Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 14, No. 1, 2025

Knowledge Graph Combined with Retrieval-Augmented Generation for Enhancing LMs Reasoning: A Survey

Haitao Wang, Yangkun Shi

School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454003, China

Abstract: Large language models (LLMs) have generated significant waves in the fields of Natural Language Processing(NLP) and artificial intelligence due to their remarkable capabilities and broad adaptability. Retrieval-Augmented Generation (RAG) techniques have been widely adopted, leveraging external retrieval systems to significantly improve the timeliness of LLMs and substantially reduce hallucinations. Although RAG and its optimization methods have addressed most hallucination issues caused by knowledge gaps and outdated information, text generation in specialized domains such as law, medicine, and science—which require multi-hop reasoning and analysis—still suffers from a lack of coherence and logical consistency, making it difficult to produce correct and valuable answers. To address these challenges, related research has introduced Knowledge Graphs (KGs). This integrated approach enhances RAG's capabilities by utilizing the structured knowledge provided by KGs, thereby improving the model's knowledge representation and reasoning abilities and enabling the generation of more accurate answers. However, there remains a lack of systematic reviews in this area. Therefore, this paper provides a comprehensive review of studies on enhancing LLMs reasoning abilities by integrating KGs with RAG. It first introduces the basic concepts, followed by an overview of the current mainstream technical approaches, and concludes with a discussion of the research challenges and future development trends in this field.

Keywords: Large Language Models; Natural Language Processing; Retrieval-Augmented Generation; Knowledge Graphs; Knowledge Reasoning.

1. Introduction

In recent years, large language models (LLMs) such as GPT-4[1], Llama-3[2], and Gemini [3] have achieved significant technological breakthroughs in the field of natural language processing, finding wide application in tasks such as text generation, dialogue systems, translation, and summarization. However, despite their impressive performance in general contexts, LLMs still face numerous limitations in specialized domains or knowledge-intensive tasks [4]. Specifically, when handling queries beyond the scope of their training data, these models often produce factually inaccurate or fabricated information, a phenomenon known as "hallucination" [5]. This hallucination not only undermines the reliability of the model's output but also hinders its further adoption in critical application areas such as healthcare, law, and scientific research. Moreover, since LLMs are typically trained on large-scale static datasets, their knowledge is limited to a specific temporal cutoff, making it difficult for them to adapt to the dynamic real-world environment or meet the rapidly evolving demands for specialized knowledge[6]. Consequently, in many cases, users require high-quality information related to the latest literature, real-time data, or domain-specific knowledge, which may not be covered by the model's training data.

To address these challenges, researchers have proposed Retrieval-Augmented Generation (RAG) techniques. RAG integrates LLMs with external knowledge bases or real-time retrieval systems, significantly expanding the model's knowledge coverage and improving the accuracy of its generated content[7]. However, RAG still exhibits notable limitations when dealing with complex knowledge reasoning tasks that require deep reasoning and responsible decision-making. These limitations are particularly pronounced in tasks that demand up-to-date information or highly specialized domain knowledge, such as medical diagnosis, legal consultation, or scientific research.

Although RAG has achieved significant results and has been widely applied across various fields, it still faces several key limitations in practical scenarios:

(i) Neglect of Structured Relationships. Text content does not exist in isolation but is interconnected through complex relational networks. Traditional RAG methods fail to effectively capture such structured relational knowledge, especially knowledge that cannot be directly expressed through semantic similarity. For example, in the citation network of academic papers, the citation relationships between papers contain rich semantic information. However, conventional RAG focuses only on semantic retrieval based on queries, neglecting these critical structured relationships, resulting in underutilization of potential information.

(ii) Introduction of Redundant Information. When retrieved text fragments are incorporated into the generation model via prompts, the content often appears in a repetitive manner, leading to excessively lengthy prompt contexts. This "verbosity effect" can cause the so-called "lost in the middle" phenomenon[8], where the model struggles to focus on core information due to overly long contexts.

(iii) Lack of Global Information. Since RAG can only retrieve a limited number of relevant document fragments, the system lacks a global perspective of the complete corpus. This deficiency leads to suboptimal performance in tasks requiring long-context summarization, such as abstracting or summarizing content.

To further optimize the RAG process and enhance retrieval accuracy, a natural and promising solution is to integrate structured external knowledge, such as Knowledge Graphs (KGs), to improve LLM reasoning. The explicit relational networks in KGs naturally support multi-hop reasoning tasks, where elements such as nodes, triples, paths, or subgraphs can

be utilized to generate responses. KGs-RAG considers the interconnections between texts, enabling more accurate and comprehensive retrieval of relational information.

When combined with RAG, KGs can retrieve relevant entities and relationships from the graph, aiding LLMs in performing complex logical reasoning and causal analysis. Moreover, KGs dynamically provide high-quality information sources, reducing the risk of hallucinations in LLM outputs. This integration also offers high interpretability, as knowledge graphs provide clear knowledge sources and reasoning paths. Together with documents retrieved by RAG, they form a solid basis for supporting the generated content. This combination improves the interpretability of the model's output, allowing users to trace the sources of information.

2. Related Work

2.1. Knowledge Graphs

Knowledge Graph is a framework for representing knowledge in a structured form, capturing entities and their relationships to reflect real-world objects and their semantic associations. A knowledge graph is composed of nodes and edges, where nodes represent entities, and edges denote semantic relationships between these entities. This structured knowledge representation, based on triples <subject, predicate, object>, effectively supports knowledge storage, retrieval, and reasoning. Knowledge graphs were first formally introduced by Google in 2012, aiming to enhance search engine results through structured information[9]. With further research, knowledge graphs have become a cornerstone of artificial intelligence and are widely applied in semantic search, intelligent question answering, recommendation systems, and decision support.

