RESEARCH-ARTICLE

Graph Retrieval-Augmented Generation: A Survey

BOCI PENG, Peking University, Beijing, China

YUN ZHU, Zhejiang University, Hangzhou, Zhejiang, China

YONGCHAO LIU, Ant group, Hangzhou, Zhejiang, China

XIAOHE BO, Renmin University of China, Beijing, China

HAIZHOU SHI, Rutgers University–New Brunswick, New Brunswick, NJ, United States

CHUNTAO HONG, Ant group, Hangzhou, Zhejiang, China

View all

Published: 23 December 2025
Online AM: 19 November 2025
Accepted: 08 September 2025
Revised: 07 September 2025
Received: 11 January 2025

Graph Retrieval-Augmented Generation: A Survey

BOCI PENG, School of Intelligence Science and Technology, Peking University, Beijing, China and State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China
YUN ZHU, College of Computer Science and Technology, Zhejiang University, Hangzhou, China
YONGCHAO LIU, Ant Group, Hangzhou, China
XIAOHE BO, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
HAIZHOU SHI, Rutgers University, New Brunswick, New Jersey, USA
CHUNTAO HONG, Ant Group, Hangzhou, China
YAN ZHANG, School of Intelligence Science and Technology, Peking University, Beijing, China and State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China
SILIANG TANG, College of Computer Science and Technology, Zhejiang University, Hangzhou, China

Recently, Retrieval-Augmented Generation (RAG) has achieved remarkable success in addressing the challenges of Large Language Models (LLMs) without necessitating retraining. By referencing an external knowledge base, RAG refines LLM outputs, effectively mitigating issues such as "hallucination," lack of domain-specific knowledge, and outdated information. However, the complex structure of relationships among different entities in databases presents challenges for RAG systems. In response, GraphRAG leverages structural information across entities to enable more precise and comprehensive retrieval, capturing relational knowledge and facilitating more accurate, context-aware responses. Given the novelty and potential of GraphRAG, a systematic review of current technologies is imperative. This article provides the first comprehensive overview of GraphRAG methodologies. We formalize the GraphRAG workflow, encompassing Graph-Based Indexing, Graph-Guided Retrieval, and Graph-Enhanced Generation. We then outline the core technologies and training methods at each stage. Additionally, we examine downstream tasks, application domains, evaluation methodologies, and industrial use cases of GraphRAG. Finally, we explore future research directions to inspire further inquiries and advance progress in the field. In order to track recent progress, we set up a repository at https://github.com/pengboci/GraphRAG-Survey.

CCS Concepts: • **Computing methodologies** → **Knowledge representation and reasoning**; • **Information systems** → **Information retrieval**; *Data mining*;

Boci Peng and Yun Zhu contributed equally to this research.
Work done while Boci Peng and Yun Zhu were interns at Ant Group.
Authors' Contact Information: Boci Peng, School of Intelligence Science and Technology, Peking University, Beijing, China and State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China; e-mail: bcpeng@stu.pku.edu.cn; Yun Zhu, College of Computer Science and Technology, Zhejiang University, Hangzhou, China; e-mail: zhuyun_dcd@zju.edu.cn; Yongchao Liu (corresponding author), Ant Group, Hangzhou, China; e-mail: yongchao.ly@antgroup.com; Xiaohe Bo, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China; e-mail: xiaohe@ruc.edu.cn; Haizhou Shi, Rutgers University, New Brunswick, New Jersey, USA; e-mail: haizhou.shi@rutgers.edu; Chuntao Hong, Ant Group, Hangzhou, China; e-mail: chuntao.hct@antgroup.com; Yan Zhang (corresponding author), School of Intelligence Science and Technology, Peking University, Beijing, China and State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China; e-mail: zhyzhy001@pku.edu.cn; Siliang Tang, College of Computer Science and Technology, Zhejiang University, Hangzhou, China; e-mail: siliang@zju.edu.cn.

ACM 1558-2868/2025/12-ART35
https://doi.org/10.1145/3777378

Additional Key Words and Phrases: Large Language Models, Graph Retrieval-Augmented Generation, Knowledge Graphs, Graph Neural Networks

ACM Reference format:
Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2025. Graph Retrieval-Augmented Generation: A Survey. *ACM Trans. Inf. Syst.* 44, 2, Article 35 (December 2025), 52 pages.
https://doi.org/10.1145/3777378

1 Introduction

The development of **Large Language Models (LLMs)** like GPT [173], Qwen [251, 252], LLaMA [44], and DeepSeek [33–35, 200] has sparked a revolution in the field of AI, fundamentally altering the landscape of **Natural Language Processing (NLP)**. These models, built on Transformer [220] architectures and trained on diverse and extensive datasets, have demonstrated unprecedented capabilities in understanding, interpreting, and generating human language. The impact of these advancements is profound, stretching across various sectors including healthcare [140, 225, 274], finance [124, 169], and education [60, 228], where they facilitate more nuanced and efficient interactions between humans and machines.

Despite their remarkable language comprehension and text generation capabilities, LLMs may exhibit limitations due to a lack of domain-specific knowledge, real-time updated information, and proprietary knowledge, which are outside LLMs' pre-training corpus. These gaps can lead to a phenomenon known as "hallucination" [84] where the model generates inaccurate or even fabricated information. Consequently, it is imperative to supplement LLMs with external knowledge to mitigate this problem. **Retrieval-Augmented Generation (RAG)** [48, 59, 81, 86, 239, 262, 273] emerged as a significant evolution, which aims to enhance the quality and relevance of generated content by integrating a retrieval component within the generation process. The essence of RAG lies in its ability to dynamically query a large text corpus to incorporate relevant factual knowledge into the responses generated by the underlying **Language Models (LMs)**. This integration not only enriches the contextual depth of the responses but also ensures a higher degree of factual accuracy and specificity. RAG has gained widespread attention due to its exceptional performance and broad applications, becoming a key focus within the field.

Although RAG has achieved impressive results and has been widely applied across various domains, it faces limitations in real-world scenarios: (1) *Neglecting Relationships*: In practice, textual content is not isolated but interconnected. Traditional RAG fails to capture significant structured relational knowledge that cannot be represented through semantic similarity alone. For instance, in a citation network where papers are linked by citation relationships, traditional RAG methods focus on finding the relevant papers based on the query but overlook important citation relationships between papers. (2) *Redundant Information*: RAG often recounts content in the form of textual snippets when concatenated as prompts. This makes context become excessively lengthy, leading to the "lost in the middle" dilemma [141]. (3) *Lacking Global Information*: RAG can only retrieve a subset of documents and fails to grasp global information comprehensively and hence struggles with tasks such as **Query-Focused Summarization (QFS)** [45].

Graph Retrieval-Augmented Generation (GraphRAG) [45, 80, 161] emerges as an innovative solution to address these challenges. Unlike traditional RAG, GraphRAG retrieves graph elements containing relational knowledge pertinent to a given query from a pre-constructed graph database, as depicted in Figure 1. These elements may include nodes, triples, paths, or subgraphs, which are utilized to generate responses. GraphRAG considers the interconnections between texts, enabling a

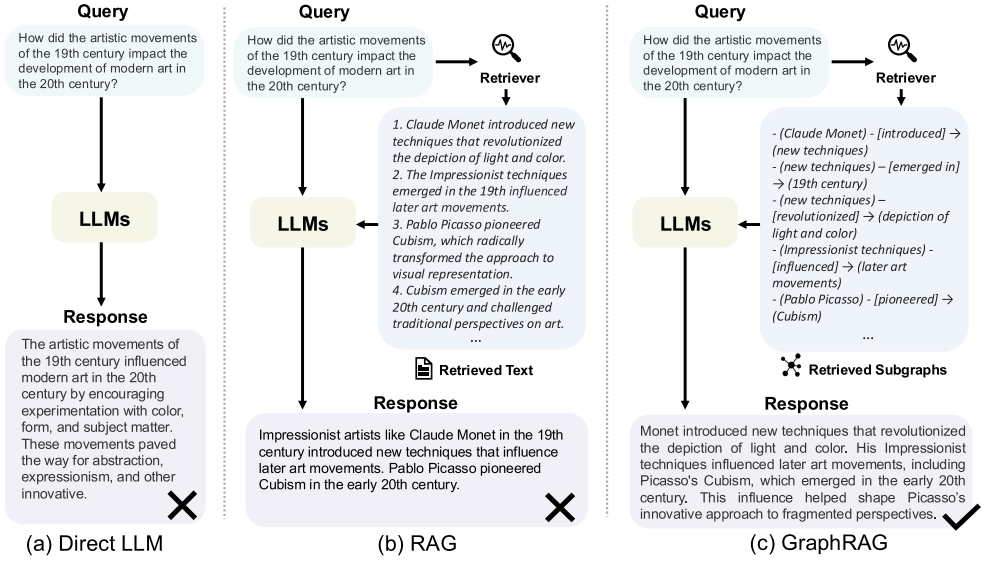

Fig. 1. Comparisons between (a) Direct LLM, (b) RAG, and (c) GraphRAG. Given a user query, direct answering by LLMs may suffer from shallow responses or a lack of specificity. RAG addresses this by retrieving relevant textual information, somewhat alleviating the issue. However, due to the text's length and flexible natural language expressions of entity relationships, RAG struggles to emphasize "influence" relations, which is the core of the question. While GraphRAG methods leverage explicit entity and relationship representations in graph data, enabling precise answers by retrieving relevant structured information.

more accurate and comprehensive retrieval of relational information. Additionally, graph data, such as **Knowledge Graphs (KGs)**, offer abstraction and summarization of textual data, thereby significantly shortening the length of the input text and mitigating concerns of verbosity. By retrieving subgraphs or graph communities, we can access comprehensive information to effectively address the QFS challenge by capturing the broader context and interconnections within the graph structure.

In this article, we are the first to provide a systematic survey of GraphRAG. Specifically, we begin by introducing the GraphRAG workflow, along with the foundational background knowledge that underpins the field. Then, we categorize the literature according to the primary stages of the GraphRAG process: **Graph-Based Indexing (G-Indexing)**, **Graph-Guided Retrieval (G-Retrieval)**, and **Graph-Enhanced Generation (G-Generation)** in Sections 5–7, respectively, detailing the core technologies and training methods within each phase. Furthermore, we investigate downstream tasks, application domains, evaluation methodologies, and industrial use cases of GraphRAG. This exploration elucidates how GraphRAG is being utilized in practical settings and reflects its versatility and adaptability across various sectors. Finally, acknowledging that research in GraphRAG is still in its early stages, we delve into potential future research directions. This prognostic discussion aims to pave the way for forthcoming studies, inspire new lines of inquiry, and catalyze progress within the field, ultimately propelling GraphRAG toward more mature and innovative horizons.

Our contributions can be summarized as follows:

—We provide a comprehensive and systematic review of existing state-of-the-art GraphRAG methodologies. We offer a formal definition of GraphRAG, outlining its universal workflow which includes G-Indexing, G-Retrieval, and G-Generation.

—We discuss the core technologies underpinning existing GraphRAG systems, including G-Indexing, G-Retrieval, and G-Generation. For each component, we analyze the spectrum of model selection, methodological design, and enhancement strategies currently being explored. Additionally, we contrast the diverse training methodologies employed across these modules.
—We delineate the downstream tasks, benchmarks, application domains, evaluation metrics, current challenges, and future research directions pertinent to GraphRAG, discussing both the progress and prospects of this field. Furthermore, we compile an inventory of existing industry GraphRAG systems, providing insights into the translation of academic research into real-world industry solutions.

The rest of the survey is structured as follows: Section 2 compares related techniques, while Section 3 outlines the general process of GraphRAG. Sections 5–7 categorize the techniques associated with GraphRAG's three stages: G-Indexing, G-Retrieval, and G-Generation. Section 8 introduces the training strategies of retrievers and generators. Section 9 summarizes GraphRAG's downstream tasks, corresponding benchmarks, application domains, evaluation metrics, and industrial GraphRAG systems. Section 10 provides an outlook on future directions. Finally, Section 11 concludes the content of this survey.

2 Comparison with Related Techniques and Surveys

In this section, we compare GraphRAG with related techniques and corresponding surveys, including RAG, LLMs on graphs, and **Knowledge Base Question Answering (KBQA)**.

2.1 RAG

RAG combines external knowledge with LLMs for improved task performance, integrating domain-specific information to ensure factuality and credibility. In the past 2 years, researchers have written many comprehensive surveys about RAG [48, 59, 81, 86, 239, 262, 273]. For example, Fan et al. [48] and Gao et al. [59] categorize RAG methods from the perspectives of retrieval, generation, and augmentation. Zhao et al. [273] review RAG methods for databases with different modalities. Yu et al. [262] systematically summarize the evaluation of RAG methods. These works provide a structured synthesis of current RAG methodologies, fostering a deeper understanding and suggesting future directions in the area. Although previous surveys on RAG have touched upon GraphRAG, they predominantly center on textual data integration. This article diverges by placing a primary emphasis on the indexing, retrieval, and utilization of structured graph data, which represents a substantial departure from handling purely textual information and spurs the emergence of many new techniques.

