.2 Results and Performance Analysis (RQ1)

Table 1 compares NGM-RAG, implemented with LightGCN and GINE, against baseline methods on HotpotQA and MultiHop-RAG across multiple backbone models. The results show that NGM-RAG consistently improves multi-hop question answering performance, achieving the best average results in most settings. Its gains are more evident with stronger backbone models, while the margin over LightRAG and GraphRAG becomes smaller on weaker models. Overall, NGM-RAG delivers stable improvements across datasets and model architectures, validating the robustness of graph neural networks in multi-hop reasoning and their adaptability to various model architectures.

On the UltraDomain benchmark, we evaluate methods on the Agriculture and Mix datasets using comprehensiveness, diversity, empowerment, directness, and overall win rate, as shown in Table 2. NGM-RAG achieves the best overall win rate on both datasets and shows strong performance across multiple dimensions. Although LightRAG obtains higher diversity on Agriculture, its keyword-based retrieval may introduce redundant information which leads additional expenses or errors. In contrast, NGM-RAG better balances richness and precision, leading to more reliable long-context question answering.

HotpotQA	NaiveRAG		LightRAG		GraphRAG		PathRAG		MiniRAG		NGM (L)		NGM (G)	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Llama3.1:8B	0.186	0.286	0.278	0.378	0.122	0.206	0.210	0.294	0.010	0.159	**0.326**	**0.441**	0.298	0.431
GPT-4o-mini	0.272	0.376	0.358	0.505	0.304	0.403	0.392	0.516	0.394	0.507	0.424	0.564	**0.440**	**0.572**
Deepseek-R1:8B	0.186	0.283	0.278	0.378	0.114	0.195	0.162	0.242	0.146	0.189	0.278	**0.414**	**0.280**	0.411
Deepseek-R1(API)	0.368	0.489	0.336	0.470	0.452	0.582	0.408	0.537	0.352	0.476	0.500	0.649	**0.506**	**0.654**

MultiHop-RAG	NaiveRAG		LightRAG		GraphRAG		PathRAG		MiniRAG		NGM (L)		NGM (G)	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Llama3.1:8B	0.554	**0.578**	0.482	0.506	0.320	0.422	0.474	0.530	0.286	0.321	**0.556**	0.566	0.536	0.545
GPT-4o-mini	0.616	0.630	0.564	0.574	0.576	0.585	0.616	0.629	0.574	0.586	0.620	0.628	**0.630**	**0.637**
Deepseek-R1:8B	0.352	0.391	0.444	0.462	0.554	0.565	0.452	0.485	0.452	0.462	**0.570**	**0.590**	0.530	0.543
Deepseek-R1(API)	0.534	0.540	0.470	0.476	0.548	0.557	0.490	0.499	0.540	0.551	**0.600**	**0.608**	0.578	0.589

Table 1: The performance of NGM-RAG and baseline methods is evaluated using EM and F1. NGM (L) and NGM (G) represent two modes of NGM-RAG using homogeneous LightGCN and heterogeneous GINE, respectively.

Metrics	Agriculture				Mix			
	NaiveRAG	**NGM-RAG**	LightRAG	**NGM-RAG**	NaiveRAG	**NGM-RAG**	LightRAG	**NGM-RAG**
Comprehensiveness	4.00%	**96.00%**	41.60%	**58.40%**	20.00%	**80.00%**	28.80%	**71.20%**
Diversity	18.40%	**81.60%**	**77.60%**	32.80%	35.20%	**64.80%**	49.60%	**50.40%**
Empowerment	7.20%	**92.80%**	49.60%	**50.40%**	23.20%	**76.80%**	31.20%	**68.80%**
Overall	4.80%	**95.20%**	41.60%	**58.40%**	22.40%	**77.60%**	29.60%	**70.40%**

Table 2: Comparison of dimension win rate and total win rate between NGM-RAG and baseline methods on the Agriculture and Mix datasets.