In recent years, multimodal knowledge graphs and dynamic knowledge graphs have emerged as research hotspots. Multimodal knowledge graphs integrate diverse data forms, such as text, images, and videos, to enhance knowledge representation and semantic richness[10]. Dynamic knowledge graphs focus on real-time updates and incremental reasoning in ever-changing environments to meet rapidly evolving business needs[11].

Knowledge graphs can significantly enhance the multi-hop reasoning capabilities of LLMs. By inherently supporting relationship chain tracing between entities, they excel in complex reasoning tasks. For instance, knowledge graph-based question-answering systems can derive answers through multi-step retrieval, a capability relatively weak in traditional LLMs. In terms of content generation interpretability, knowledge graphs provide a reliable explanatory foundation by explicitly showcasing reasoning paths and data sources, which is especially critical in high-stakes fields requiring transparency, such as healthcare and finance.

2.2. Retrieval-Augmented Generation

Retrieval-Augmented Generation is a technique that combines information retrieval and generative models to enhance the capabilities of text generation systems. Unlike traditional generative models that rely solely on training data to produce text, RAG incorporates an external retrieval module to dynamically fetch external knowledge relevant to the user input. This approach improves the accuracy and richness of the generated content, making it particularly suitable for tasks requiring open-domain knowledge, such as question answering, dialogue generation, and automated writing.

The concept of RAG originates from the integration of information retrieval and generative models. Early retrieval-augmented methods were primarily applied in question-answering systems. For instance, DrQA [12] combined traditional retrieval models (e.g., TF-IDF) with simple reader models to provide robust support for natural language question answering. With the advancement of deep learning, models like Dense Passage Retrieval (DPR) [13], which leverage vector-based semantic retrieval, gradually replaced traditional sparse retrieval methods. These approaches use pre-trained models (e.g., BERT) to generate dense vector representations, significantly improving retrieval accuracy. The RAG framework was subsequently introduced by Lewis et al. in 2020 [14], combining a retriever to fetch relevant documents with a generative model (e.g., BART or T5) for answer generation. This framework pioneered the deep integration of retrieval and generation.

Naive RAG follows a classic three-stage process: Indexing, Retrieval, and Generation.

(i) Indexing: First, the system processes and cleans raw data from diverse sources, which may be stored in formats such as PDF, HTML, Word, or Markdown. To standardize knowledge representation, these files are converted into plain text format. Since LLMs typically have limitations on input context length, long documents are further divided into smaller, semantically coherent text chunks to ensure efficient processing and understanding by the model. These chunks are then encoded into vector representations using an embedding model to capture their semantic features. The resulting vectors are stored in an efficient vector database, enabling fast similarity retrieval in subsequent stages.

(ii) Retrieval: When the system receives a user query, it uses the same embedding model from the indexing phase to encode the query into a vector representation. By calculating similarity scores between the query vector and all text chunk vectors in the database, the system effectively identifies the content most relevant to the query. To enhance efficiency and relevance, the system typically retrieves the Top-K text chunks with the highest similarity scores. These highly relevant chunks serve as extended context, providing semantic support for the generation phase.

(iii) Generation: The user query and the relevant document chunks retrieved from the index are combined into a coherent prompt, which is then fed into a LLM to generate the final response. Depending on the task requirements, LLMs can adopt flexible response strategies. One approach leverages the model's inherent parametric knowledge to produce answers, while another strictly constrains the generated content to the retrieved document information to ensure accuracy and verifiability. Additionally, in dialogue scenarios, RAG systems can incorporate dialogue history into the prompt, supporting multi-turn interactions. This design not only enhances the contextual coherence of the generated content but also enables the system to handle complex and dynamic interaction demands.

In Naive RAG, retrieval is achieved by calculating the similarity (e.g., cosine similarity) between the query and document chunk embeddings[15], where the semantic representation capability of the embedding model plays a crucial role. This primarily includes sparse encoders (e.g., BM25)[16] and dense retrievers (e.g., pre-trained language models based on the BERT architecture). The structure of the

Naive RAG framework is illustrated in Figure 1.