From a broad perspective, GraphRAG can be seen as a branch of RAG. However, as shown in Figure 2, GraphRAG differs from text-based RAG in the following key aspects: in terms of indexing, text-based RAG directly vectorizes text chunks, whereas GraphRAG first decomposes raw text data into a graph (e.g., KG) before building the index. This process typically involves information filtering and summarization to enhance the refinement of information within the graph, which simultaneously establishes associations between different chunks in the form of edges. For retrieval, text-based RAG employs text retrieval algorithms to fetch relevant chunks, while GraphRAG relies on graph search algorithms [67, 68, 76, 212] to retrieve information from the graph. Graph information primarily consists of entities, relationships, and concise textual elements, which to some extent mitigates the issue of information redundancy inherent in RAG approaches. Regarding generation, text-based RAG can seamlessly incorporate retrieved chunks into prompts, whereas GraphRAG requires an additional step of converting graph-structured data into a format that LMs can process.

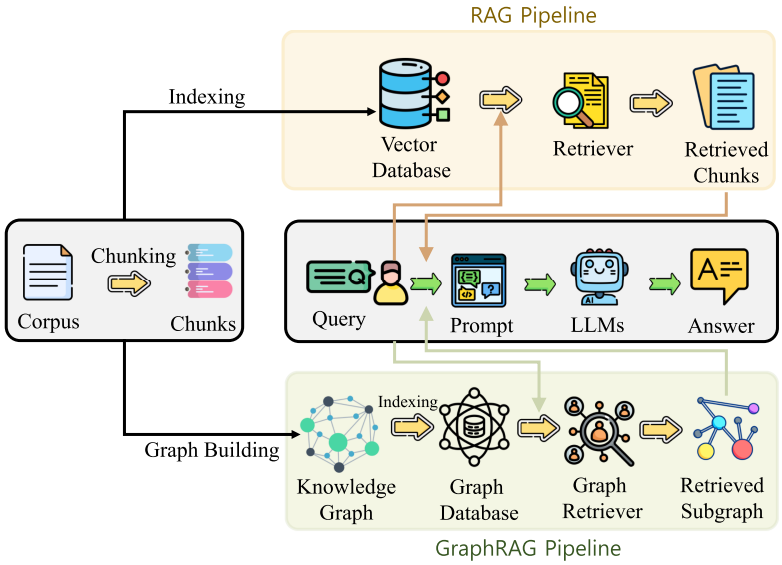

Fig. 2. Comparisons between RAG and GraphRAG.

Vanilla RAG suffers from inherent shortcomings in multi-hop complex reasoning, primarily due to its inability to effectively model intricate relationships between entities or across documents. In contrast, GraphRAG achieves significant performance improvements by introducing graph structures, with its advantage lying in the explicit modeling of multi-hop semantic relationships between entities, thereby systematically enhancing reasoning capabilities [70, 276]. Specifically, the graph architecture of GraphRAG enables multi-hop reasoning, that is, the ability to comprehend and answer complex queries by traversing multiple nodes in the graph. This process does not rely solely on isolated chunks or simple semantic similarity but instead synthesizes information from multiple sources through edges (relationships) in the graph, yielding more accurate and comprehensive answers. For instance, when handling multi-hop queries, GraphRAG can identify relevant subgraphs or paths within the graph, where edges connect these paths. Each edge represents a relationship between different entities, thereby providing high-order information to LMs [98, 153, 155, 212, 243].

Additionally, GraphRAG effectively addresses the limitations of vanilla RAG in global summarization tasks (e.g., QFS) by leveraging the inherent advantages of graph structures. Its core innovation lies in utilizing graph community detection to automatically identify and aggregate semantically related entity clusters [45, 79]. Through joint abstraction of these interconnected entities, GraphRAG generates more comprehensive, semantically coherent, and thematically focused summaries. In contrast, vanilla RAG approaches, constrained by their isolated chunk retrieval mechanism, can only produce separate summaries of discrete text chunks, resulting in summaries that lack a global perspective.

2.2 LLMs on Graphs

LLMs have demonstrated exceptional capabilities in natural language understanding, reasoning, and generation. However, their inherent design for processing sequential text data limits their ability to handle non-Euclidean graph structures directly [63, 224]. To bridge this gap, numerous studies [20, 49, 99, 123, 139, 158, 177, 178] have explored methods to integrate LLMs with graph-structured

data. These approaches can be broadly categorized into three main paradigms [26, 123] based on the role assigned to the LLM:

LLM as Enhancer. In this paradigm, LLMs are used to augment the textual attributes of nodes or edges in a graph. The enriched graph is then processed by **Graph Neural Networks (GNNs)** for downstream tasks. For example, LLMs generate detailed node/edge descriptions [26, 74] or embeddings [43, 135, 244], which GNNs use for tasks such as node classification or link prediction. This approach leverages LLMs' semantic understanding to improve feature quality but still relies on GNNs for structural reasoning.

LLM as Predictor. This approach positions the LLM as the central reasoning engine for end-to-end predictions on graph tasks. Typical methods include flattening graph data into prompts [63, 224, 272], or incorporating GNNs as auxiliary components to provide graph representations by leveraging their powerful graph modeling capabilities for LLMs [16, 216, 232, 278]. While this fully utilizes the LLM's reasoning capabilities, its main challenge lies in the potential loss of complex topological information during the serialization process.

GNN-LLM Alignment. These methods employ a dual-module architecture where GNNs (for structure) and LLMs (for text) operate in parallel. Their respective representations are then aligned and fused in a shared semantic space, often through techniques like contrastive learning or attention mechanisms [12, 46, 236, 277]. This offers a powerful synthesis of structural and semantic information, with the core technical challenge being the effective alignment of representations from these two distinct modalities.

It is crucial to distinguish the LLMs on graphs paradigm from GraphRAG. The former uses LLMs to enhance performance on graph-centric tasks (e.g., node classification). In contrast, GraphRAG, the focus of this survey, adopts the reverse approach: it retrieves relevant graph structures from external knowledge bases to enhance an LLM's performance on natural language tasks, particularly in complex question-answering and reasoning scenarios. This survey provides a detailed introduction to the technologies and applications specific to GraphRAG, a topic not covered in-depth by previous surveys on LLMs on graphs.

2.3 KBQA

KBQA is a significant task in NLP, aiming to respond to user queries based on external knowledge bases [55, 107, 108, 255], thereby achieving goals such as fact verification, passage retrieval enhancement, and text understanding. Previous surveys typically categorize existing KBQA approaches into two main types: **Information Retrieval (IR)**-based methods and **Semantic Parsing (SP)**-based methods. Specifically, IR-based methods [94, 95, 153, 155, 212, 226, 243, 265] retrieve information related to the query from the KG and use it to enhance the generation process. While SP-based methods [17, 24, 50, 62, 210, 258] generate a **Logical Form (LF)** for each query and execute it against knowledge bases to obtain the answer.

GraphRAG and KBQA are closely related, with IR-based KBQA methods representing a subset of GraphRAG approaches focused on downstream applications. These two approaches exhibit fundamental technical differences: while KBQA operates exclusively on KGs, GraphRAG treats graph construction as a configurable component capable of generating various graph types (including KGs [153, 212] or text-augmented variants [45, 66]) adapted to specific application scenarios; in terms of retrieved information, KBQA is limited to relationship-centric elements like triplets [120] and paths [98], whereas GraphRAG extends to richer textual content such as entity/relationship descriptions [66, 247], graph community summaries [45, 79], and even raw text chunks [25, 68, 247]; furthermore, GraphRAG requires more sophisticated prompt engineering than KBQA to

properly integrate heterogeneous information types (structural relationships, short-text attributes, and long-form descriptions) during answer generation [63]. In this work, we extend the discussion beyond KBQA to include GraphRAG's applications across various downstream tasks. Our survey provides a thorough and detailed exploration of GraphRAG technology, offering a comprehensive understanding of existing methods and potential improvements.

3 Preliminaries

In this section, we introduce background knowledge of GraphRAG for easier comprehension of our survey. First, we introduce **Text-Attributed Graphs (TAGs)** which is a universal and general format of graph data used in GraphRAG. Then, we provide formal definitions for two types of models that can be used in the retrieval and generation stages: GNNs and LMs.

3.1 TAGs

The graph data used in GraphRAG can be represented uniformly as TAGs, where nodes and edges possess textual attributes. For example, Sun et al. [212], Ma et al. [155], and Luo et al. [153] use a KG as the TAG, where nodes are entities, edges are relations between entities, and text attributes are the names of entities and relations. While Guo et al. [66] and Xu et al. [247] additionally extracts detailed textual descriptions of nodes and edges as supplementary attribute information. Formally, a TAG can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \{\mathbf{x}_v\}_{v\in\mathcal{V}}, \{\mathbf{e}_{i,j}\}_{i,j\in\mathcal{E}})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, $\mathcal{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the adjacent matrix. Additionally, $\{\mathbf{x}_v\}_{v\in\mathcal{V}}$ and $\{\mathbf{e}_{i,j}\}_{i,j\in\mathcal{E}}$ are textual attributes of nodes and edges, respectively.

3.2 GNNs

GNNs are a kind of deep-learning framework to model the graph data. Classical GNNs, e.g., GCN [106], GAT [221], GraphSAGE [69], adopt a message-passing manner to obtain node representations. Formally, each node representation $\mathbf{h}_i^{(l-1)}$ in the lth layer is updated by aggregating the information from neighboring nodes and edges:

$$\mathbf{h}_i^{(l)} = \mathbf{UPD}(\mathbf{h}_i^{(l-1)}, \mathbf{AGG}_{j\in\mathcal{N}(i)}\mathbf{MSG}(\mathbf{h}_i^{(l-1)}, \mathbf{h}_j^{(l-1)}, \mathbf{e}_{i,j}^{(l-1)})), \tag{1}$$

where $\mathcal{N}_{(i)}$ represents the neighbors of node i. **MSG** denotes the message function, which computes the message based on the node, its neighbor, and the edge between them. **AGG** refers to the aggregation function that combines the received messages using a permutation-invariant method, such as mean, sum, or max. **UPD** represents the update function, which updates each node's attributes with the aggregated messages.

Subsequently, a readout function, e.g., mean, sum, or max pooling, can be applied to obtain the global-level representation:

$$\mathbf{h}_G = \mathbf{READOUT}_{i\in\mathcal{V}_G}(\mathbf{h}_i^{(L)}). \tag{2}$$

In GraphRAG, GNNs can be utilized to obtain representations of graph data for the retrieval phase, as well as to model the retrieved graph structures.

3.3 LMs

LMs excel in language understanding and are mainly classified into two types: discriminative and generative. Discriminative models, like BERT [40], RoBERTa [147] and SentenceBERT [190], focus on estimating the conditional probability $P(\mathbf{y}|\mathbf{x})$ and are effective in tasks such as text classification and sentiment analysis. In contrast, generative models, including GPT-3 [13] and GPT-4 [173], aim to model the joint probability $P(\mathbf{x}, \mathbf{y})$ for tasks like machine translation and text generation. These generative pre-trained models have significantly advanced the field of NLP by leveraging massive

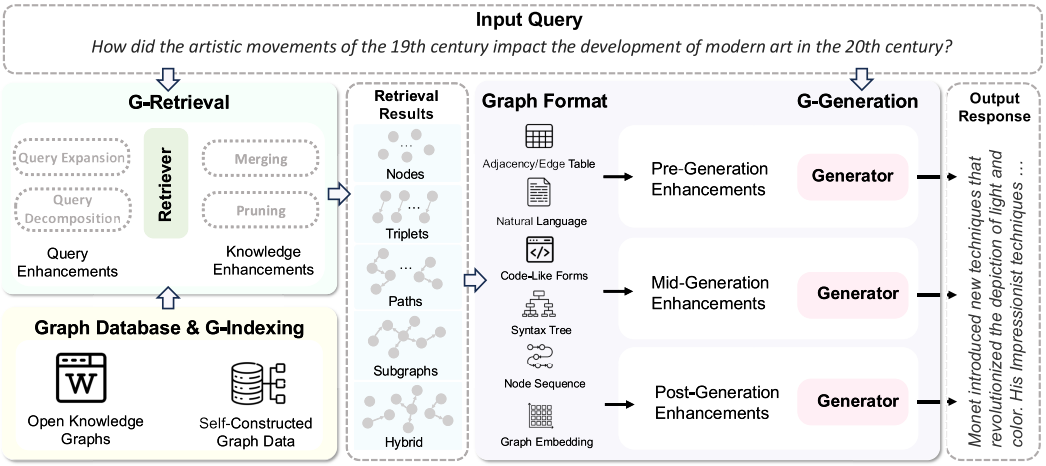

Fig. 3. The overview of the GraphRAG framework for question-answering task. In this survey, we divide GraphRAG into three stages: G-Indexing, G-Retrieval, and G-Generation. We categorize the retrieval sources into open source KGs and self-constructed graph data. Various enhancing techniques like query enhancement and knowledge enhancement may be adopted to boost the relevance of the results. Unlike RAG, which uses retrieved text directly for generation, GraphRAG requires converting the retrieved graph information into patterns acceptable to generators to enhance the task performance.

datasets and billions of parameters, contributing to the rise of LLMs with outstanding performance across various tasks.

In the early stages, RAG and GraphRAG focused on improving pre-training techniques for discriminative LMs [40, 147, 190]. Recently, LLMs such as ChatGPT [174], LLaMA [44], and Qwen2 [252] have shown great potential in language understanding, demonstrating powerful in-context learning capabilities. Subsequently, research on RAG and GraphRAG shifted toward enhancing IR for LMs, addressing increasingly complex tasks and mitigating hallucinations, thereby driving rapid advancements in the field.