Based on the above analysis, we conclude the following:

- **NGM-RAG is a versatile method for diverse tasks**: Its graph-based architecture delivers strong performance in both multi-hop question answering and long-context pairwise comparison, demonstrating broad applicability and adaptability.

- **NGM-RAG significantly enhances RAG task performance with stability**: Experimental results show significant improvements across various models and settings, highlighting its technical robustness and reliability in complex retrieval-augmented generation tasks.

5.3 Parameter Analysis (RQ2)

In this section, we conduct a series of parameter sensitivity analyses to evaluate how different configurations affect the performance of NGM-RAG.

5.3.1 Impact of Retrieval Depth k

We study the effect of retrieval depth k using Llama3.1:8B. As shown in Table 3, EM and F1 improve as k increases from 1 to 5, with F1 peaking at $k = 5$ (0.441). Although $k = 10$ slightly improves EM to 0.330, its lower F1 suggests that excessive retrieval may introduce redundancy or noise, weakening generation quality.

Top-k Retrieved	EM	F1
$k = 1$	0.240	0.345
$k = 3$	0.294	0.388
$k = 5$	0.326	**0.441**
$k = 10$	**0.330**	0.432

Table 3: HotpotQA performance of NGM-RAG under varying k values.

5.3.2 Impact of Weight

We conduct a detailed analysis of two critical hyperparameters, λ_1 and λ_2, which govern the weights assigned to text similarity and neural graph matching, respectively. To understand their impact on model performance, we vary each parameter across $\{0, 1, 5, 10\}$.

As illustrated in Figure 3, text similarity module exhibits its optimal effectiveness at $\lambda_1 = 1$, while neural graph matching module remains relatively robust under this condition. We observe that the performance differences across nearby weight choices are generally minor, which we attribute in part to the inherent stochasticity of LLM-generated outputs rather than a consistent trend induced by the weighting scheme itself. Consequently, assigning a uniform weight ($\lambda_1 = \lambda_2 = 1$) serves as a balanced, fair, and empirically sound configuration for NGM-RAG.

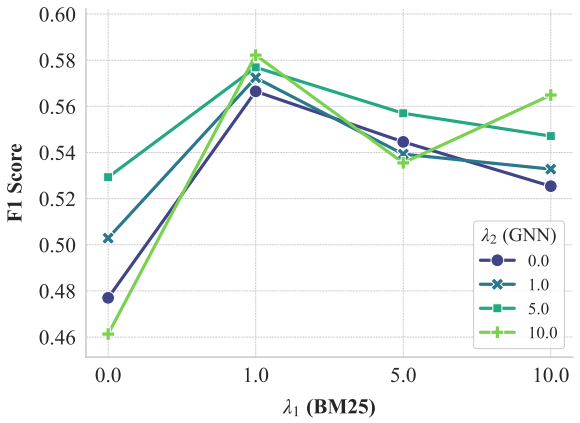

Figure 3: Impact of λ_1 (BM25 weight) and λ_2 (GNN weight) on HotpotQA performance.

GNN Matching	Text Similarity	EM	F1
✗	✗	0.367	0.477
✓	✗	0.410	0.529
✗	✓	0.410	0.545
✓	✓	**0.424**	**0.564**

Table 4: HotpotQA performance of NGM-RAG in ablation experiments.

5.4 Ablation Study (RQ3)

The matching step in our method integrates three key modules: direct matching, text similarity, and neural graph matching. To assess the individual contributions of these components, we performed ablation experiments by sequentially removing the neural graph matching and text similarity modules.

As shown in Table 4, all three modules contribute to the final performance. Ablation studies highlight the importance of all three modules for optimal performance. Direct matching ensures that nodes with identical or similar names are matched. The text similarity module, based on TF-IDF, effectively identifies relevant content by weighting term importance—crucial for accurate retrieval in multi-hop question answering. Neural graph matching captures complex semantic relationships, further enhancing matching accuracy. Together, these modules ensure robust and precise performance.