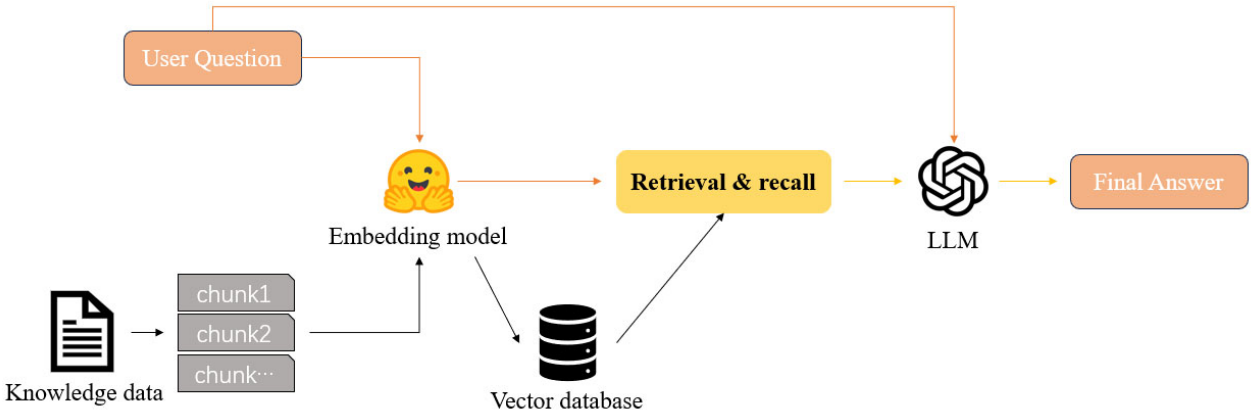

Figure 1. Naive RAG Framework

3. KGs-RAG Technical Approach

The integration of KGs and RAG represents a significant convergence in modern artificial intelligence technologies. KGs, with their structured knowledge representation and semantic reasoning capabilities, address the potential lack of relevance in RAG's retrieval results. Meanwhile, RAG's dynamic retrieval and generation capabilities effectively broaden the application scenarios of KGs. This combination provides robust technical support for fields such as open-domain question answering, dialogue systems, and recommendation systems.

The overall framework of KGs-RAG is illustrated in Figure 2. The entire process can be divided into three key steps: **Index Construction**, **Subgraph Retrieval**, and **Generation Phase**. In the index construction step, NLP techniques are used to extract entities, relationships, and attributes from raw data, while noise is removed to ensure the semantic consistency of the knowledge graph. For example, the sentence "Bill Gates founded Microsoft" can be converted into the triple <Bill Gates, founded, Microsoft>. The construction and indexing of the graph database form the foundation of KGs-RAG, with the quality of the graph database directly influencing its performance. In the subgraph retrieval step, the system parses the user query to identify core entities or keywords and rapidly locates the subgraph relevant to the query from the large-scale knowledge graph. Finally, in the generation phase, the retrieved subgraph serves as contextual information, which, combined with the user query, is fed into the generative model to produce a natural language response that aligns with the semantic requirements.

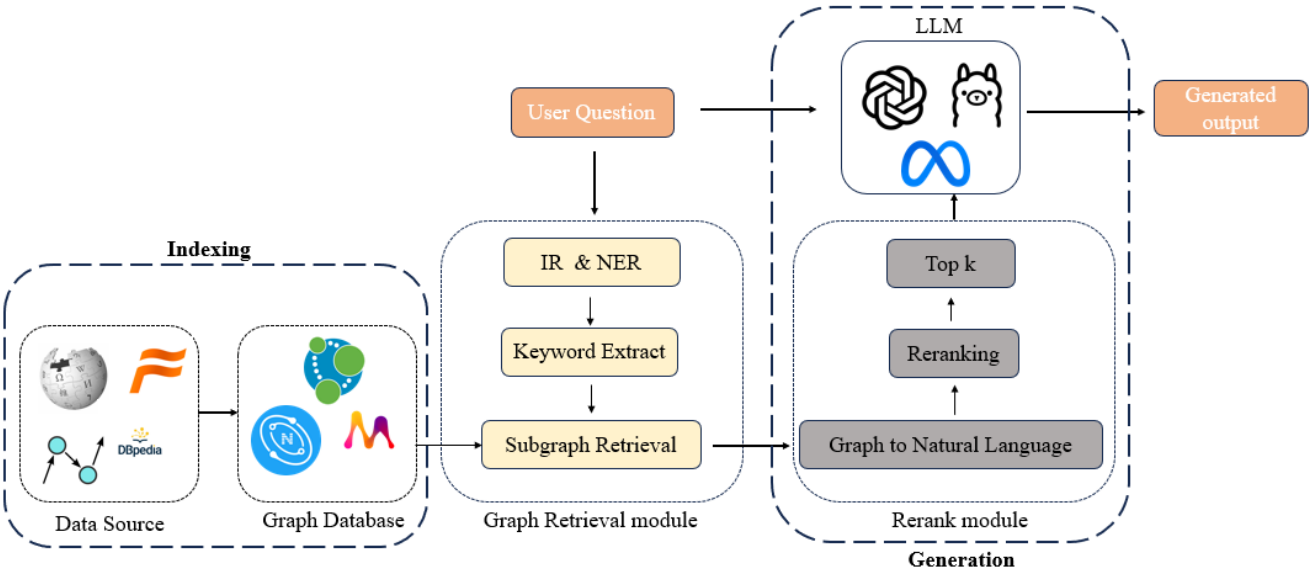

Figure 2. Framework of KGs-RAG

3.1. Index Construction

Knowledge graphs can be classified by data source into open-source knowledge graphs and private knowledge graphs. Based on domain knowledge, they can be further divided into general knowledge graphs and domain-specific knowledge graphs. Common examples of general and domain-specific knowledge graphs are shown in Figure 3.