4 Overview of GraphRAG

GraphRAG is a framework that leverages external structured KGs to improve contextual understanding of LMs and generate more informed responses, as depicted in Figure 3. The goal of GraphRAG is to retrieve the most relevant graph data from graph databases, thereby enhancing the answers to downstream tasks. The process can be defined as

$$a^* = \arg\max_{a \in A} p(a|q, \mathcal{G}), \tag{3}$$

where a^* is the optimal answer of the query q given the TAG $\mathcal{G}$, and A is the set of possible responses. After that, we jointly model the target distribution $p(a|q, \mathcal{G})$ with a graph retriever $p_\theta(G|q, \mathcal{G})$ and an answer generator $p_\phi(a|q, G)$ where θ, ϕ are learnable parameters, and utilize the total probability formula to decompose $p(a|q, \mathcal{G})$, which can be formulated as

$$\begin{aligned} p(a|q, \mathcal{G}) &= \sum_{G \subseteq \mathcal{G}} p_\phi(a|q, G) p_\theta(G|q, \mathcal{G}) \\ &\approx p_\phi(a|q, G^*) p_\theta(G^*|q, \mathcal{G}), \end{aligned} \tag{4}$$

where G^* is the optimal subgraph. The first line of Equation (4) is approximated by the second line. This approximation is necessary because the summation over all possible subgraphs $G \subseteq \mathcal{G}$

in the first line is computationally intractable due to the exponential growth of the search space. Following common practice in RAG [111], we approximate marginalization by conditioning on the most relevant subgraph G^*, assuming that the majority of the probability mass is concentrated on the optimal subgraph. Specifically, a graph retriever is employed to extract the optimal subgraph G^*, after which the generator produces the answer based on the retrieved subgraph.

Therefore, in this survey, we decompose the entire GraphRAG process into three main stages: G-Indexing, G-Retrieval, and G-Generation. The overall workflow of GraphRAG is illustrated in Figure 3.

G-Indexing. G-Indexing constitutes the initial phase of GraphRAG, aimed at identifying or constructing a graph database $\mathcal{G}$ that aligns with downstream tasks and establishing indices on it. The graph database can originate from public KGs [4, 10, 136, 193, 207, 222], graph data [165], or be constructed based on proprietary data sources such as textual [45, 67, 116, 231] or other forms of data [249]. The indexing process typically includes mapping node and edge properties, establishing pointers between connected nodes, and organizing data to support fast traversal and retrieval operations. Indexing determines the granularity of the subsequent retrieval stage, playing a crucial role in enhancing query efficiency.

G-Retrieval. Following G-Indexing, the G-Retrieval phase focuses on extracting pertinent information from the graph database in response to user queries or input. Specifically, given a user query q which is expressed in natural language, the retrieval stage aims to extract the most relevant elements (e.g., entities, triplets, paths, subgraphs) from KGs, which can be formulated as

$$\begin{aligned} G^* &= \textbf{G-Retriever}(q, \mathcal{G}) \\ &= \arg\max_{G \subseteq \mathcal{R}(\mathcal{G})} p_\theta(G|q, \mathcal{G}) \\ &= \arg\max_{G \subseteq \mathcal{R}(\mathcal{G})} \textbf{Sim}_\theta(q, G), \end{aligned} \tag{5}$$

where G^* is the optimal retrieved graph elements and $\textbf{Sim}_\theta(\cdot, \cdot)$ is a function that measures the semantic similarity between user queries and the graph data. $\mathcal{R}(\cdot)$ represents adopting specific retrieval paradigms to narrow the search scope of subgraphs, thereby improving retrieval efficiency.

G-Generation. The G-Generation phase involves synthesizing meaningful outputs or responses based on the retrieved graph data. This could encompass answering user queries, generating reports, and so on. In this stage, a generator takes the query, retrieved graph elements, and an optional prompt as input to generate a response, which can be denoted as

$$\begin{aligned} a^* &= \textbf{G-Generator}(q, G^*) \\ &= \arg\max_{a \in A} p_\phi(a|q, G^*) \\ &= \arg\max_{a \in A} p_\phi(a|\mathcal{F}(q, G^*)), \end{aligned} \tag{6}$$

where $\mathcal{F}(\cdot, \cdot)$ is a function that converts graph data into a form the generator can process, typically non-parametric rules or algorithms.

In the following sections, we provide detailed introductions of each stage and summarize the methods employed by existing research at each stage, as shown in Table 1.

5 G-Indexing

GraphRAG employs G-Indexing to systematically organize and store information, leveraging the inherent relationships and structure of graph architectures as the foundation for efficient data

Table 1. Summary of Existing Studies on GraphRAG

Model	Graph Data	Indexing	Retriever	Retrieval Paradigm	Retrieval Granularity	Query Enhancement	Knowledge Enhancement	Generator	Graph Format	Generation Enhancement
GRAFT-Net [209]	Self-constructed	Graph, Text	Non-Parametric	Once	Nodes	-	-	GNNs	Embeddings	Pre
KagNet [130]	Open	Graph	Non-Parametric	Once	Subgraphs	-	Pruning	Hybrid	Embeddings	Post
PullNet [208]	Open	Graph, Text	GNN-Based	Iterative	Subgraphs	-	-	GNNs	Embeddings	Pre
QA-GNN [256]	Open	Graph	Non-Parametric	Once	Subgraphs	-	Pruning	Hybrid	Embeddings	Pre
SR [265]	Open	Graph	LM-Based	Iterative	Subgraphs	-	Merging	-	-	-
G-G-E [57]	Open	Graph	Non-Parametric	Once	Subgraphs	-	Merging, Pruning	GNNs	Embeddings	-
Greaselm [270]	Open	Graph	Non-Parametric	Once	Subgraphs	-	-	Hybrid	Embeddings	Pre, Post
SAFE [93]	Open	Graph	Non-Parametric	Once	Subgraphs	-	-	Hybrid	Embeddings	Post
UniKGQA [95]	Open	Graph	LM-Based	Multi-stage	Subgraphs	-	-	LMs	Embeddings	-
R-R-A [243]	Open	Graph, Text	LM-Based	Iterative	Subgraphs	-	Pruning	LMs	Languages	Pre
HyKGE [98]	Open	Graph	Non-Parametric	Once	Paths	Expansion	Pruning	LMs	Languages	-
KN [65]	Open	Graph, Text	LM-Based	Iterative	Subgraphs	-	Merging, Pruning	LMs	Languages	-
KG-GPT [104]	Open	Graph	LM-Based	Once	Subgraphs	Decomposition	-	LMs	Languages	-
GrapeQA [217]	Open	Graph	Non-Parametric	Once	Subgraphs	-	Pruning	Hybrid	Embeddings	Pre
MVP-Tuning [87]	Open	Text	Non-Parametric	Once	Triplets	-	-	LMs	Languages	Pre
StructGPT [92]	Open	Graph	LM-Based	Iterative	Hybrid	-	-	LMs	Languages	-
RASR [150]	Open	Graph	Non-Parametric	Once	Subgraphs	-	Pruning	GNNs	Embeddings	-
KAPING [6]	Open	Text	LM-Based	Once	Triplets	-	Pruning	LMs	Languages	-
OpenCSR [71]	Self-constructed	Graph	GNN-Based	Iterative	Subgraphs	-	Pruning	GNNs	Embeddings	-
SKP [41]	Open	Text	Non-Parametric	Once	Subgraphs	-	-	LMs	Embeddings	-
KnowledGPT [229]	Open	Graph, Text	LM-Based	Iterative	Hybrid	-	-	LMs	Languages	-
LARK [29]	Open	Graph	Non-Parametric	Once	Subgraphs	Decomposition	-	LMs	Languages	-
KALMV [7]	Open	Graph	Non-Parametric	Once	Triplets	-	-	LMs	Languages	Post
HamQA [42]	Open	Graph	Non-Parametric	Once	Subgraphs	-	-	GNNs	Embeddings	-
DecAF [260]	Open	Text	LM-Based	Once	Subgraphs	-	-	LMs	Embeddings	-
G-Retriever [76]	Open	Graph, Vector	Hybrid	Multi-stage	Subgraphs	-	-	Hybrid	Embeddings	-
GraphRAG [45]	Self-constructed	Graph, Text	LM-Based	Once	Subgraphs	-	-	LMs	Languages	-
RoK [230]	Open	Graph	Non-Parametric	Multi-stage	Subgraphs	Expansion	Pruning	LMs	Languages	-
Graph CoT [100]	Open	Graph	LM-Based	Adaptive	Hybrid	-	-	LMs	Languages	-
RoG [153]	Open	Graph	Hybrid	Once	Paths	Expansion	-	LMs	Languages	-
ToG [212]	Open	Graph	LM-Based	Iterative	Subgraphs	-	-	LMs	Languages	-
TEMPLE-MQA [27]	Self-constructed	Graph	Non-Parametric	Iterative	Subgraphs	Expansion	Pruning	LMs	Languages	-
KG-Agent [94]	Open	Graph	LM-Based	Iterative	Hybrid	-	-	LMs	Languages	-
RA-Sim [183]	Self-constructed	Graph	Non-Parametric	Once	Subgraphs	-	-	GNNs	Embeddings	-
GenTKGQA [58]	Open	Graph	Non-Parametric	Once	Triplets	Expansion	-	Hybrid	Embeddings	-
HippoRAG [67]	Self-constructed	Graph	Non-Parametric	Once	Nodes	-	-	LMs	Languages	-
HippoRAG2 [68]	Self-constructed	Graph, Text, Vector	Non-Parametric	Once	Nodes	-	-	LMs	Languages	-
MindMap [235]	Open	Graph	Non-Parametric	Once	Subgraphs	-	Merging	LMs	Languages	Mid
DALK [116]	Self-constructed	Graph	Non-Parametric	Once	Subgraphs	-	Merging, Pruning	LMs	Languages	Pre
KGP [231]	Self-constructed	Graph	Non-Parametric	Iterative	Subgraphs	Expansion	-	LMs	Languages	-
ODA [213]	Open	Graph, Text	LM-Based	Iterative	Hybrid	-	-	LMs	Languages	-
GNN-RAG [161]	Open	Graph	Hybrid	Multi-stage	Paths	-	-	LMs	Languages	-
GRAG [80]	Open	Graph, Vector	LM-Based	Once	Subgraphs	-	Pruning	Hybrid	Embeddings	-
EtD [134]	Open	Graph, Vector	Hybrid	Iterative	Nodes	-	-	LMs	Languages	Pre
EWEK-QA [36]	Open	Graph, Text	LM-Based	Iterative	Subgraphs	-	-	LMs	Languages	-
KELP [137]	Open	Graph	Non-Parametric	Once	Paths	-	Pruning	LMs	Languages	-
ToG 2.0 [155]	Open	Graph, Text	LM-Based	Iterative	Subgraphs	Expansion	Pruning	LMs	Languages	-
REANO [51]	Self-constructed	Graph, Text	GNN-Based	Once	Triplets	-	-	LMs	Embeddings	-
SubgraphRAG [120]	Open	Graph, Text	Hybrid	Once	Triplets	-	-	LMs	Languages	-
LightRAG [66]	Self-constructed	Graph, Text, Vector	LM-Based	Once	Nodes	Decomposition	-	LMs	Languages	-
NodeRAG [247]	Self-constructed	Graph, Text, Vector	Non-Parametric	Once	Nodes	-	-	LMs	Languages	-

retrieval. In this section, we categorize and summarize the graph data and various indexing methods that have been employed. The overview of G-Indexing is illustrated in Figure 4.

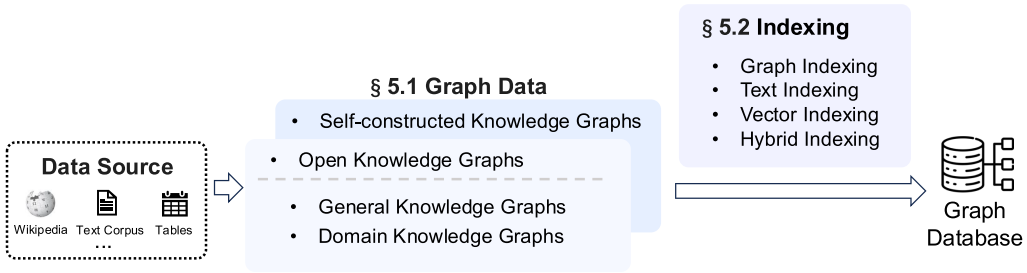

Fig. 4. The overview of G-Indexing.

5.1 Graph Data

Various types of graph data are utilized in GraphRAG for retrieval and generation. Here, we categorize these data into two categories based on their sources, including open KGs and self-constructed graph data. The former mainly applies to scenarios using pre-defined KGs or open-domain QA, while the latter specifically targets retrieval and question answering from unstructured text or tabular data sources.

5.1.1 Open KGs. Open KGs refer to graph data sourced from publicly available repositories or databases [4, 10, 207, 222]. Using these KGs could dramatically reduce the time and resources required to develop and maintain. In this survey, we further classify them into two categories according to their scopes, i.e., General KGs and Domain KGs. The former primarily encompasses general-domain knowledge (e.g., encyclopedic and commonsense knowledge), while the latter focuses on specific domains (such as e-commerce, biomedical fields, etc.).