5.5 Cost Analysis (RQ4)

We evaluate the efficiency of NGM-RAG against baselines using two metrics: **Token Cost**, the number of input tokens per query, and **Inference Latency**, the end-to-end time from query input to final response.

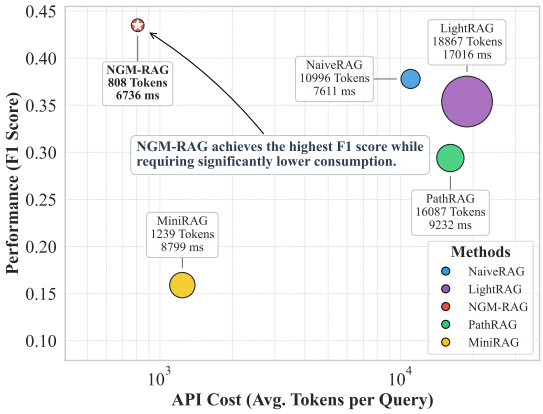

Figure 4: Inference cost comparison. Bubble size indicates runtime. NGM-RAG strikes a balance between efficiency and performance.

As shown in Figure 4, NGM-RAG achieves the best balance between cost and performance. It obtains the highest F1 score while reducing the average token cost to 808.72 tokens, over 90% lower than NaiveRAG with 10,996.07 tokens. NGM-RAG also achieves lower latency than complex graph-based baselines, showing that neural graph matching can remove redundant context and alleviate the high-cost, high-latency bottleneck of existing GRAG methods.

6 Conclusion

In this paper, we present NGM-RAG, a graph matching framework for Retrieval-Augmented Generation. By integrating Levenshtein-based direct matching, BM25-based text similarity, and GNN-based neural graph matching, NGM-RAG improves retrieval precision and captures both textual and structural evidence. Experiments on multi-hop QA and long-context summarization show that NGM-RAG consistently outperforms strong RAG and GRAG baselines while reducing token cost and latency. These results demonstrate the effectiveness and efficiency of neural graph matching for complex RAG tasks.

Acknowledgements

This work is supported by the Natural Science Foundation of China (No.62402398, No.72374173), Lab of High Confidence Embedded Software Engineering Technology and the High Performance Computing clusters at Southwest University.

Limitations

While our method exhibits strong performance, it is important to acknowledge its limitations, which highlight areas for future improvement. First, the computational complexity of graph matching can significantly increase both training and inference time, particularly when applied to large-scale datasets. This results in substantial costs associated with graph construction and model training, potentially limiting its practicality in real-time or resource-constrained scenarios. Second, although NGM-RAG achieves impressive results in multi-hop reasoning and long-context summarization, its applicability to other tasks, such as open-domain question answering or dialogue systems, remains unverified. Further research is needed to assess its generalization capabilities across diverse domains. In addition, in pairwise comparisons of long context summaries, using a large language model for evaluation may lead to slightly unstable results. Finally, the current implementation relies on pre-trained language models and graph neural networks, which may inherit biases present in the training data. Addressing these limitations—through optimization of computational efficiency, exploration of broader applications, and mitigation of data biases—will be a central focus of our future work.

Potential Risks

While our method demonstrates promising results, there are several potential risks that need to be considered. First, the reliance on pre-trained language models and graph neural networks may introduce biases present in the training data, which could lead to unfair or inaccurate outcomes in certain scenarios. Second, the computational complexity of graph matching and the integration of multiple similarity metrics may result in high resource consumption, making it less accessible for users with limited computational resources. Third, the performance of NGM-RAG is highly dependent on the quality of the input data, such as the accuracy of node names and the relevance of the retrieved documents. Noisy or incomplete data could significantly degrade the model's effectiveness. Finally, the generalization of our method to other tasks and domains remains to be thoroughly tested, and there is a risk that it may not perform as well in different contexts. Addressing these risks will be crucial for the broader adoption and reliability of NGM-RAG.