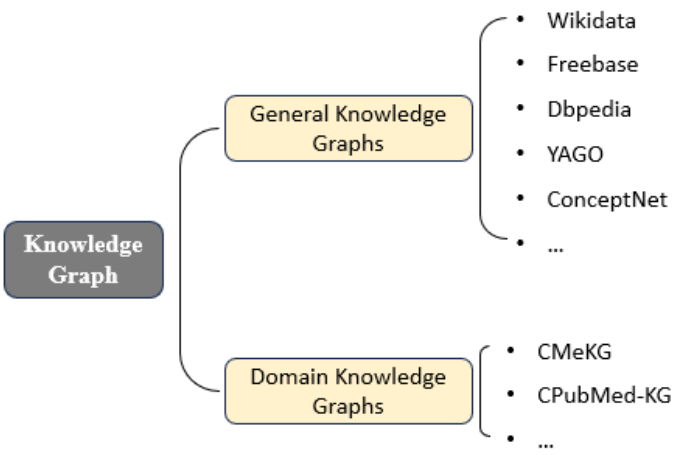

Figure 3. Classification of Knowledge Graphs

Indexing in the construction of KGs is a critical step to ensure efficient data storage and retrieval, enabling rapid access to entities, relationships, and attributes. By establishing a fast localization mechanism, indexing enhances retrieval accuracy and response speed. Common indexing structures include inverted indexes, hash indexes, and graph database-specific indexes, which support semantic queries, path searches, and dynamic updates[17,18]. In recent years, the integration of deep learning techniques, such as entity embeddings and graph neural networks (GNNs), has further optimized indexing performance[19,20].

3.2. Subgraph Retrieval

Subgraph retrieval is a key step in the KGs-RAG framework, responsible for efficiently extracting knowledge subgraphs relevant to the input query from large-scale knowledge graphs, providing a structured knowledge foundation for subsequent generation tasks.

(a) **Path-Based Expansion Retrieval**: This approach constructs subgraphs by expanding from the initial entity within a fixed number of hops, including related nodes and relationships. Common expansion strategies include:

Fixed-Hop Expansion: Restricts the expansion to a specific number of hops, such as retrieving all nodes and relationships within one to two hops from the initial entity.

Semantic-Relevant Expansion: Calculates semantic similarity between nodes and retains only those paths that are semantically relevant to the query.

(b) **Query-Based Subgraph Pattern Matching**: This method maps the query to a graph pattern and retrieves subgraphs from the knowledge graph that match the pattern. For example, for the query "Bill Gates and Microsoft collaboration," a graph pattern containing the nodes "Bill Gates" and "Microsoft" and their possible relationships is constructed to match relevant subgraphs.

(c) **Embedding-Based Semantic Retrieval**: This approach embeds the nodes and edges of the knowledge graph into a high-dimensional vector space, performing subgraph retrieval based on vector similarity. For instance, embedding models like TransE and RotatE can compute the similarity between the query entity and other entities in the knowledge graph to dynamically retrieve subgraphs relevant to the query[21].

Subgraph retrieval is a core component of the KGs-RAG framework, responsible for extracting relevant knowledge subgraphs from large-scale knowledge graphs to provide critical support for generative models. Through methods such as entity matching, path expansion, and semantic retrieval, subgraph retrieval significantly enhances the accuracy and coherence of generated content. As shown in Figure 4, various retrievers in KGs-RAG exhibit unique strengths in handling different aspects of retrieval tasks. Based on their underlying models, retrievers are classified into three types: traditional graph retrieval algorithm-based retrievers, language model-based retrievers, and graph neural network (GNN)-based retrievers.

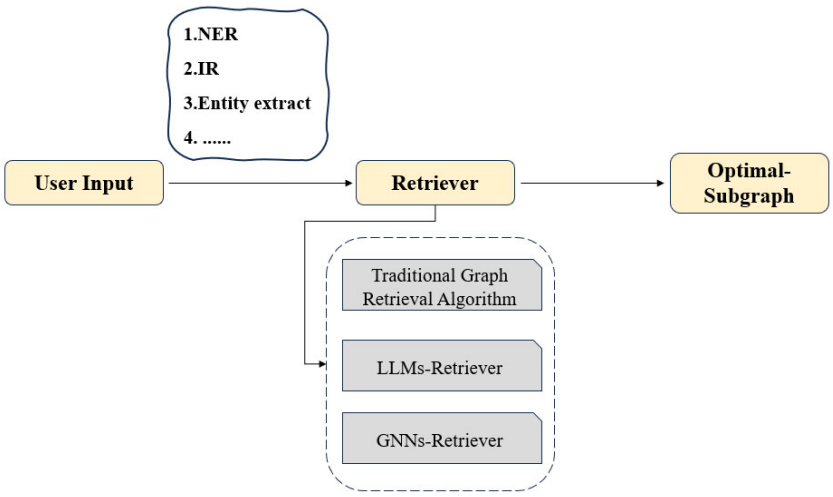

Figure 4. Retriever of KGs-RAG

(a) Methods based on heuristic rules or traditional graph search algorithms enable efficient retrieval without relying on deep learning models. For example, QA-GNN[22] and GrapeQA[23] proposed retrieving k-hop paths from the topic entity for each question-option pair to achieve information extraction. G-Retriever[24] improved the traditional Prize-Collecting Steiner Tree (PCST) algorithm by introducing edge weight mechanisms and optimizing the relevant subgraph extraction process, thereby enhancing retrieval performance. Additionally, methods by Delile et al. (2024)[25] and GNN-RAG[26] first identify relevant entities from the query and then retrieve the shortest paths associated with these entities. Such methods typically include an entity linking preprocessing step to identify target nodes in the graph prior to retrieval.

(b) LMs serve as highly effective retrievers in the KGs-RAG framework due to their powerful natural language understanding capabilities. These models excel in processing and interpreting diverse natural language queries, demonstrating broad applicability in graph-based retrieval tasks. LMs are primarily categorized into two types: discriminative language models and generative language models. Subgraph Retriever[27] utilizes RoBERTa[28] as a trained retriever to expand and retrieve relevant paths from a topic entity through a sequential decision-making process. KG-GPT[29] employs (LLMs to generate the Top-K sets of relations associated with specific entities. Text-to-KG[30] fine-tuned GPT-2 to generate reasoning paths. StructGPT[31] leverages LLMs to automatically invoke multiple predefined functions, retrieving and integrating relevant information to support more complex reasoning tasks.