(1) General KGs. General KGs primarily store general, structured knowledge and typically rely on collective input and updates from a global community, ensuring a comprehensive and continually refreshed repository of information. Encyclopedic KGs and commonsense KGs are two representative types of general KGs.

Encyclopedic KGs contain large-scale real-world knowledge collected from human experts and encyclopedias. For example, Wikidata[1] [222] is a free and open knowledge base that stores structured data of its Wikipedia sister projects like Wikipedia, Wikivoyage, Wiktionary, and others. Freebase[2] [10] is an extensive, collaboratively edited knowledge base that compiles data from various sources, including individual contributions and structured data from databases like Wikipedia. DBpedia[3] [4] represents information about millions of entities, including people, places, and things, by leveraging the infoboxes and categories present in Wikipedia articles. YAGO[4] [207] collects knowledge from Wikipedia, WordNet, and GeoNames. All the aforementioned KGs are open source and can provide world factual knowledge.

Commonsense KGs are another type of general KG. Different from encyclopedic KGs, they include abstract commonsense knowledge, such as semantic associations between concepts and causal relationships between events. Typical commonsense KGs include: ConceptNet[5] [136] is a semantic network built from nodes representing words or phrases connected by edges denoting semantic relationships. ATOMIC [88, 193] models the causal relationships between events. Commonsense

[1] https://www.wikidata.org/.
[2] https://www.freebase.be/.
[3] https://www.dbpedia.org/.
[4] https://yago-knowledge.org/.
[5] https://conceptnet.io/.

Table 2. Summary of Graph Construction Strategies

Graph Types	Nodes	Edges	Key Techniques	Representative Works
Tree Structures	Leaf nodes: Text chunks Non-leaf nodes: Semantic summaries	Hierarchical semantic relationships	Text chunk embedding, recursive clustering, LLM-generated summaries	RAPTOR [196], SiReRAG [267]
Document Graphs	Documents/Chunks	Inter-document semantic similarity, shared metadata (keywords/citations)	Rule-based (similarity/metadata), GNN modeling	ATLANTIC [166], KGP [231], GNN-Net [128], R4 [269]
KGs	Entities	Relational facts	NER + RE, textual enrichment, community detection	DALK [116], HippoRAG [67], REANO [51], LightRAG [66], TCR-QF [85], TEMPLE-MQA [27], Microsoft-GraphRAG [45]
Heterogeneous Graphs	Multi-type nodes	Multi-type edges	Network schema definition, node/edge instantiation, heterogeneous data integration	GRAFT-Net [209], NodeRAG [247], KG-Retriever [25], RA-Sim [183], HippoRAG2 [68]

KGs are primarily designed for commonsense reasoning tasks [162, 215], assisting models in reasoning by providing causal relationships between events.

(2) Domain KGs. As discussed in Section 1, domain-specific KGs are crucial for enhancing LLMs in addressing domain-specific questions. These KGs offer specialized knowledge in particular fields, aiding models in gaining deeper insights and a more comprehensive understanding of complex professional relationships. In the biomedical field, CMeKG[6] encompasses a wide range of data, including diseases, symptoms, treatments, medications, and relationships between medical concepts. CPubMed-KG[7] is a medical knowledge database in Chinese, building on the extensive repository of biomedical literature in PubMed. In the movie domain, Wiki-Movies [163] extracts structured information from Wikipedia articles related to films, compiling data about movies, actors, directors, genres, and other relevant details into a structured format. Additionally, Jin et al. [100] construct a dataset named GR-Bench, which includes five domain KGs spanning academic, E-commerce, literature, healthcare, and legal fields. Furthermore, He et al. [76] convert triplet-format and JSON files from ExplaGraphs and SceneGraphs into a standard graph format and selects questions requiring 2-hop reasoning from WebQSP to create the universal graph-format dataset GraphQA for evaluating GraphRAG systems.

5.1.2 Self-Constructed Graph Data. Self-constructed graphs enable the integration of proprietary or domain-specific knowledge into the retrieval process, offering significant flexibility for downstream tasks. Based on their structural characteristics and information organization principles, these graphs can be categorized into four main types, each with distinct construction methodologies and applications. Table 2 shows the summary of graph construction strategies.

Tree Structures. Tree structures hierarchically organize information using leaf nodes for text chunks and non-leaf nodes for summaries that aggregate semantically related content. Construction typically follows a bottom-up approach involving embedding text chunks, recursive clustering, and employing LLMs to generate cluster summaries [196, 267]. The key advantage lies in these summary nodes providing higher-level abstractions and more global contextual information, making this structure particularly valuable for tasks like QFS. However, this approach fails to capture fine-grained information such as specific entities and their relationships, limiting its effectiveness for fact-intensive retrieval tasks.

Document Graphs. Document graphs represent documents or passages as nodes, with edges modeling semantic or structural relationships such as citation networks [166]. The core principle involves capturing macroscopic discourse structure and thematic connections across document collections. Construction typically employs rule-based methods (e.g., connecting documents based on similarity

[6] https://cmekg.pcl.ac.cn/.
[7] https://cpubmed.openi.org.cn/graph/wiki.

or shared metadata [128, 231]) or learned approaches using GNNs [269]. Such graphs effectively capture thematic connections and inter-document structures, while the relationships represented by the edges are often shallow, limiting their ability to express complex semantic relationships.

KGs. KGs represent entities as nodes and relationships as edges, forming a semantic network that structures factual knowledge extracted from text. Typical construction begins with **Named Entity Recognition (NER)** and **Relation Extraction (RE)** models to extract triples from unstructured text [38, 51, 67, 85, 116]. An important extension is the textual KG, where nodes and edges are enriched with additional textual information. This enrichment takes two primary forms: some methods attach descriptive text or attributes to entities and relations [27, 66], while others employ text clustering coupled with LLM summarization to generate more global, structured summaries that encapsulate broader contextual information [45]. These textually enriched graphs significantly enhance semantic representation and help bridge the gap between structured knowledge and unstructured text, thereby facilitating more effective comprehension and processing by LLMs. However, their construction remains highly dependent on the performance of NER and RE models, and the incorporation of dense textual information can substantially increase the complexity of indexing and retrieval operations.

Heterogeneous Graphs. Heterogeneous graphs contain multiple types of nodes (e.g., entities, documents, summaries) and multiple types of edges, enabling the representation of complex, domain-specific knowledge structures. The core principle involves modeling rich interactions and semantics using a meta-schema that defines multiple entity and relation types. A typical construction method involves first defining a network schema, then instantiating nodes and edges of different types based on rules or learned models [25, 68, 209, 247]. This framework offers exceptional flexibility for domain-specific applications. For instance, patent analysis leverages co-occurrence and citation relations to connect patents and phrases [183], while customer service systems employ tree-structured representations with semantic similarity thresholds to model issue relations [249]. While this graph type enables powerful relational reasoning across diverse applications, the complex schema design and multi-relation retrieval typically introduce higher system complexity and computational overhead.

Discussion. Compared with open KGs, self-constructed graphs offer greater design flexibility and can be customized for specific downstream tasks. However, this approach still faces two critical challenges that require further research. First, since existing public resources cannot be directly utilized, self-constructed graphs typically rely on LLMs to extract information from raw documents, which is a computationally expensive process that requires balancing effectiveness against cost [45]. One potential solution involves utilizing open source general KGs as the initial graph structure, while dynamically extracting query-relevant structured information from the corpus during online queries to serve as supplementary data. Second, graph construction is inherently an information compression process, raising the fundamental question of how to maximize the retention of key semantic content while maintaining information conciseness [247]. This translates to determining the optimal graph construction paradigm that can effectively distill essential information while minimizing information loss.

5.2 Indexing

G-Indexing plays a crucial role in enhancing the efficiency and speed of query operations on graph databases, directly influencing subsequent retrieval methods and granularity. Common G-Indexing methods include graph indexing, text indexing, and vector indexing.

5.2.1 Graph Indexing. Graph indexing represents the most commonly used approach, preserving the entire structure of the graph. This method ensures that for any given node, all its edges and neighboring nodes are easily accessible. On the one hand, graph indexing can establish hierarchical relationships between elements. For instance, RAPTOR [196] and SiReRAG [267] index text chunks in tree structures, preserving the hierarchical relationships between text. KG-Retriever [25] develops a hierarchical index graph that models the relationships between entities and between documents separately while preserving the inclusion relationship between documents and entities. On the other hand, the graph method ensures that for any given node, all its edges and neighboring nodes are easily accessible. During subsequent retrieval stages, classic graph theory algorithms and graph traversals such as **Breadth-First Search (BFS)** or DFS can be employed to facilitate retrieval tasks [98, 100, 153, 155, 212, 217, 256].

5.2.2 Text Indexing. Text indexing involves converting graph data into textual descriptions to optimize retrieval processes. These descriptions are stored in a text corpus, where various text-based retrieval techniques, such as sparse retrieval, can be applied. Some approaches transform KGs into human-readable text using pre-defined rules or templates. For instance, Li et al. [121], Huang et al. [87], and Li et al. [127] use pre-defined templates to convert each triple in KGs into natural language, while Yu et al. [260] further merge triplets with the same head entity into passages. Additionally, some methods convert subgraph-level information into textual descriptions. For example, Edge et al. [45] perform community detection on the graph and generate summaries for each community using LLMs.

5.2.3 Vector Indexing. Vector indexing transforms graph data into vector representations to enhance retrieval efficiency, facilitating rapid retrieval and effective query processing. For example, **Entity Linking (EL)** and retrieval can be seamlessly applied through query embeddings, and efficient vector search algorithms such as Locality Sensitive Hashing [90] can be utilized. G-Retriever [76] employs LMs to encode textual information associated with each node and edge within the graph, while GRAG [80] and SEPTA [181] use GNNs and LMs to convert k-hop ego networks into graph embeddings, thereby better preserving structural information. It is noteworthy that existing literature has paid insufficient attention to the potential text embedding collapse phenomenon [275] in GraphRAG caused by long textual descriptions. Although this issue has not been systematically investigated, some studies have indirectly explored relevant solutions: for instance, Li et al. [121], He et al. [76], and Huang et al. [87] employ vectorized indexing for short texts like node/edge names or attributes, while Edge et al. [45] utilize LLMs to process longer texts such as community summaries. The differentiated approach may offer potential technical pathways for mitigating semantic collapse. The further utilization of robust embedding models represents a promising research direction worthy of future exploration [170].

Discussion. While the three aforementioned indexing methods differ in their implementation mechanisms, they exhibit close interconnections in practical applications. Specifically, graph search processes typically require simultaneous utilization of both topological structure information from neighboring nodes and textual attributes. In practice, this dual dependency necessitates a coordinated operation between graph and text indexing. Conversely, the generation of many textual descriptions (such as graph community summaries) inherently relies on the structured information provided by graph indexing. Meanwhile, text indexing and vector indexing are intrinsically linked through text vectorization techniques: converting text into dense vectors enables efficient semantic similarity matching. Together, these three indexing methods form a multi-layered retrieval system where graph indexing facilitates easy access to structural information, text indexing simplifies retrieval of textual content, and vector indexing enables quick and efficient searches. Therefore,

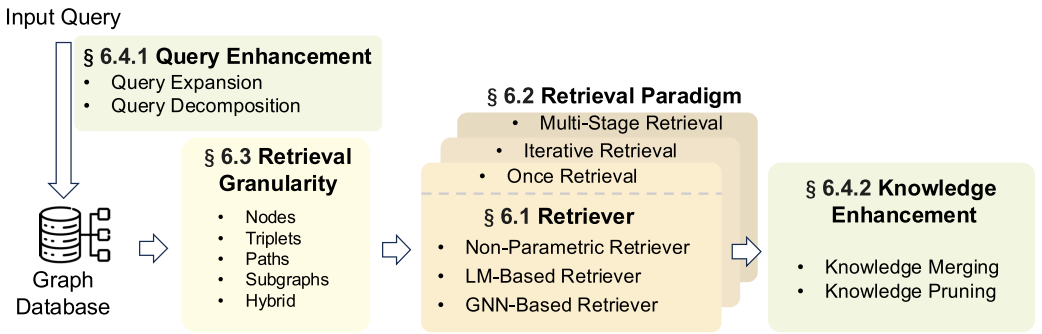

Fig. 5. The general architectures of G-Retrieval.

in practical applications, a hybrid approach combining these indexing methods is often preferred over relying solely on one. For instance, HybridRAG [195] retrieves both vector and graph data simultaneously to enhance the content retrieved, and EWEK-QA [36] uses both text and KGs. Furthermore, NodeRAG [247] simultaneously integrates graph indexing, text indexing, and vector indexing, which respectively employ exact matching for short texts, vector similarity search to identify relevant texts, and graph search to retrieve critical nodes.

6 G-Retrieval

G-Retrieval utilizes the interconnectedness of the graph to navigate and extract relevant information, enhancing the accuracy and relevance of the search process. The retrieval process is crucial for ensuring the quality and relevance of generated outputs, which mainly faces two significant challenges: (1) *Explosive Candidate Subgraphs*: As the graph size increases, the number of candidate subgraphs grows exponentially, requiring heuristic search algorithms to efficiently explore and retrieve relevant subgraphs. (2) *Insufficient Similarity Measurement*: Accurately measuring similarity between textual queries and graph data necessitates the development of algorithms capable of understanding both textual and structural information.