References

Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. 2025. Pathrag: Pruning graph-based retrieval augmented generation with relational paths. *arXiv preprint arXiv:2502.14902*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR.

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

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *EACL 2024 Demos*, pages 150–158.

Tianyu Fan, Jingyuan Wang, Xubin Ren, and Chao Huang. 2025. Minirag: Towards extremely simple retrieval-augmented generation. *arXiv preprint arXiv:2501.06713*.

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

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *ArXiv*, abs/1706.02216.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR 2020*, pages 639–648.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. *arXiv: Learning*.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, Jure Leskovec, et al. 2020. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

Tuan-Anh Phan, Ngoc-Dung Ngoc Nguyen, and Khac-Hoai Nam Bui. 2022. Hetergraphlongsum: Heterogeneous graph neural network with passage aggregation for extractive long document summarization. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6248–6258.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. 2025. Memorag: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM Web Conference 2025 (TheWebConf 2025)*, Sydney, Australia. ACM. ArXiv:2409.05591.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Indradyumna Roy, Venkata Sai Baba Reddy Velugoti, Soumen Chakrabarti, and Abir De. 2022. Interpretable neural subgraph matching for graph retrieval. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8115–8123.

Alireza Salemi and Hamed Zamani. 2024. Evaluating retrieval quality in retrieval-augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2395–2400.

M. Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. In *Extended Semantic Web Conference*.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Yixuan Tang and Yi Yang. 2024. Multihop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In *First Conference on Language Modeling*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. 2017. Graph attention networks. *ArXiv*, abs/1710.10903.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. Pmlr.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024. Do large language models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10210–10229.

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 *EMNLP2018*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024. Knowledge graph enhanced large language model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22647–22662.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

A Appendix

A.1 Datasets

In this section, we provide a basic introduction to the dataset used.

HotpotQA (Yang et al., 2018): A large-scale multi-hop QA dataset with 113,000 Wikipedia-based QA pairs. It requires reasoning across multiple documents and provides sentence-level supporting facts, making it ideal for evaluating explainable multi-hop reasoning.

MultiHop-RAG (Tang and Yang, 2024): A benchmark with 2,556 metadata-rich queries, each requiring reasoning over 2–4 documents. It simulates real-world RAG scenarios by combining document retrieval and multi-hop reasoning.

UltraDomain benchmark (Qian et al., 2025): A long-context QA dataset derived from 428 textbooks across 18 domains. We use the **Agriculture** and **Mix** subsets, which contain 2,017,886 tokens and 619,009 tokens respectively. This dataset tests the robustness of the model in both technical and humanities domains through a large amount of text input.

A.2 Baselines

In this section, we introduce the baselines against which we compare NGM-RAG.

- **NaiveRAG** (Gao et al., 2023): Naive RAG combines LLMs with vector-based retrieval from chunked external documents. The method segments raw text into chunks and stores them in a vector database using embeddings. It retrieves top-matching text via embedding similarity.

- **GraphRAG** (Edge et al., 2024): GraphRAG extracts entities and relationships from the source documents, then represents them as nodes and edges in a graph. It handles global queries across corpora by generating community summaries for related groups of entities and enhances retrieval performance.

- **LightRAG** (Guo et al., 2024): LightRAG uses a two-tier retrieval mechanism based on knowledge graphs, focuses on both low-level details and high-level concepts of queries, and supports incremental updates.

- **PathRAG** (Chen et al., 2025): PathRAG explicitly models relational paths within an indexing graph to jointly enhance retrieval effectiveness and generation coherence.

- **MiniRAG** (Fan et al., 2025): MiniRAG is a lightweight RAG framework designed for environments with limited computing resources.

A.3 Prompts

In this section, we provide a detailed overview of the prompts used throughout the various components of our pipeline. Each prompt is tailored to a specific task—ranging from knowledge graph construction to long-context answer generation and qualitative answer assessment. The prompts are designed to elicit consistent, structured, and task-relevant responses from large language models (LLMs), supporting both upstream (e.g., graph and question generation) and downstream (e.g., QA and evaluation) components in the retrieval-augmented generation (RAG) framework.