(c) Graph Neural Networks (GNNs) offer significant advantages in understanding and leveraging complex graph structures. GNN-based retrievers typically encode graph data and score different retrieval granularities based on their similarity to the query, enabling efficient information extraction. For instance, GNN-RAG[26] encodes the graph, assigns relevance scores to each entity, and retrieves entities related to the query based on a predefined threshold. EtD[32] employs an iterative approach to retrieve relevant paths. In each iteration, the method uses LLaMA2[60] to select edges connected to the current node, then calculates embeddings for the next layer of nodes using GNNs, thereby supporting the next round of LLM-based selection.

3.3. Enhanced Generation

In the KGs-RAG framework, the generation phase is another critical step that deeply integrates the retrieved subgraph data from the knowledge graph with the user query, significantly enhancing the accuracy, logic, and semantic coherence of the generated content. This phase not only involves selecting a generation model suitable for the downstream task but also requires effective preprocessing and format conversion of the retrieved graph data to seamlessly incorporate it into the generator's input. Additionally, by introducing techniques for enhanced generation, the interaction between the query and graph data can be further optimized, enriching the diversity and depth of the generated content, ultimately improving the quality of the output and the overall user experience.

3.3.1. Selection of Generation Models

The first step in the generation phase is to select an appropriate generation model based on the specific requirements of the downstream task. These models are typically built on pre-trained deep learning architectures, such as BERT[32], T5[33], the GPT series, or more advanced multimodal models like PaLM-E[34], to ensure that the generated output can handle various task types, including question answering, text summarization, and dialogue generation. The selection of a generation model should consider the following factors:

Task Suitability

Different tasks have varying requirements for generation models. For instance, in open-domain question answering, models capable of handling complex semantics and producing precise answers (e.g., GPT-4) are preferred. In contrast, for scientific literature generation or technical document drafting, models capable of generating coherent long-form text with domain-specific knowledge (e.g., T5-11B) are more appropriate.

Model Scalability and Reasoning Capability

The generation model should possess the ability to infer implicit relationships from graph data. Integrating graph neural networks (GNNs) and knowledge graph embeddings can enhance the model's performance in generating responses for semantically complex queries.

Computational Efficiency and Real-Time Responsiveness

In scenarios requiring real-time response generation (e.g., dialogue systems), inference speed and computational cost are critical considerations. Lightweight generation models or those optimized using knowledge distillation techniques may offer significant advantages.

3.3.2. Preprocessing and Format Conversion of Graph Data

The core of the generation phase lies in converting the retrieved graph data into an input format directly usable by the generation model. This process involves the following key steps:

Linearization of Structured Knowledge

Knowledge graphs are typically represented in the form of triples (e.g., <entity1, relationship, entity2>), while generation models usually process natural language text. Therefore, the graph data needs to be linearized into a natural language format. For example, the triple <Bill Gates, founded, Microsoft> can be transformed into "Bill Gates founded Microsoft."

Semantic Expansion of Nodes and Relationships

During format conversion, semantic descriptions of nodes and relationships can be added to enrich the context of the graph data. For instance, the node "Bill Gates" could be expanded to "Bill Gates, the co-founder of Microsoft."

Context Concatenation and Query Fusion

To enhance the relevance between the graph data and the user query, the graph data should be concatenated with the query into a complete input sequence. For example, for the query "Who founded Microsoft?", the concatenated input could be: "Query: Who founded Microsoft? Context: Bill Gates founded Microsoft; Microsoft is a technology company."

Integration of Multimodal Information

In some applications, nodes in the knowledge graph may include multimodal information such as images, videos, or audio. By introducing multimodal encoders, this information can be converted into embeddings compatible with the generation model, enabling richer content generation.

3.3.3. Introduction of Enhanced Generation Techniques

To further improve the output quality in the generation phase, GraphRAG can leverage enhanced generation techniques to optimize the interaction between the query and graph data while enriching the generated content. These techniques primarily include the following:

Generation Optimization via Graph Attention Mechanism

By incorporating a Graph Attention Mechanism, the generation model can dynamically focus on key nodes and relationships within the retrieved subgraph, improving the relevance of the generated output. For example, in a complex medical question-answering task, the attention mechanism can guide the model to focus on the core connections between symptoms and treatment methods.

Generation-Guided Dynamic Graph Expansion

During the generation process, the model can dynamically issue supplementary queries to retrieve additional relevant graph data, enabling continuous content optimization. For instance, in technical document generation, the model can perform "supplementary retrieval" to obtain the latest industry standards or research advancements.

Enhanced Contrastive Learning and Content Calibration

Contrastive Learning allows the model to learn to distinguish between high-quality and low-quality generated content, thereby improving the reliability of the output. Additionally, a content calibration mechanism can be introduced to align the generated content with the structured data in the knowledge graph, ensuring accuracy and consistency.

Stylized and Personalized Generation

Depending on user needs and task contexts, the generated content can be stylistically adjusted. For example, in tasks aimed at general users, the model can adopt a simple and easy-to-understand language style, whereas in academic scenarios, it can generate more specialized and formal content.