Considerable efforts have previously been dedicated to optimizing the retrieval process to address the above challenges. This survey focuses on examining various aspects of the retrieval process within GraphRAG, including the selection of the retriever, retrieval paradigm, retrieval granularity, and effective enhancement techniques. The general architectures of G-Retrieval are depicted in Figure 5.

6.1 Retriever

In GraphRAG, various retrievers possess unique strengths for addressing different aspects of retrieval tasks. We categorize retrievers into three types based on their underlying models: non-parametric retriever, LM-based retriever, and GNN-based retriever.

6.1.1 Non-Parametric Retriever. Non-parametric retrievers, based on heuristic rules or traditional graph search algorithms, do not rely on deep-learning models, thereby achieving high retrieval efficiency. For instance, Yasunaga et al. [256] and Taunk et al. [217] retrieve k-hop paths containing the topic entities of each question-choice pair. G-Retriever [76] enhances the conventional **Prize-Collecting Steiner Tree (PCST)** algorithm by incorporating edge prices and optimizing relevant subgraph extraction. Delile et al. [38], Mavromatis and Karypis [161] retrieve shortest paths related to entities mentioned in queries. Gutiérrez et al. [67], Gutiérrez et al. [68], and Alonso and Millidge [2] enhance the PageRank algorithm to rank relevant text chunks or entities. The selection of

specific non-parametric retrievers typically depends on both the requirements of downstream scenarios and the expertise of human specialists. It is important to note that these methods often involve an EL pre-processing step to locate nodes in the graph before retrieval.

6.1.2 LM-Based Retriever. LMs serve as effective retrievers in GraphRAG due to their strong natural language understanding capabilities. These models excel in processing and interpreting diverse natural language queries, making them versatile for a wide range of retrieval tasks within graph-based frameworks. We primarily categorized LMs into three types. The first one is to use LMs as encoders to generate embeddings of queries and then search for relevant subgraphs in the vector database [120, 121, 260, 263]. The second approach employs LMs to pre-generate relation paths or rules, which are then used to extract qualifying subgraphs from the graph. For instance, KG-GPT [104] adopts LLMs to generate the set of top-k relevant relations of the specific entity. Wold et al. [237] utilize fine-tuned GPT-2 to generate reasoning paths. StructGPT [92] utilizes LLMs to automatically invoke several pre-defined functions, by which relevant information can be retrieved and combined to assist further reasoning. The third type is to use LMs as searchers to search for reasoning paths on graphs. For example, Subgraph Retriever [265] trains RoBERTa [147] as the retriever, which expands from the topic entity and retrieves the relevant paths in a sequential decision process. With the advent of LLMs, numerous studies [155, 212] explore replacing smaller LMs with LLMs to leverage their powerful text comprehension and reasoning capabilities for graph searching. While the following approaches leverage LMs for retrieval, they exhibit distinct characteristics. Specifically, the first one demonstrates the highest efficiency but primarily focuses on textual information, with limited utilization of graph structures. The second heavily depends on the LM's ability to perceive graph structures, with its performance largely determined by the quality of generated relations or rules. The third approach progressively searches through the graph while simultaneously leveraging both structural and textual information, typically achieving superior performance at the cost of computational efficiency.

6.1.3 GNN-Based Retriever. GNNs are adept at understanding and leveraging complex graph structures. GNN-based retrievers typically encode graph data and subsequently score different retrieval granularities based on their similarity to the query. It should be noted that since these methods inherently involve query understanding, they necessarily rely on LMs for query representation. For example, GNN-RAG [161] first encodes the graph, assigns a score to each entity, and retrieves entities relevant to the query based on a threshold. However, due to modality differences, directly applying GNNs to represent graphs may result in graph embeddings residing in a different semantic space from query embeddings. To address this, Peng et al. [181] propose first aligning graph embeddings with text embeddings through contrastive learning, followed by employing pre-trained GNNs for graph retrieval.

Discussion. During the retrieval process, non-parametric retrievers exhibit good retrieval efficiency, but they may suffer from inaccurate retrieval due to a lack of training on downstream tasks. Meanwhile, although LM-based retrievers and GNN-based retrievers offer higher retrieval accuracy, they require significant computational overhead. Considering this complementarity, many methods propose hybrid retrieval approaches to improve both retrieval efficiency and accuracy. Many approaches adopt a multi-stage retrieval strategy, employing different models at each stage. For example, RoG [153] first utilizes LLMs to generate planning paths and then extracts paths satisfying the planning paths from KGs. GenTKGQA [58] infers crucial relations and constraints from the query using LLMs and extracts triplets according to these constraints. EtD [134] iterates multiple times to retrieve relevant paths. During each iteration, it first uses LLaMA2 [219] to select

edges connecting the current node, then employs GNNs to obtain embeddings of the new layer of nodes for the next round of LLM selection.

6.2 Retrieval Paradigm

Within GraphRAG, different retrieval paradigms, including once retrieval, iterative retrieval, and multi-stage retrieval, play crucial roles in improving the relevance and depth of the retrieved information. Once retrieval aims to gather all pertinent information in a single operation. Iterative retrieval conducts further searches based on previously retrieved information, progressively narrowing down to the most relevant results. Here, we further divide iterative retrieval into adaptive retrieval and non-adaptive retrieval, with the only difference lying in whether the stopping of the retrieval is determined by the model. Another retrieval paradigm is multi-stage retrieval, where retrieval is divided into multiple stages. Different types of retrievers may be employed at each stage for more precise and diversified search results. Below, we will provide a detailed introduction to these types of retrieval paradigms.

6.2.1 Once Retrieval. Once retrieval aims to retrieve all the relevant information in a single query. One category of approaches [67, 80, 121] utilizes embedding similarities to retrieve the most relevant pieces of information, which primarily employs the first type of LM-based retriever. Another category of methods designs pre-defined rules or patterns to directly extract specific structured information such as triplets, paths, or subgraphs from graph databases. For example, G-Retriever [76] utilizes an extended PCST algorithm to retrieve the most relevant subgraph. KagNet [130] extracts paths between all pairs of topic entities with lengths not exceeding k. Yasunaga et al. [256] and Taunk et al. [217] extract the subgraph that contains all topic entities along with their 2-hop neighbors. These methods adopt a non-parametric retriever or the second type of LM-based retriever.

Furthermore, in this subsection, we also include some multiple retrieval methods that involve decoupled and independent retrievals, allowing them to be computed in parallel and executed only once. For example, Luo et al. [153] first instruct LLMs to generate multiple reasoning paths and then use a BFS retriever to sequentially search for subgraphs in the KGs that match each path. KG-GPT [104] decomposes the original query into several sub-queries, retrieving relevant information for each sub-query in a single retrieval process.

6.2.2 Iterative Retrieval. In iterative retrieval, a series of similar retrieval steps is employed, with subsequent searches depending on the results of prior retrievals, which uses an iterative LM-based retriever. These methods aim to deepen the understanding or completeness of the retrieved information over successive iterations. In this survey, we further classify iterative retrieval into two categories: (1) non-adaptive and (2) adaptive retrieval. We provide a detailed summary of these two categories of methods below.

(1) Non-Adaptive Retrieval. Non-adaptive methods typically follow a fixed sequence of retrieval, and the termination of retrieval is determined by setting a maximum time or a threshold. For example, PullNet [208] retrieves problem-relevant subgraphs through T iterations. In each iteration, the article designs a retrieval rule to select a subset of retrieved entities and then expands these entities by searching relevant edges in the KG. In each iteration, KGP [231] first selects seed nodes based on the similarity between the context and the nodes in the graph. It then uses LLMs to summarize and update the context of the neighboring nodes of the seed nodes, which is utilized in the subsequent iteration.

(2) Adaptive Retrieval. One distinctive characteristic of adaptive retrieval is to let models autonomously determine the optimal moments to finish the retrieval activities, which means models can determine whether the retrieval process should continue based on the currently retrieved

information. For instance, Wu et al. [243] and Guo et al. [65] leverage an LM for hop prediction, which serves as an indicator to end the retrieval. There is also a group of researchers who utilize model-generated special tokens or texts as termination signals for the retrieval process. For example, ToG [155, 212] prompts the LLM agent to explore the multiple possible reasoning paths until the LLM determines that the question can be answered based on the current reasoning path. Zhang et al. [265] train a RoBERTa to expand a path from each topic entity. In the process, a virtual relation named "[END]" is introduced to terminate the retrieval process. Another common approach [22, 92, 94, 100, 203, 213, 229, 248] involves treating LLMs as agents for graph retrieval, enabling it to directly generate answers to questions to signal the end of iteration. These agents could autonomously determine the information for retrieval, invoke the pre-defined retrieval tools, and cease the retrieval process based on the retrieved information.

Compared to non-adaptive retrieval, adaptive retrieval can dynamically adjust based on query complexity, effectively addressing two critical limitations of traditional methods: under-retrieval and over-retrieval. Within this adaptive retrieval framework, the accuracy of termination directly impacts downstream task performance. Therefore, it requires both the design of rational adaptive control strategies and the selection of foundation models with strong reasoning capabilities.

6.2.3 Multi-Stage Retrieval. Multi-stage retrieval divides the retrieval process linearly into multiple stages, with additional steps such as retrieval enhancement, and even generation processes occurring between these stages. In multi-stage retrieval, different stages may employ various types of retrievers, which enables the system to incorporate various retrieval techniques tailored to different aspects of the query. For example, Wang et al. [230] first utilize a non-parametric retriever to extract n-hop paths of entities in the query's reasoning chain, then after a pruning stage, it further retrieves the one-hop neighbors of the entities in the pruned subgraph. OpenCSR [71] divides the retrieval process into two stages. In the first stage, it retrieves all 1-hop neighbors of the topic entity. In the second stage, it compares the similarity between these neighbor nodes and other nodes, selecting the top-k nodes with the highest similarity for retrieval. GNN-RAG [161] first employs GNNs to retrieve the top-k nodes most likely to be the answer. Subsequently, it retrieves all shortest paths between query entities and answer entities pairwise.

Discussion. In GraphRAG, once retrieval is typically exhibited lower complexity and shorter response times, making it suitable for scenarios requiring real-time responsiveness. In contrast, iterative retrieval often involves higher time complexity, especially when employing LLMs as retrievers, potentially leading to longer processing times. However, this approach can yield higher retrieval accuracy by iteratively refining retrieved information and generating responses. Therefore, the choice of retrieval paradigm should balance accuracy and time complexity based on specific use cases and requirements.

6.3 Retrieval Granularity

According to different task scenarios and indexing types, researchers design distinct retrieval granularities (i.e., the form of related knowledge retrieved from graph data), which can be divided into nodes, triplets, paths, and subgraphs. Each retrieval granularity has its own advantages, making it suitable for different practical scenarios. We will introduce the details of these granularities in the following sections.

6.3.1 Nodes. Nodes allow for precise retrieval focused on individual elements within the graph, which is ideal for targeted queries and specific information extraction. In general, for KGs, nodes refer to entities. For other types of text attribute graphs, nodes may include textual information that describes the node's attributes. By retrieving nodes within the graph, GraphRAG systems

could provide detailed insights into their attributes, relationships, and contextual information. For example, Munikoti et al. [166], Li et al. [128], and Wang et al. [231] construct document graphs and retrieve relevant passage nodes. Liu et al. [134] and Gutiérrez et al. [67] retrieve entities from constructed KGs.

6.3.2 Triplets. Generally, triplets consist of entities and their relationships in the form of subject-predicate-object tuples, providing a structured representation of relational data within a graph. The structured format of triplets allows for clear and organized data retrieval, making it advantageous in scenarios where understanding relationships and contextual relevance between entities is critical. Yang et al. [253] retrieve triplets containing topic entities as relevant information. Huang et al. [87], Li et al. [121], and Li et al. [127] first convert each triplet of graph data into textual sentences using pre-defined templates and subsequently adopt a text retriever to extract relevant triplets. However, directly retrieving triplets from graph data may still lack contextual breadth and depth, thus being unable to capture indirect relationships or reasoning chains. To address this challenge, Wang et al. [223] propose to generate the logical chains based on the original question and retrieve the relevant triplets of each logical chain.

6.3.3 Paths. The retrieval of path-granularity data can be seen as capturing sequences of relationships between entities, enhancing contextual understanding and reasoning capabilities. In GraphRAG, retrieving paths offers distinct advantages due to their ability to capture complex relationships and contextual dependencies within a graph.

However, path retrieval can be challenging due to the exponential growth in possible paths as the graph size increases, which escalates computational complexity. To address this, some methods retrieve relevant paths based on pre-defined rules. For example, Wang et al. [230] and Lo and Lim [148] first select entity pairs in the query and then traverse to find all the paths between them within n-hop. HyKGE [98] first defines three types of paths: path, co-ancestor chain, and co-occurrence chain and then utilizes corresponding rules to retrieve each of these three types of paths. In addition, some methods utilize models to perform path searching on graphs. ToG [155, 212] propose to prompt the LLM agent to perform the beam search on KGs and find multiple possible reasoning paths that help answer the question. Luo et al. [153], Wu et al. [243], and Guo et al. [65] first utilize the model to generate faithful reasoning plans and then retrieve relevant paths based on these plans. GNN-RAG [161] first identifies the entities in the question. Subsequently, all paths between entities that satisfy a certain length relationship are extracted.