A.3.1 Graph Construction

In this section, we describe the prompts used to extract knowledge graphs from the query and corpus for the RAG task.

Figure 5 presents a prompt of entities and relations extracted from the corpus during the graph building process. The model is instructed to identify entities based on specified types, describe them, and detect meaningful relationships between them, including explanations, strength scores, and thematic keywords. The output also includes high-level content keywords and is formatted as a structured list in the specified language.

A.3.2 Question Generation

In this section, we describe the prompts used to generate global sensemarking questions via specific corpus.

Figure 6 shows a prompt designed to generate high-level sensemaking questions based on a dataset description. It instructs the model to identify five potential users, list five tasks for each user, and generate five questions per user-task pair that require a comprehensive understanding of the dataset. The prompt facilitates structured exploration of the dataset's relevance and utility from multiple perspectives.

A.3.3 Multihop-QA Answer Generation

In this section, we describe the prompt used to generate concise answers for multihop question an-

swering tasks, emphasizing direct responses without supporting context.

Figure 7 presents a prompt designed for generating answers to multihop question answering (QA) tasks. The prompt instructs the model to answer queries using the provided context when relevant, or to generate answers directly when context is insufficient. It emphasizes brevity and prohibits any explanation or elaboration, requiring direct answers in either factual or yes/no form, depending on the question type. This setup ensures consistent, concise responses.

A.3.4 Long-Context Answer Generation

In this section, we describe the prompt used to generate comprehensive and evidence-based answers for long-context question answering, encouraging contextual summarization and the cautious integration of common knowledge.

Figure 8 shows a prompt designed for generating answers in long-context question answering scenarios. The prompt guides the model to summarize all relevant information from the provided context and optionally integrate common knowledge, while strictly avoiding unsupported content. This ensures that answers are both comprehensive and grounded in evidence.

A.3.5 LLM Assessments Generation

In this section, we describe the prompt used to judge and compare two LLM-generated answers across multiple qualitative metrics, producing structured and justified judgments to support consistent comparative analysis.

Figure 9 presents a prompt for evaluating two LLM-generated answers to the same question using four metrics: comprehensiveness, diversity, empowerment, and directness. The model is asked to select a winner for each metric with justification, then determine an overall winner. The output is structured in a standardized JSON format to support consistent and explainable comparative assessments.

A.4 Experiment based on TF-IDF

In the preliminary phase of our experiments, the Text Similarity module was implemented using the classic TF-IDF algorithm. However, subsequent comparative studies demonstrated that the BM25 algorithm yielded superior performance for our specific task. Consequently, the pure TF-IDF implementation was replaced by BM25 in the final model presented in the main text. As shown in Table 5 and Table 6, to ensure completeness and provide a comparative baseline, the results from the initial TF-IDF experiments are included as supplementary experiments in the Appendix.

HotpotQA	**NGM-RAG(LightGCN)**		**NGM-RAG(GINE)**	
	EM	**F1**	**EM**	**F1**
Llama3.1:8B	0.286	0.401	0.331	0.414
GPT-4o-mini	0.382	0.526	0.370	0.503
Deepseek-R1:8B	0.238	0.353	0.222	0.341
Deepseek-R1:671B	0.440	0.594	0.430	0.578

MultiHop-RAG	**NGM-RAG(LightGCN)**		**NGM-RAG(GINE)**	
	EM	**F1**	**EM**	**F1**
Llama3.1:8B	0.496	0.522	0.472	0.490
GPT-4o-mini	0.642	0.654	0.648	0.658
Deepseek-R1:8B	0.476	0.499	0.480	0.498
Deepseek-R1:671B	0.588	0.612	0.558	0.577

Table 5: The performance of NGM-RAG variants (LightGCN and GINE) evaluated using EM and F1 on the HotpotQA and MultiHop-RAG datasets based on TF-IDF