4. Optimization

4.1. Query Expansion

Since queries are often short and lack sufficient information, **query expansion** aims to improve search results by supplementing or refining relevant terms or concepts in the original query. Luo et al. [19] enhanced retrieval queries by generating relation paths based on knowledge graphs (KGs). SPARQL[36] is used to retrieve all aliases of query entities from Wikidata, expanding retrieval queries to capture lexical variations of the same entity. Huang et al. (2023a)[37] proposed a consensus-view knowledge retrieval method, which improves retrieval performance and accuracy by discovering semantically related queries and reweighting original query terms. HyKGE[17] leverages large models to generate hypothetical outputs for the query and appends them to the original query as input for the retriever. Golden-Retriever (An et al., 2024)[2] identifies technical terms or jargon in the query and retrieves their explanations to supplement the query content.

4.2. Query Decomposition

Query Decomposition techniques aim to reduce the complexity and ambiguity of user queries by breaking them down into smaller, more specific subqueries. Each subquery typically focuses on a particular aspect or component of the original query, thereby capturing user intent more precisely and improving retrieval performance. For example, Complex-Logical-Reasoning[39] and Kg-gpt[40] proposed methods to decompose complex primary questions into multiple clauses, with each clause corresponding to an independent semantic relation. Relevant knowledge graph triples are then retrieved for each clause separately. This process not only enhances the specificity and efficiency of retrieval but also lays the foundation for subsequent knowledge integration and reasoning, demonstrating significant potential in handling complex question-answering tasks or multi-hop reasoning scenarios.

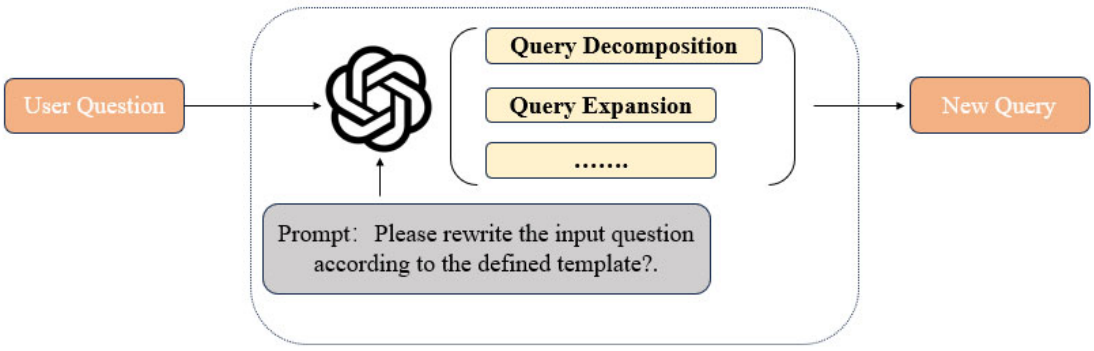

Figure 5. Query Rewrite

4.3. Re-ranking Module

Knowledge Pruning Techniques aim to refine retrieval results by filtering out irrelevant or redundant information, thereby enhancing system performance. Existing knowledge pruning methods can be broadly categorized into two types: re-ranking methods and large language model (LLM)-based methods. Re-ranking methods improve the relevance and quality of retrieval results by applying customized metrics to reorder or prioritize the retrieved information.

One common re-ranking approach involves using more powerful models to optimize the ranking of retrieval results. For instance, Li et al. (2023)[41] proposed a method that concatenates each retrieved triple with a question-option pair and uses a pre-trained cross-encoder to re-rank the triples, improving the accuracy of relevance evaluation. Another

method, HyKGE[17], employs FlagEmbedding for efficient text encoding, which reorders the top k documents returned by the embedding model "bge_reranker_large," significantly refining the retrieval results.

These re-ranking methods not only enhance the alignment between queries and retrieval results but also provide higher-quality input for complex reasoning tasks, such as knowledge graph-based question answering and multimodal retrieval. This highlights the growing importance of pruning techniques in future information retrieval and knowledge management systems, particularly as model capabilities continue to advance.