6.3.4 Subgraphs. Retrieving subgraphs offers significant advantages due to its ability to capture comprehensive relational contexts within a graph. This granularity enables GraphRAG to extract and analyzes complex patterns, sequences, and dependencies embedded within larger structures, facilitating deeper insights and a more nuanced understanding of semantic connections.

To ensure both information completeness and retrieval efficiency, some methods propose an initial rule-based approach to retrieve candidate subgraphs, which are subsequently refined or processed further. Peng and Yang [183] retrieve the ego graph of the patent phrase from the self-constructed patent-phrase graph. Yasunaga et al. [256], Feng et al. [54], and Taunk et al. [217] first select the topic entities and their two-hop neighbors as the node set and then choose the edges with head and tail entities both in the node set to form the subgraph. Besides, there are also some embedding-based subgraph retrieval methods. For example, Hu et al. [80] first encode all the k-hop ego networks from the graph database, then retrieve subgraphs related to the query based on the similarities between embeddings. Wen et al. [235] and Li et al. [116] extract two types of graphs, including Path evidence subgraphs and Neighbor evidence subgraphs, based on pre-defined rules.

OpenCSR [71] starts from a few initial seed nodes and gradually expands to new nodes, eventually forming a subgraph.

In addition to the aforementioned direct subgraph retrieval methods, some works propose first retrieving relevant paths and then constructing related subgraphs from them. For instance, Zhang et al. [265] train a RoBERTa [147] to identify multiple reasoning paths through a sequential decision process, subsequently merging identical entities from different paths to induce a final subgraph.

6.3.5 Hybrid Granularities. Considering the advantages and disadvantages of various retrieval granularities mentioned above, some researchers propose using hybrid granularities, that is, retrieving relevant information in multiple granularities from graph data. This type of granularity enhances the system's ability to capture both detailed relationships and broader contextual understanding, thus reducing noise while improving the relevance of the retrieved data. Various previous works propose to utilize LLM agents to retrieve complex hybrid information. Jin et al. [100], Jiang et al. [92], Jiang et al. [94], Wang et al. [229], and Sun et al. [213] propose to adopt LLM-based agents for adaptively selecting nodes, triplets, paths, and subgraphs.

Discussion. (1) In real applications, there are no clear boundaries between these retrieval granularities, as subgraphs can be composed of multiple paths, and paths can be formed by several triplets. (2) Various granularities such as nodes, triplets, paths, and subgraphs offer distinct advantages in the GraphRAG process. Balancing between retrieval content and efficiency is crucial when selecting the granularity, depending on the specific context of the task. For straightforward queries or when efficiency is paramount, finer granularities such as entities or triplets may be preferred to optimize retrieval speed and relevance. In contrast, complex scenarios often benefit from a hybrid approach that combines multiple granularities. This approach ensures a more comprehensive understanding of the graph structure and relationships, enhancing the depth and accuracy of the generated responses. Thus, GraphRAG's flexibility in granularity selection allows it to adapt effectively to diverse IR needs across various domains.

6.4 Retrieval Enhancement

To ensure high retrieval quality, researchers propose techniques to enhance both user queries and the knowledge retrieved, which respectively enhance the input and output of the retrieval process. In this article, we categorize query enhancement into query expansion and query decomposition, and knowledge enhancement into merging and pruning. These strategies collectively optimize the retrieval process. Although other techniques such as query rewriting [156, 159, 182, 187] are commonly used in RAG, they are less frequently applied in GraphRAG. We do not delve into these methods, despite their potential adaptation for GraphRAG.

6.4.1 Query Enhancement. Strategies applied to queries typically involve pre-processing techniques that enrich the information for better retrieval. This may include query expansion and query decomposition. The former primarily focuses on supplementing information for short queries, while the latter mainly specializes in decomposing complex queries.

(1) Query Expansion. Due to the generally short length of queries and their limited information content, query expansion aims to improve search results by supplementing or refining the original query with additional relevant terms or concepts. The majority of current related work primarily relies on the intrinsic knowledge of LLMs. For instance, Luo et al. [153] and Cheng et al. [27] generate relation paths grounded by KGs with LLMs to enhance the retrieval query. HyKGE [98] utilizes a large model to generate the hypothesis output of the question, concatenating the hypothesis output with the query as input to the retriever. However, LLMs may suffer from hallucination issues, potentially introducing incorrect information into enhanced queries, which is a critical aspect

that has received limited attention in prior work. A promising solution involves incrementally expanding queries during retrieval, similar to the ITER-RETGEN [199].

(2) Query Decomposition. Query decomposition techniques break down or decompose the original user query into smaller, more specific sub-queries. Each sub-query typically focuses on a particular aspect or component of the original query, which successfully alleviates the complexity and ambiguity of language queries. For instance, Kim et al. [104] and Choudhary and Reddy [29] break down the primary question into sub-sentences, each representing a distinct relation, and sequentially retrieve the pertinent triplets for each sub-sentence. The aforementioned methods primarily perform linear decomposition of queries, breaking down the original query into several parallel sub-questions. However, in the RAG framework, some studies [211] adopt recursive tree-structured decomposition of queries, which remains understudied in GraphRAG.

6.4.2 Knowledge Enhancement. After retrieving initial results, knowledge enhancement strategies are employed to refine and improve the retriever's results. This phase often involves knowledge merging and knowledge pruning processes to present the most pertinent information prominently. These techniques aim to ensure that the final set of retrieved results is not only comprehensive but also highly relevant to the user's information needs.

(1) Knowledge Merging. The retrieved information often exhibits distinctively structured relational patterns. Specifically, adjacent triplets can form continuous reasoning paths, while triplets sharing common entities constitute localized semantic subgraphs. By effectively integrating these intrinsically connected information units, one can construct more comprehensive knowledge structures. Knowledge merging retrieved information enables compression and aggregation of information, which assists in obtaining a more comprehensive view by consolidating relevant details from multiple sources. This approach not only enhances the completeness and coherence of the information but also mitigates issues related to input length constraints in models. KnowledgeNavigator [65] merges nodes and condenses the retrieved sub-graph through triple aggregation to enhance the reasoning efficiency. In subgraph retrieval [265], after retrieving top-k paths from each topic entity to form a single subgraph, researchers propose to merge the same entities from different subgraphs to form the final subgraph. Wen et al. [235] and Li et al. [116] merge retrieved subgraphs based on relations, combining head entities and tail entities that satisfy the same relation into two distinct entity sets, ultimately forming relation paths.

(2) Knowledge Pruning. The informativeness of retrieval results does not exhibit a positive correlation with model performance. Excessive retrieved content often contains semantically irrelevant information, and such noisy data may interfere with the model's reasoning process and degrade prediction accuracy [64]. Consequently, effective noise filtering becomes essential. Knowledge pruning involves filtering out less relevant or redundant retrieved information to refine the results. Previous approaches for pruning encompass two main categories: reranking-based approaches and LLM-based approaches.

Reranking methods involve the reordering or prioritization of retrieved information using tailored metrics or criteria. One line of methods introduces stronger models for reranking. For example, Li et al. [121] concatenate each retrieved triplet with the question-choice pair and adopt a pre-trained cross-encoder [190] to rerank the retrieved triplets. Jiang et al. [98] utilize the FlagEmbedding to encode the text to rerank top-k documents returned by embedding model "BGE Reranker." Another category utilizes more fine-grained similarity information for ranking. For instance, Cheng et al. [27] rerank the candidate subgraphs based on the similarity for both relation and fine-grained concept between subgraphs and the query. Taunk et al. [217] first cluster the 2-hop neighbors and then delete the cluster with the lowest similarity score with the input query. Yasunaga et al. [256]

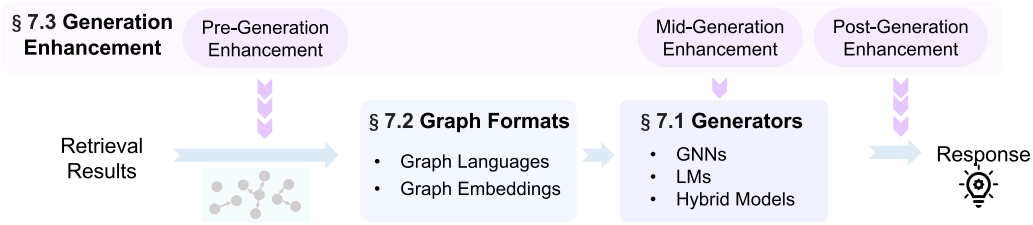

Fig. 6. The overview of G-Generation.

prune the retrieved subgraph according to the relevance score between the question context and the KG entity nodes calculated by a pre-trained LM. G-G-E [57] first divides the retrieved subgraph into several smaller subgraphs and then compares the similarity between each smaller subgraph and the query. Subgraphs with low similarity are removed, and the remaining smaller subgraphs are merged into a larger subgraph. Additionally, a third category of methods introduces new metrics for reranking. For example, Munikoti et al. [166] propose a metric that measures both the impact and recency of the retrieved text chunks. KagNet [130] decomposes the retrieved paths into triplets and reranks the paths based on the confidence score measured by the KG embedding techniques. Wang et al. [230], Jiang et al. [95], and Luo et al. [150] adopt the Personalized PageRank algorithm to rank the retrieved candidate information for further filtering.

LLM-based methods excel in capturing complex linguistic patterns and semantic nuances, which enhances their ability to rank search results or generate responses more accurately. These pruning techniques typically treat the pruning process as a discriminative task, where the LLM is employed to make informed judgments about which data points are pertinent and which should be discarded. For example, Wang et al. [230] propose to prune the irrelevant graph data by calling LLMs to check. By leveraging the LLM's deep understanding of language and context, the pruning process becomes more precise, effectively filtering out unnecessary or misleading information that could degrade the quality of the results. From another perspective, LLM-based methods inherently perform coarse-grained reranking by retaining relevant information while filtering out irrelevant content.

7 G-Generation

G-Generation integrates the rich contextual and relational data from the graph to produce more coherent and comprehensive generated content, thereby improving the overall quality and depth of the output. In this stage, suitable generation models must be selected based on the downstream tasks. The retrieved graph data are then transformed into formats compatible with the generators. The generator takes both the query and the transformed graph data as inputs to produce the final response. Beyond these fundamental processes, generation enhancement techniques can further improve the output by intensifying the interaction between the query and the graph data and enriching the content generation itself. The organization of this section and the overview of G-Generation are depicted in Figure 6.

7.1 Generators

The selection of generators often depends on the type of downstream task at hand. For discriminative tasks (e.g., multi-choice question answering) or generative tasks that can be formulated as discriminative tasks (e.g., KBQA), one can utilize GNNs or discriminative LMs to learn representations of the data. These representations can then be mapped to the logits associated with different answer options to provide responses. Alternatively, generative LMs can be employed to directly generate answers. For generative tasks, however, the use of GNNs and discriminative LMs alone

is insufficient. These tasks require the generation of text, which necessitates the deployment of decoders.

7.1.1 GNNs. Due to the powerful representational capabilities of GNNs for graph data, they are particularly effective for discriminative tasks. GNNs can directly encode graph data, capturing complex relationships and node features inherent in the graph structure. This encoding is then processed through a **Multi-Layer Perceptron (MLP)** to generate predictive outcomes. These approaches primarily utilize classical GNN models (e.g., GCN [106], GAT [221], GraphSAGE [69], and Graph Transformers [201]), either in their original form or modified to enhance the expressive power and better align with downstream tasks. For example, HamQA [42] designs a hyperbolic GNN to learn the representations of retrieved graph data, which learns from the mutual hierarchical information between query and graphs. Sun et al. [209] compute PageRank scores for neighboring nodes and aggregates them weighted by these scores, during message passing. This approach enhances the central node's ability to assimilate information from its most relevant neighboring nodes. Mavromatis and Karypis [160] decode the query into several vectors (instructions) and enhance instruction decoding and execution for effective reasoning by emulating BFS with GNNs to improve instruction execution and using adaptive reasoning to update the instructions with KG-aware information. RAGraph [97] utilizes the strong GNN representation power to process the retrieved subgraphs. It should be noted that the use of GNNs as generators does not completely exclude the involvement of LM. In practice, most current studies first utilize LMs to encode textual information in the graph, deriving features for nodes or edges, while simultaneously incorporating the query's semantic information into the graph structure.

7.1.2 LMs. LMs possess strong capabilities in text understanding, which also allows them to function as generators. In the context of integrating LMs with graph data, it is necessary to first convert the retrieved graph data into specific graph formats. This conversion process ensures that the structured information is effectively understood and utilized by the LMs. These formats, which will be elaborated on in Section 7.2, are crucial for preserving the relational and hierarchical structure of the graph data, thereby enhancing the model's ability to interpret complex data types. Once the graph data is formatted, it is then combined with a query and fed into an LM.

For encoder-only models, such as BERT [40] and RoBERTa [147], their primary use is in discriminative tasks. Similar to GNNs, these models first encode the input text and then utilize MLPs to map it to the answer space [87, 95, 121]. On the other hand, encoder-decoder and decoder-only models, such as T5 [188], GPT-4 [173], and LLaMA [44], are adept at both discriminative and generative tasks. These models excel in text understanding, generation, and reasoning, allowing them to process textual inputs directly and generate textual responses [45, 98, 100, 153, 161, 212, 223, 230]. Additionally, due to their powerful zero-shot and few-shot capabilities, rendering fine-tuning no longer mandatory, LLMs have emerged as the predominant generator paradigm in recent years.