Agriculture Dataset				
Metrics	vs NaiveRAG		vs LightRAG	
	NaiveRAG	**NGM-RAG**	LightRAG	**NGM-RAG**
Comprehensiveness	0.00%	**100.00%**	19.20%	**80.80%**
Diversity	8.00%	**92.00%**	**67.20%**	32.80%
Empowerment	0.00%	**100.00%**	28.80%	**71.20%**
Overall	0.00%	**100.00%**	23.20%	**76.80%**

Mix Dataset				
Metrics	vs NaiveRAG		vs LightRAG	
	NaiveRAG	**NGM-RAG**	LightRAG	**NGM-RAG**
Comprehensiveness	17.60%	**82.40%**	25.60%	**74.40%**
Diversity	34.40%	**65.60%**	48.00%	**52.00%**
Empowerment	19.20%	**80.80%**	25.60%	**74.40%**
Overall	19.20%	**80.80%**	25.60%	**74.40%**

Table 6: Comparison of win rates between NGM-RAG and baseline methods on the Agriculture and Mix datasets (TF-IDF based). Results are split vertically for single-column layout.

A.5 Case Study: Multihop-QA example

This case involves a complex multi-hop question requiring identification of a country based on cues from magazine names and historical events. It evaluates each method's ability to integrate dispersed information for accurate reasoning.

NaiveRAG relies on semantic vector similarity but tends to associate high-frequency terms with dominant entities like the United States, resulting in reasoning bias in long queries.

GraphRAG employs complex matching but may misclassify questions on smaller models, occasionally producing irrelevant response type.

LightRAG extracts salient keywords but lacks

semantic depth, misattributing the context to Egypt due to noise sensitivity.

NGM-RAG, by integrating direct matching, text similarity, and neural graph matching, accurately identifies key nodes and relations (e.g., "often featured in"), enabling correct inference *Israel*. Its robustness stems from a multi-module fusion architecture that balances semantic depth and structural alignment.

Overall, NGM-RAG shows clear advantages in complex multi-hop QA, with results detailed in Table 7.

Query	Which country, often ... to prevent major attacks?
Correct Answer	Israel
NaiveRAG	The United States
GraphRAG	Yes
LightRAG	Egypt
NGM-RAG	Israel

Table 7: Case Study:Example questions from the MultiHop-RAG dataset, and the answers generated by the baseline method and NGM-RAG when using Deepseek-R1 as the foundation model.

A.6 Case Study: Long-Context QA example

This case focuses on a question-answering task from the Agriculture dataset, comparing the performance of LightRAG and NGM-RAG in generating summarization-based answers. The experimental results demonstrate the strengths of each method in addressing the query. LightRAG shows better performance in Diversity, providing analysis and explanations from multiple perspectives. However, NGM-RAG outperforms LightRAG in other key metrics, including Comprehensiveness, Empowerment, and Directness, resulting in a higher overall evaluation score. The detail results are shown in Table 8.

LightRAG's strength lies in its ability to generate diverse responses by incorporating various angles of analysis. This is particularly evident in its coverage of different aspects related to the query, which enhances the richness of the generated content. However, its focus on diversity sometimes comes at the cost of depth and precision, leading to less comprehensive and direct answers. NGM-RAG, on the other hand, excels in delivering comprehensive and direct answers. By leveraging its multi-module fusion design, NGM-RAG effectively balances semantic understanding, contextual relevance, and factual accuracy. This enables it to provide answers that are not only detailed but also highly relevant and actionable, as reflected in its superior performance in Comprehensiveness, Empowerment, and Directness.

The results of this case study highlight the trade-offs between diversity and depth in summarization tasks. While LightRAG's diverse outputs are valuable for exploring multiple perspectives, NGM-RAG's ability to deliver comprehensive and direct answers makes it more suitable for applications requiring precise and actionable insights. This finding underscores the importance of selecting methods based on specific task requirements and evaluation metrics.