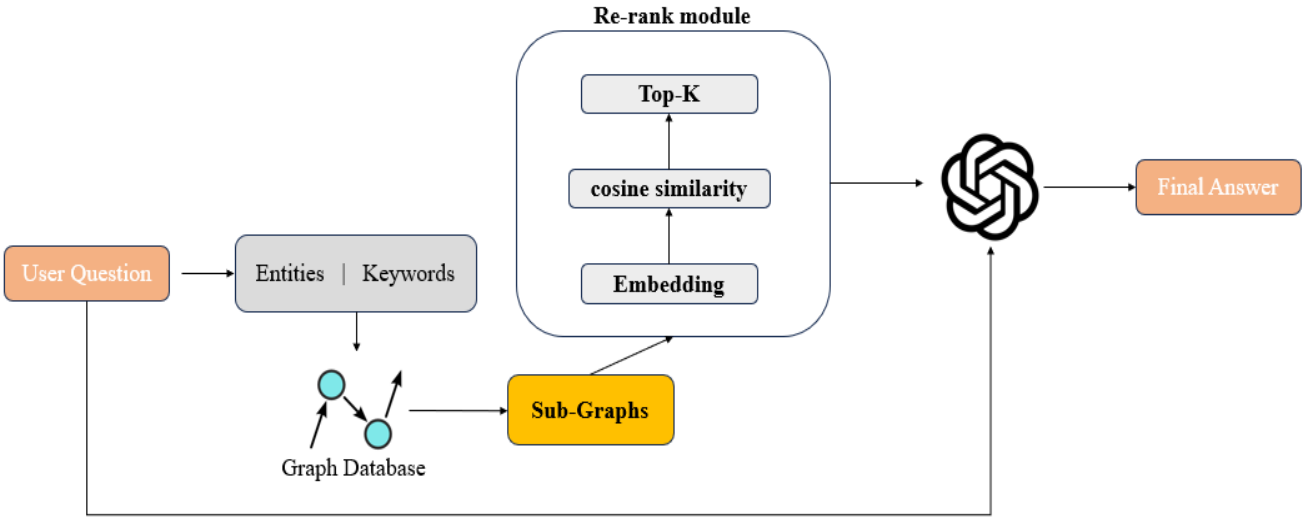

Figure 6. Framework of Re-rank

5. Future Prospects

In the future, constructing knowledge graphs that incorporate multimodal information will become a significant development direction. For instance, integrating information from text, images, and videos into a unified knowledge graph can help provide a more comprehensive description of entities and their relationships. This not only expands the application scenarios of knowledge graphs, such as visual question answering and video content analysis, but also provides diverse retrieval content for RAG models, thereby enhancing their reasoning capabilities. Achieving this goal requires research into designing effective multimodal feature representations, cross-modal alignment, and multimodal fusion strategies. Additionally, the construction of multimodal knowledge graphs must address technical challenges such as heterogeneous data processing and cross-modal association learning.

The static nature of knowledge graphs limits their applications in rapidly changing domains, such as news, social media analysis, and financial market forecasting. Therefore, enabling dynamic updates to knowledge graphs is key to enhancing the reasoning capabilities of large models. Future research should focus on developing efficient methods for dynamic knowledge graph updates, including automated data collection, knowledge extraction, and update mechanisms. At the same time, dynamic updates must balance data timeliness with the structural stability of the knowledge graph to ensure that newly added knowledge does not introduce redundancy or contradictions. Additionally, to support real-time reasoning, efficient knowledge retrieval and reasoning algorithms need to be designed to enable models to quickly adapt to updated knowledge. This necessitates the integration of stream data processing techniques, online learning algorithms, and optimization strategies for real-time reasoning in knowledge graph construction.

6. Conclusion

KGs-RAG combines graph-structured data with the RAG framework, providing an efficient and accurate solution for knowledge-intensive tasks. Compared to traditional retrieval methods, KGs-RAG leverages the rich semantic and relational information embedded in graph structures, enabling deeper query analysis and reasoning. This framework integrates the strengths of knowledge graphs, graph neural networks (GNNs), and large language models (LLMs), demonstrating strong adaptability and scalability.

In practical applications, KGs-RAG has achieved remarkable progress. For instance, graph-based query expansion supplements relevant semantic information to address issues of query brevity and ambiguity. Path retrieval algorithms identify the most relevant subgraphs or relational paths, laying a solid foundation for multi-hop reasoning. Knowledge pruning effectively filters out redundant or irrelevant information, significantly improving result accuracy. Additionally, generative reasoning leverages retrieved information to perform complex question-answering and knowledge generation tasks through powerful language generation capabilities. These methods work in tandem to greatly enhance the ability to acquire knowledge and solve complex problems.

Despite its significant advancements, KGs-RAG faces several challenges and directions for future research. First, improving the efficiency and robustness of retrieval and generation processes, particularly when dealing with large-scale graph-structured data, remains a pressing issue. Second, the dynamic and uncertain nature of graph data imposes higher demands for real-time retrieval and generation, requiring more adaptive models. Additionally, balancing the roles of retrieval, pruning, and generation across different tasks to optimize overall performance warrants further exploration. Lastly, the application of KGs-RAG in domain-

specific tasks, such as medical diagnosis, legal reasoning, and financial analysis, requires further research to better address diverse real-world needs.

References

[1] Achiam J, Adler S, Agarwal S, et al. Gpt-4 technical report[J]. arXiv preprint arXiv:2303.08774, 2023.

[2] Dubey A, Jauhri A, Pandey A, et al. The llama 3 herd of models[J]. arXiv preprint arXiv:2407.21783, 2024.

[3] Team G, Anil R, Borgeaud S, et al. Gemini: a family of highly capable multimodal models[J]. arXiv preprint arXiv:2312.11805, 2023.

[4] Kandpal N, Deng H, Roberts A, et al. Large language models struggle to learn long-tail knowledge[C]//International Conference on Machine Learning. PMLR, 2023: 15696-15707

[5] Zhang Y, Li Y, Cui L, et al. Siren's song in the AI ocean: a survey on hallucination in large language models[J]. arXiv preprint arXiv:2309.01219, 2023.

[6] Kandpal N, Deng H, Roberts A, et al. Large language models struggle to learn long-tail knowledge[C]//International Conference on Machine Learning. PMLR, 2023: 15696-15707.

[7] Cheng X, Luo D, Chen X, et al. Lift yourself up: Retrieval-augmented text generation with self-memory[J]. Advances in Neural Information Processing Systems, 2024, 36.

[8] Liu N F, Lin K, Hewitt J, et al. Lost in the middle: How language models use long contexts[J]. Transactions of the Association for Computational Linguistics, 2024, 12: 157-173.

[9] Singhal A. Introducing the knowledge graph: things, not strings[J]. Official google blog, 2012, 5(16): 3.

[10] Ji S, Pan S, Cambria E, et al. A survey on knowledge graphs: Representation, acquisition, and applications[J]. IEEE transactions on neural networks and learning systems, 2021, 33(2): 494-514.

[11] Tresp, V., Gärtner, T., & Nickel, M. (2021). Dynamic knowledge graphs: A survey. Artificial Intelligence Review, 55, 349–392.