7.1.3 Hybrid Models. Considering the strengths of GNNs at representing the structure of graph data, and the robust understanding of text demonstrated by LMs, many studies are exploring the integration of these two technologies to generate coherent responses. This article categorizes the hybrid generative approaches into two distinct types: cascaded paradigm and parallel paradigm.

(1) Cascaded Paradigm. In the cascaded approaches, the process involves a sequential interaction where the output from one model serves as the input for the next. Specifically, the GNN processes the graph data first, encapsulating its structural and relational information into a form that the LM can understand. Subsequently, this transformed data is fed into the LM, which then generates the final text-based response. These methods leverage the strengths of each model in a step-wise fashion, ensuring detailed attention to both structural and textual data.

In these methods, prompt tuning [110, 122, 143, 144] is a typical approach, where GNNs are commonly employed to encode the retrieved graph data. The encoded graph data are subsequently pre-pended as a prefix to the input text embeddings of an LM. The GNN is then optimized through downstream tasks to produce enhanced encodings of the graph data [58, 76, 80, 266].

(2) Parallel Paradigm. On the other hand, the parallel approach operates by concurrently utilizing the capabilities of both the GNN and the LLM. In this setup, both models receive the initial inputs simultaneously and work in tandem to process different facets of the same data. The outputs are then merged, often through another model or a set of rules, to produce a unified response that integrates insights from both the graphical structure and the textual content.

In the parallel paradigm, a typical approach involves separately encoding inputs using both GNNs and LMs, followed by integrating these two representations, or directly integrating their output responses. For instance, Jiang et al. [93] aggregate predictions from GNNs and LMs by weighted summation to obtain the final answer. Lin et al. [130] and Pahuja et al. [176] integrate the graph representations derived from GNNs and the text representations generated by LMs using attention mechanisms. Yasunaga et al. [256], Munikoti et al. [166], and Taunk et al. [217] directly concatenate graph representations with text representations.

Another approach involves designing dedicated modules that integrate GNNs with LMs, enabling the resulting representations to encapsulate both structural and textual information. For instance, Zhang et al. [270] introduce a module called the GreaseLM Layer, which incorporates both GNN and LM layers. At each layer, this module integrates textual and graph representations using a two-layer MLP before passing them to the next layer. Similarly, ENGINE [278] proposes G-Ladders, which combine LMs and GNNs through a side structure, enhancing node representations for downstream tasks.

Hybrid models that harness both the representation capabilities of GNNs for graph data and LMs for text data hold promising applications. However, effectively integrating information from these two modalities remains a significant challenge.

Discussion. The choice of generators (i.e., GNNs, LMs, or hybrid models) depends on task-specific requirements. While GNNs excel at discriminative tasks by explicitly encoding graph structures, they lack generative capabilities. In contrast, LMs perform well in both discriminative and generative tasks but heavily rely on effective graph serialization. Hybrid models combine the strengths of both approaches but face integration challenges, including modality alignment and computational inefficiency. Future research should focus on developing seamless, lightweight fusion techniques to preserve structural and semantic information, optimizing joint training frameworks, and reducing computational overhead. Addressing these challenges will be critical for unlocking the full potential of hybrid architectures as powerful, versatile generators.

7.2 Graph Formats

When using GNNs as generators, the graph data can be directly encoded. However, when utilizing LMs as generators, the non-Euclidean nature of graph data poses a challenge, as it cannot be directly combined with textual data for input into the LMs. To address this, graph translators are employed to convert the graph data into a format compatible with LMs. This conversion enhances the generative capabilities of LMs by enabling them to effectively process and utilize structured graph information. In this survey, we summarize two distinct graph formats: graph languages and graph embeddings. The former primarily converts graph data into a specific language format for LLM input via prompts, while the latter integrates graphs into LLMs from a semantic space perspective.

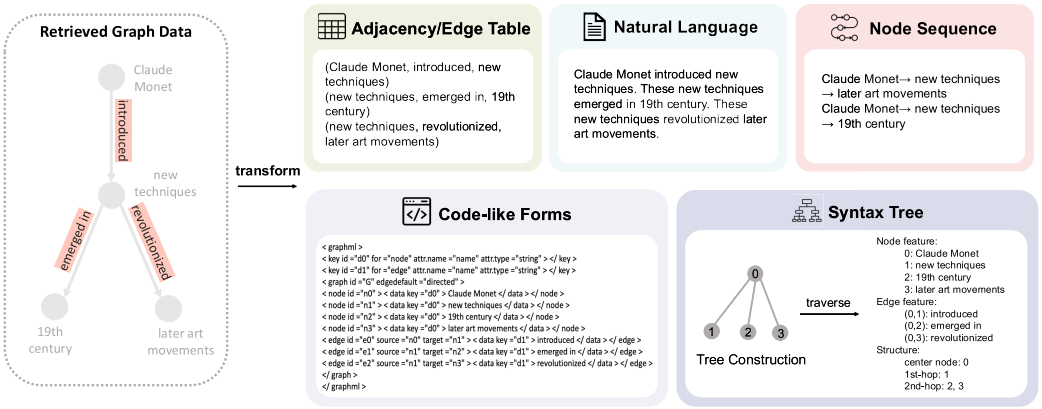

Fig. 7. Illustration of the graph languages. Given the retrieved subgraph on the left part, we show how to transform it into an adjacency/edge table, natural language, node sequence, code-like forms, and syntax trees to adapt the input form requirements of different generators.

7.2.1 Graph Languages. A graph description language is a formalized system of notation that is specifically crafted to characterize and represent graph data. It prescribes a uniform syntax and semantic framework that describes the components and interconnections within a graph. Through these languages, users can consistently generate, manipulate, and interpret graph data in a comprehensible format for machines. They enable the definition of graph architectures, the specification of attributes for nodes and edges, and the implementation of operations and queries on graph structures. Next, we will introduce five types of graph languages separately: Adjacency/Edge Table, Natural Language, Codes, Syntax Tree, and Node Sequence. We illustrate this process with an example in Figure 7, with detailed introductions provided below.

(1) Adjacency/Edge Table. The adjacency table and the edge table are widely used methods for describing graph structures [52, 63, 125, 224]. The adjacency table enumerates the immediate neighbors of each vertex, offering a compact way to represent connections in sparse graphs. For example, KG-GPT [104] linearizes the triples in the retrieved subgraph, which are then concatenated and fed into the LLMs. Conversely, the edge table details all the edges within the graph, providing a straightforward representation that is particularly useful for processing and analyzing graphs in a linear format. Both two methods are brief, easy to understand, and intuitive.

(2) Natural Language. Given that user queries are typically presented in natural language, and considering the outstanding natural language comprehension capabilities of LMs, it becomes a compelling approach to describe the retrieved graph data using natural language. By translating graph data into descriptive, easily comprehensible language, LMs can bridge the gap between raw data representation and user-friendly information, facilitating more effective interactions with data-driven applications. For example, some researchers [87, 121] propose defining a natural language template for each type of edge in advance and subsequently filling in the endpoints of each edge into the corresponding template based on its type. Ye et al. [257] employ natural language to describe the information of 1-hop and 2-hop neighboring nodes of the central node. Edge et al. [45] utilize LLMs to generate report-like summaries for each detected graph community. Wu et al. [243] and Guo et al. [65] adopt LMs to rewrite the edge table of retrieved subgraphs, generating a natural language description. Fatemi et al. [52] explore different representations of nodes (e.g., Integer encoding, alphabet letters, names) and edges (e.g., parenthesis, arrows, incident). Jin et al.

[100], Jiang et al. [92], Jiang et al. [94], Wang et al. [229], and Sun et al. [213] integrate information from different granularities within the graph into prompts through natural language in the form of dialogue.

(3) Code-Like Forms. Considering that natural language descriptions and other 1D sequences are inherently inadequate for directly representing the 2D structure of graph data and given the robust code comprehension capabilities of LMs, many researchers [63] explore using code-like formats to represent graph structures. For example, Guo et al. [63] examine the use of Graph Modeling Language [77] and Graph Markup Language [191] for representing graphs. These standardized languages are specifically designed for graph data, providing comprehensive descriptions that encompass nodes, edges, and their interrelationships.

(4) Syntax Tree. Compared to direct flattening of graphs, some research [272] propose transforming graphs into structures akin to syntax trees. Syntax trees possess a hierarchical structure and, being topological graphs, also maintain a topological order. This method retains more structural information, enhancing the understanding and analysis of the graph's intrinsic properties. Such a transformation not only preserves the relational dynamics between different graph elements but also facilitates more sophisticated algorithms for graph analysis and processing. GRAPHTEXT [272] proposes transforming the ego network of a central node into a graph-syntax tree format. This format not only encapsulates structural information but also integrates the features of the nodes. By traversing this syntax tree, it is possible to obtain a node sequence that maintains both topological order and hierarchical structure.

(5) Node Sequence. Some studies [23, 161, 250] propose representing graphs through sequences of nodes, which are often generated using pre-defined rules. Compared to natural language descriptions, these node sequences are more concise and incorporate prior knowledge, specifically the structural information emphasized by the rules. Luo et al. [153] and Sun et al. [212] transform the retrieved paths into node sequences and input them into an LLM to enhance the task performance. LLaGA [23] proposes two templates that can transform graphs into node sequences. The first template, known as the Neighborhood Detail Template, offers a detailed examination of the central node along with its immediate surroundings. The second, termed the Hop-Field Overview Template, provides a summarized perspective of a node's neighborhood, which can be expanded to encompass broader areas. GNN-RAG [161] inputs the retrieved reasoning paths into LMs in the form of node sequences as prompts. Xypolopoulos et al. [250] propose several graph linearization methods, including graph centrality, degeneracy, and node relabeling schemes to enhance the ability of LLMs to understand graph data.

Discussion. Good graph languages should be complete, concise, and comprehensible. Completeness entails capturing all essential information within the graph structure, ensuring no critical details are omitted. Conciseness refers to the necessity of keeping textual descriptions brief to avoid the "lost in the middle" phenomenon [141] or exceeding the length limitations of LMs. Lengthy inputs can hinder LMs' processing capabilities, potentially causing loss of context or truncated data interpretation. Comprehensibility ensures that the language used is easily understood by LLMs, facilitating accurate representation of the graph's structure. Among these graph languages, the edge table offers advantages in conciseness and comprehensibility. However, its loss of 2D topological features significantly increases the difficulty for models to perform complex graph reasoning tasks. Similarly, natural language provides better comprehensibility but suffer from reduced conciseness. In contrast, code-like forms and syntax trees can better preserve structural features of graphs, though at the cost of noticeably compromised comprehensibility. Furthermore, node sequences designed based on heuristic rules can highlight specific structural characteristics,

but the effectiveness of such methods heavily relies on domain experts' prior knowledge. Due to the characteristics of different graph languages, their choice can significantly impact the performance of downstream tasks [52]. However, as there is currently no consensus regarding the selection of these graph languages, empirical experiments may be required to determine the optimal choice in practical applications.

7.2.2 Graph Embeddings. The above graph language methods transform graph data into text sequences, which may result in overly lengthy contexts, incurring high computational costs and potentially exceeding the processing limits of LLMs. Additionally, LLMs currently struggle to fully comprehend graph structures even with graph languages [63]. Thus, using GNNs to represent graphs as embeddings presents a promising alternative. The core challenge lies in integrating graph embeddings with textual representations into a unified semantic space. Current research focuses on utilizing prompt tuning methodologies, as discussed earlier. There are also some methods that adopt Fusion-in-Decoder [89, 261], which first convert the graph data into text, then encode it using an LM-based encoder and input it into the decoders [41, 51, 260]. Notably, feeding graph representations into LMs is feasible primarily with open source LMs, not closed-source models like GPT-4 [173]. While graph embedding methods avoid handling long text inputs, they face other challenges. For instance, there is difficulty in preserving precise information like specific entity names and poor generalization, and it requires large-scale data for training graph encoders.

7.3 Generation Enhancement

In the generation phase, besides converting the retrieved graph data into formats acceptable by the generator and inputting it together with the query to generate the final response, many researchers explore various methods of generation enhancement techniques to improve the quality of output responses. These methods can be classified into three categories based on their application stages: pre-generation enhancement, mid-generation enhancement, and post-generation enhancement. Pre-generation enhancement focuses on augmenting the generator's input, primarily aiming to more tightly integrate the retrieved graphs with the query. Mid-generation enhancement mainly employs constrained decoding approaches and reasoning paradigms like CoT to refine the generation process. Post-generation enhancement improves the output quality, typically through aggregation of multiple generated results.

7.3.1 Pre-Generation Enhancement. Pre-generation enhancement techniques focus on improving the quality of input data or representations before feeding them into the generator. In fact, there is no clear boundary between pre-generation enhancement and retrieval. In this survey, we categorize the retrieval stage as the process of retrieving knowledge from the original graph, and merging and pruning retrieved knowledge. Subsequent operations are considered pre-generation enhancements.