A.7 Evaluation Metric Definitions

To support the qualitative evaluation of long-context answers, we adopt these human-aligned metrics introduced in GraphRAG. Below are the definitions used during model assessment:

- **Comprehensiveness**: Measures how thoroughly the answer addresses all aspects of the question, reflecting the level of detail and coverage provided.
- **Diversity**: Evaluates the richness and variety of perspectives or supporting points included in the answer, indicating the breadth of reasoning and insight.
- **Empowerment**: Assesses the answer's ability to enhance the reader's understanding and help them form informed judgements about the topic.

After calculating these metrics, the LLM judge will evaluate the overall performance and determine the overall winner.

Prompt for entity and relationship 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}` (default to English)as the output language.
Steps:

1. Identify all entities. For each identified entity, extract the following information:
   - **entity_name**: Name of the entity, using the same language as the input text. If English, capitalize the name.
   - **entity_type**: One of the following types: [`{entity_types}`]
   - **entity_description**: A comprehensive description of the entity's attributes and activities.
2. From the entities identified in Step 1, identify all pairs of (source_entity, target_entity) that are clearly related to each other. For each pair, extract the following:
   - **source_entity**: Name of the source entity.
   - **target_entity**: Name of the target entity.
   - **relationship_description**: Explanation of why the two entities are related.
   - **relationship_strength**: A numeric score indicating the strength of the relationship.
   - **relationship_keywords**: High-level keywords summarizing the nature of the relationship.
3. Identify high-level keywords that summarize the main concepts or themes of the entire text.
4. Return all output in `{language}`, as a single list of entities and relationships. Use ## as the delimiter between records.
5. At the end of the output, include the marker: <|COMPLETE|>

Figure 5: The prompt for entity and relationship extraction during graph constructing.

Prompt for global sensemaking question generation

Prompt:
Given the following description of a dataset:
`{total_description}`
Please identify 5 potential users who would engage with this dataset. For each user, list 5 tasks they would perform with this dataset. Then, for each (user, task) combination, generate 5 questions that require a high-level understanding of the entire dataset.
Output the results in the following structure:

User 1: [user description]

Task 1: [task description]
Question 1:
Question 2:
...
Question 5:
Task 2: [task description]
...
Task 5: [task description]

User 2: [user description]

...

User 5: [user description]

...

Figure 6: The prompt for global sensemaking question generation.

Prompt for multihop-QA generating

Prompt:
You are a helpful assistant that uses provided context to answer queries.
If there is relevant information in the context provided, please follow it.
If there is really no relevant content in the relevant context, please generate it yourself.
But please don't explain the answer anyway, just answer.
Example 1:
Question: Who is the individual associated with the cryptocurrency industry facing a criminal trial on fraud and conspiracy charges, as reported by both The Verge and TechCrunch, and is accused by prosecutors of committing fraud for personal gain?
Answer(No explain): Sam Bankman-Fried
Example 2: Question: Were Scott Derrickson and Ed Wood of the same nationality?
Answer(No explain): Yes
If it is not a judgment type, just output the answer as in Example 1.
If your answer is a yes or no judgment, just output Yes and No like Example 2.
All in all, do not explain the answer and don't provide a note either, anyway!
Please output the answer, keep your answers as short as possible. (Even there is no context)
Context: `{context}`
Query: `{query}`

Figure 7: The prompt for multihop-QA generating.

Prompt for long-context generating

Prompt:
You are a helpful assistant that uses provided context to answer queries.
Generate responses to user questions that summarize all information in the context provided and can incorporate any relevant common knowledge.
Do not include information without supporting evidence.
Context: `{context}`
Query: `{query}`

Figure 8: The prompt for long-context generating.

Prompt for LLM assessments generating

Prompt:
You will evaluate two answers to the same question based on these metrics: **Comprehensiveness**, **Diversity** and **Empowerment**.