[12] Chen D. Reading Wikipedia to answer open-domain questions[J]. arXiv preprint arXiv:1704.00051, 2017.

[13] Karpukhin V, Oğuz B, Min S, et al. Dense passage retrieval for open-domain question answering [J]. arXiv preprint arXiv: 2004.04906, 2020.

[14] Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks[J]. Advances in Neural Information Processing Systems, 2020, 33: 9459-9474.

[15] Gunawan D, Sembiring C A, Budiman M A. The implementation of cosine similarity to calculate text relevance between two documents[C]//Journal of physics: conference series. IOP Publishing, 2018, 978: 012120.

[16] Ramelan M. A Query Expansion Using Support Vector Machine (SVM) and Best Matching 25 (BM25)[J]. International Journal of Computer and Information System (IJCIS), 2024, 5(2): 73-77.

[17] Jiang X, Zhang R, Xu Y, et al. HyKGE: A Hypothesis Knowledge Graph Enhanced Framework for Accurate and Reliable Medical LLMs Responses[J]. arXiv preprint arXiv:2312.15883, 2024.

[18] Jin B, Xie C, Zhang J, et al. Graph chain-of-thought: Augmenting large language models by reasoning on graphs[J]. arXiv preprint arXiv:2404.07103, 2024.

[19] Luo L, Li Y F, Haffari G, et al. Reasoning on graphs: Faithful and interpretable large language model reasoning[J]. arXiv preprint arXiv:2310.01061, 2023.

[20] Ma S, Xu C, Jiang X, et al. Think-on-graph 2.0: Deep and interpretable large language model reasoning with knowledge graph-guided retrieval [J]. arXiv e-prints, 2024: arXiv: 2407.10805.

[21] Yang H, Liu J. Knowledge graph representation learning as groupoid: unifying TransE, RotatE, QuatE, ComplEx [C]// Proceedings of the 30th ACM international conference on information & knowledge management. 2021: 2311-2320.

[22] Yasunaga M, Ren H, Bosselut A, et al. QA-GNN: Reasoning with language models and knowledge graphs for question answering[J]. arXiv preprint arXiv:2104.06378, 2021.

[23] Taunk D, Khanna L, Kandru S V P K, et al. GrapeQA: Graph augmentation and pruning to enhance question-answering[C]//Companion Proceedings of the ACM Web Conference 2023. 2023: 1138-1144.

[24] He X, Tian Y, Sun Y, et al. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering[J]. arXiv preprint arXiv:2402.07630, 2024.

[25] Delile J, Mukherjee S, Van Pamel A, et al. Graph-Based Retriever Captures the Long Tail of Biomedical Knowledge[J]. arXiv preprint arXiv:2402.12352, 2024.

[26] Mavromatis C, Karypis G. GNN-RAG: Graph Neural Retrieval for Large Language Model Reasoning[J]. arXiv preprint arXiv:2405.20139, 2024.

[27] Zhang J, Zhang X, Yu J, et al. Subgraph retrieval enhanced model for multi-hop knowledge base question answering[J]. arXiv preprint arXiv:2202.13296, 2022.

[28] Liu Y. Roberta: A robustly optimized bert pretraining approach[J]. arXiv preprint arXiv:1907.11692, 2019, 364.

[29] Kim J, Kwon Y, Jo Y, et al. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models[J]. arXiv preprint arXiv:2310.11220, 2023.

[30] Wold S, Øvrelid L, Velldal E. Text-To-KG Alignment: Comparing Current Methods on Classification Tasks[J]. arXiv preprint arXiv:2306.02871, 2023.

[31] Jiang J, Zhou K, Dong Z, et al. Structgpt: A general framework for large language model to reason over structured data[J]. arXiv preprint arXiv:2305.09645, 2023.

[32] Liu G, Zhang Y, Li Y, et al. Explore then Determine: A GNN-LLM Synergy Framework for Reasoning over Knowledge Graph[J]. arXiv preprint arXiv:2406.01145, 2024.

[33] Kenton J D M W C, Toutanova L K. Bert: Pre-training of deep bidirectional transformers for language understanding [C] // Proceedings of naacL-HLT. 2019, 1: 2.

[34] Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer[J]. Journal of machine learning research, 2020, 21(140): 1-67.

[35] Driess D, Xia F, Sajjadi M S M, et al. Palm-e: An embodied multimodal language model [J]. arXiv preprint arXiv: 2303.03378, 2023.

[36] Cheng K, Lin G, Fei H, et al. Multi-hop question answering under temporal knowledge editing [J]. arXiv preprint arXiv: 2404.00492, 2024.

[37] Huang Y, Li Y, Xu Y, et al. Mvp-tuning: Multi-view knowledge retrieval with prompt tuning for commonsense reasoning[C]//Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2023: 13417-13432.

[38] An Z, Ding X, Fu Y C, et al. Golden-Retriever: High-Fidelity Agentic Retrieval Augmented Generation for Industrial Knowledge Base[J]. arXiv preprint arXiv:2408.00798, 2024.

[39] Choudhary N, Reddy C K. Complex logical reasoning over knowledge graphs using large language models[J]. arXiv preprint arXiv:2305.01157, 2023.

[40] Kim J, Kwon Y, Jo Y, et al. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models[J]. arXiv preprint arXiv:2310.11220, 2023.

[41] Li S, Gao Y, Jiang H, et al. Graph reasoning for question answering with triplet retrieval[J]. arXiv preprint arXiv: 2305.18742, 2023.