- **Comprehensiveness**: How much detail does the answer provide to cover all aspects and details of the question?
- **Diversity**: How varied and rich is the answer in providing different perspectives and insights on the question?
- **Empowerment**: How well does the answer help the reader understand and make informed judgments about the topic?

For each metric, choose the better answer (either Answer 1 or Answer 2) and explain why. Then, select an overall winner based on these four categories.

Here is the question:`{query}`

Here are the two answers: Answer 1:`{answer1}` **Answer 2:**`{answer2}`

Evaluate both answers using the four metrics listed above and provide detailed explanations for each metric.

Output your evaluation in the following JSON format:

```
{
    "Comprehensiveness": {
        "Winner": "[Answer 1 or Answer 2]",
        "Explanation": "[Provide explanation here]"
    },
    ...
    "Overall Winner": {
        "Winner": "[Answer 1 or Answer 2]",
        "Explanation": "[Summarize why this answer is the overall winner
        based on the four metrics]"
    }
}
```

Figure 9: The prompt for LLM assessments generating.

Query	What role does consumer demand play in shaping beekeeping practices and sustainability efforts? (Role: Policy Maker, An official involved in creating and implementing agricultural policies.)
LightRAG	Consumer demand significantly influences beekeeping practices and sustainability efforts in several ways, reflecting both environmental considerations and market needs. Shaping Beekeeping Practices: 1. Honey Production Quality[...] 2. Diverse Product Offerings: [...] Sustainability Efforts: 1. Support for Pollinator Health: [...] 2. Certification and Transparency: [...] Conclusion: Ultimately, the interplay between consumer demand and beekeeping practices becomes a catalyst for broader sustainability efforts. As consumers increasingly value sustainability and ethical sourcing, beekeepers are more likely to implement and expand practices that not only fulfill market demand but also enhance the health of bee ecosystems. This synergy fosters a cycle where responsible consumer choices lead to advancements in beekeeping methods, promoting both local economies and environmental sustainability.
NGM-RAG	Consumer demand significantly influences beekeeping practices and sustainability efforts by driving beekeepers to adopt methods that align with the preferences of health-conscious and environmentally aware consumers. As awareness of the declining honeybee populations and the importance of pollinators for food production rises, consumers increasingly seek products that reflect sustainable and ethical practices. 1. Healthier Products: [...] 2. Local Sourcing: [...] 3. Sustainable Practices: [...] 4. Education and Engagement: [...] 5. Market Trends: [...] In summary, consumer demand acts as a powerful catalyst for beekeeping practices, encouraging sustainability and the adoption of methods that support environmental health, bee welfare, and the broader sustainability movement.
LLM Decision	(Answer1: NGM-RAG Answer2: LightRAG) **Comprehensiveness Winner**:Answer 1 **Explanation**: Answer 1 provides a detailed breakdown of how consumer demand influences various aspects of beekeeping practices, including health, local sourcing, sustainable practices, education, and market trends. It covers a wider range of factors and implications compared to Answer 2, which, while informative, is less detailed in its exploration of the topic. **Diversity Winner**:Answer 2 **Explanation**: Answer 2 presents a broader range of perspectives by discussing not only the quality of honey production but also the diversification of products and the importance of certifications. It highlights different aspects of consumer demand and its impact on beekeeping, making it more diverse in its insights compared to Answer 1. **Empowerment Winner**:Answer 1 **Explanation**: Answer 1 empowers the reader by providing specific examples of how consumer demand shapes beekeeping practices and sustainability efforts. It emphasizes the role of education and community engagement, which can help readers understand how they can contribute to sustainable practices and make informed choices. **Overall Winner**:Answer 1 **Explanation**: Overall, Answer 1 is the winner as it excels in comprehensiveness and empowerment, providing a detailed and clear response that directly addresses the question. While Answer 2 offers valuable diversity, it does not match the depth and clarity of Answer 1, making the latter the more effective answer.

Table 8: Case Study: Example question for Agriculture dataset, with generated answers from LightRAG and NGM-RAG, as well as evaluation by LLM