Commonly used pre-generation enhancement approaches primarily involve semantically enriching the retrieved graph data to achieve tighter integration between the graph data and textual query. Wu et al. [243] employ LLMs to rewrite retrieved graph data, enhancing the naturalness and semantic richness of the transformed natural language output. This method not only ensures that graph data is converted into more fluent and natural language but also enriches its semantic content. Conversely, DALK [116] utilizes the retrieved graph data to rewrite the query. Cheng et al. [27] first leverage LLMs to generate a reasoning plan and answer queries according to the plan. Taunk et al. [217] and Yasunaga et al. [256] aim to enhance GNNs by enabling them to learn graph representations relevant to queries. They achieve this by extracting all nouns from the QA pairs (or the QA pairs themselves) and inserting them as nodes into the retrieved subgraph. Mavromatis and Karypis [160] propose a method where, prior to generation, the representation of the query is decomposed into multiple vectors termed "instructions," each representing different features

of the query. These instructions are used as conditions during message passing when applying GNNs to learn from retrieved subgraphs. In addition, there are methods that incorporate additional information beyond graph data. For example, PullNet [208] incorporates documents relevant to entities, and MVP-Tuning [87] retrieves other related questions.

7.3.2 Mid-Generation Enhancement. Mid-generation enhancement involves techniques applied during the generation process. These methods typically adjust the generation strategies based on intermediate results or contextual cues. TIARA [205] introduces constrained decoding to control the output space and reduce generation errors. When generating LFs, if the constrained decoder detects that it is currently generating a pattern item, it restricts the next generated token to options that exist in tries containing KB classes and relations. Compared with the Beam Search, this approach ensures that pattern items generated are guaranteed to exist in the KG, thereby reducing generation errors. There are other methods of adjusting the prompts of LLMs to achieve multi-step reasoning. For example, MindMap [235] not only produces answers but also generates the reasoning process.

7.3.3 Post-Generation Enhancement. Post-generation enhancement occurs after the initial response is generated. Post-generation enhancement methods primarily involve integrating multiple generated responses to obtain the final response. Some methods adopt a divide and conquer paradigm, which focuses on integrating outputs from the same generator under different conditions or inputs. For example, Edge et al. [45] generate a summary for each graph community, followed by generating responses to queries based on the summary and then scoring these responses using an LLM. Ultimately, the responses are sorted in descending order according to their scores and sequentially incorporated into the prompt until the token limit is reached. Subsequently, the LLM generates the final response. Wang et al. [223] and Kim et al. [104] first decompose the query into several sub-questions, then generate answers for each sub-question, and finally merge the answers of all sub-questions to obtain the final answer. Alternatively, other methods combine or select responses generated by different models, which can be discarded as an ensemble strategy. Lin et al. [130] and Jiang et al. [93] combine the outputs generated by both GNNs and LLMs to reach a synergistic effect. UniOQA [127] explores two methods for generating answers: one involves generating queries in Cypher Query Language to execute and obtain results, while the other method directly generates answers based on retrieved triplets. The final answer is determined through a dynamic selection mechanism. In EmbedKGQA [197], besides the learned scoring function, researchers additionally design a rule-based score based on the graph structures. These two scores are combined to find the answer entity. Li et al. [125] combine answers based on retrieved graph data with responses generated according to the LLM's own knowledge. In addition to integrating multiple responses, KALMV [7] trains a verifier to judge whether the generated answer is correct, and if it is not, to further determine whether the error is due to generation or retrieval.

8 Training

In this section, we summarize the individual training of retrievers, generators, and their joint training. We categorize previous works into training-free and training-based approaches based on whether explicit training is required. Training-free methods are commonly employed when using closed-source LLMs such as GPT-4 [173] as retrievers or generators. These methods primarily rely on carefully crafted prompts to control the retrieval and generation capabilities of LLMs. Despite LLMs' strong abilities in text comprehension and reasoning, a challenge of training-free methods lies in the potential sub-optimality of results due to the lack of specific optimization for downstream tasks. Conversely, training-based methods involve training or fine-tuning models

using supervised signals. These approaches enhance the model performance by adapting them to specific task objectives, thereby potentially improving the quality and relevance of retrieved or generated content. Joint training of retrievers and generators aims to enhance their synergy, thereby boosting performance on downstream tasks. This collaborative approach leverages the complementary strengths of both components to achieve more robust and effective results in IR and content generation applications.

8.1 Training Strategies of Retriever

8.1.1 Training-Free. There are two primary types of training-free retrievers currently in use. The first type consists of non-parametric retrievers. These retrievers rely on pre-defined rules or traditional graph search algorithms rather than specific models [217, 256]. The second type utilizes pre-trained LMs as retrievers. Specifically, one group of works utilizes pre-trained embedding models to encode the queries and perform retrieval directly based on the similarity between the query and graph elements [121]. Another group of works adopts generative LMs for training-free retrieval. Candidate graph elements such as entities, triples, paths, or subgraphs are included as part of the prompt input to the LLMs. The LLMs then leverage semantic associations to select appropriate graph elements based on the provided prompt [45, 100, 104, 161, 212, 223, 230]. These methods harness the powerful semantic understanding capabilities of LMs to retrieve relevant graph elements without the need for explicit training.

8.1.2 Training-Based. Compared to training-free methods, model training becomes essential when the retriever employs either GNNs or smaller LMs. Furthermore, training the retriever on downstream tasks enables it to learn the correspondence between queries and target graphs, thereby significantly enhancing retrieval performance.

When the retrieval granularity is nodes or triplets, many methods train retrievers to maximize the similarity between the retrieval ground truth and the query. For instance, MemNNs [11] leverages metric learning to closely align the ground truth with the query in semantic space while differentiating unrelated facts from the query. On the contrary, when the retrieval granularity is paths, training retrievers often adopts an autoregressive approach, where the previous relationship path is concatenated to the end of the query. The model then predicts the next relation based on the concatenated input [65, 243].

However, the lack of ground truth for retrieval content in the majority of datasets poses a significant challenge. To address this issue, many methods attempt to construct reasoning paths based on distant supervision to guide retriever training. For example, Zhang et al. [265], Feng et al. [53], and Luo et al. [153] extract all paths (or shortest paths) between entities in the queries and entities in the answers, using them as training data for the retriever. In addition, Zhang et al. [265] also employ a relationship extraction dataset for distant supervision in unsupervised settings. There is another category of methods that utilize implicit intermediate supervision signals to train Retrievers. For instance, NSM [73] employs a bidirectional search strategy, where two retrievers start searching from the head entity and tail entity, respectively. The supervised objective is to ensure that the paths searched by the two retrievers converge as closely as possible. KnowGPT [268] and MINERVA [32] treat the selection of adjacent nodes to build paths or subgraphs as a Markov process. They design the reward function around the inclusion of the answer in the retrieved information and adopt reinforcement learning methods, e.g., policy gradient to optimize the retriever.

Some methods argue that distant supervision signals or implicit intermediate supervision signals may contain considerable noise, making it challenging to train effective retrievers. Therefore, they consider employing self-supervised methods for pre-training retrievers. SKP [41] pre-trains the

Dense Passage Retrieval model [103]. Initially, it conducts random sampling on subgraphs and transforms the sampled subgraphs into passages. Subsequently, it randomly masks passages, trains the model using a Masked LM, and employs contrastive learning by treating the masked passages and original passages as positive pairs for comparison.

8.2 Training of Generator

8.2.1 Training-Free. Training-free generators primarily cater to closed-source LLMs or scenarios where avoiding high training costs is essential. In these methods, the retrieved graph data is fed into the LLM alongside the query. The LLMs then generate responses based on the task description provided in the prompt, heavily relying on their inherent ability to understand both the query and the graph data.

8.2.2 Training-Based. Training the generator can directly receive supervised signals from downstream tasks. For generative LLMs, fine-tuning can be achieved using supervised fine-tuning, where task descriptions, queries, and graph data are inputted, and the output is compared against the ground truth for the downstream task [76, 80, 153]. On the other hand, for GNNs or discriminative models functioning as generators, specialized loss functions tailored to the downstream tasks are employed to train the models effectively [93, 121, 217, 256, 270].

8.3 Joint Training

Jointly training retrievers and generators simultaneously enhances performance on downstream tasks by leveraging their complementary strengths. The key to joint training lies in promoting the synergistic enhancement of the performance of both the retriever and the generator, while avoiding any mutual constraints between them. Some approaches unify retrievers and generators into a single model, typically LLMs, and train them with both retrieval and generation objectives simultaneously [153]. This method capitalizes on the cohesive capabilities of a unified architecture, enabling the model to seamlessly retrieve relevant information and generate coherent responses within a single framework.

Other methodologies involve initially training retrievers and generators separately, followed by joint training techniques to fine-tune both components. For instance, Subgraph Retriever [265] adopts an alternating training paradigm, where the retriever's parameters are fixed to use the graph data for training the generator. Subsequently, the generator's parameters are fixed, and feedback from the generator is used to guide the retriever's training. This iterative process helps both components refine their performance in a coordinated manner.

9 Applications and Evaluation

In this section, we will summarize the downstream tasks, application domains, benchmarks and metrics, and industrial applications related to GraphRAG. Table 3 collects existing GraphRAG techniques, categorizing them by downstream tasks, benchmarks, methods, and evaluation metrics. This table serves as a comprehensive overview, highlighting the various aspects and applications of GraphRAG technologies across different domains.

9.1 Downstream Tasks

GraphRAG is applied in various downstream tasks (especially NLP tasks) by effectively leveraging external graph-structured knowledge, including question answering, information extraction, and others.

Table 3. The Tasks, Benchmarks, Methods, and Metrics of GraphRAG

Tasks		Benchmarks	Methods	Metrics
QA	KBQA	WebQSP [259]	[5–7, 22, 28, 36, 41, 57, 65, 73, 92, 94, 95, 96, 134, 150, 152, 153, 155, 161, 197, 205, 208, 209, 212, 223, 226, 243, 248, 260, 265]	Accuracy, Hit@1, F1, BERT4Score, GPT4Score
		WebQ [8]	[11, 36, 83, 95, 160, 212, 230, 243]	
		CWQ [214]	[22, 28, 36, 57, 73, 83, 94, 95, 109, 125, 134, 150, 152, 153, 160, 161, 208, 212, 226, 248, 260, 265]	
		GrailQA [61]	[36, 94, 205, 212]	
		QALD10-en [184]	[125, 155, 212, 213]	
		SimpleQuestions [11]	[5, 11, 36, 212]	
		CMCQA	[230]	
		MetaQA [271]	[28, 65, 73, 92, 95, 104, 134, 137, 153, 160, 197, 208, 223, 243]	
		Mintaka [198]	[5–7, 125]	
		FreebaseQA [96]	[150, 260]	
	CSQA	CSQA [215]	[42, 53, 87, 121, 130, 217, 256]	Accuracy
		OBQA [162]	[42, 53, 71, 87, 121, 217, 256]	
		MedQA [101]	[53, 116, 217]	
		SocialIQA [194]	[87]	
		PIQA [9]	[87]	
		RiddleSenseQA [131]	[87]	
IE	EL	ZESHEL [149]	[241]	Recall@K
		CoNLL [78]	[241]	
	RE	T-Rex [47]	[212, 213]	Hit@1
		ZsRE [185]	[125, 155, 212, 213]	
Others	Fact Verification	Creak [172]	[125, 155, 212, 213]	Accuracy, F1
		FACTKG [105]	[104, 109, 137]	
	Link Prediction	FB15K-237 [218]	[29, 176]	MRR, Hits@K
		FB15k [10]	[29]	
		WN18RR [39]	[176]	
		NELL995 [15]	[29]	
	Dialogue Systems	OpenDialKG [164]	[5]	MRR, Hits@K
	Recommendation	Yelp	[126, 227]	NDCG@K, Recall@K

9.1.1 Question Answering. The QA tasks specifically include KBQA and **CommonSense Question Answering (CSQA)**.

(1) KBQA. KBQA [30] is a key downstream task for GraphRAG, where questions target KGs and answers involve entities, relations, or set operations. It evaluates structured knowledge retrieval and reasoning, critical for complex queries. GraphRAG improves multi-hop and abstract reasoning by integrating subgraph retrieval with generative models.

(2) CSQA. Unlike KBQA, CSQA [215] adopts a multiple-choice format, where each question presents several candidate answers—either entity names or statements. The task requires machines to retrieve relevant knowledge from commonsense KGs (e.g., ConceptNet) and perform reasoning to select the correct answer. GraphRAG enhances this process by leveraging meaningful paths from such KGs to improve complex question answering.

9.1.2 IR. IR tasks consist of two categories: EL and RE. GraphRAG augments information extraction by grounding text in KGs.

(1) EL. EL [202] identifies text mentions and links them to KG entities. GraphRAG advances EL by: (i) retrieving candidates via mention-context embeddings [171], (ii) disambiguating with graph-aware scoring [31], and (iii) enabling zero-shot linking using KG embeddings [21].

(2) RE. RE [168] identifies and classifies semantic relationships between entities in text. GraphRAG enhances RE by leveraging graph structures to model entity interdependencies, enabling more accurate and context-aware RE [125, 212, 213].

9.1.3 Others. In addition to the aforementioned downstream tasks, GraphRAG can be applied to various other tasks in the realm of NLP such as fact verification, link prediction, dialogue systems, and recommendation.

(1) Fact Verification. Fact verification evaluates the truthfulness of statements using KGs, where models validate factual claims against structured knowledge. GraphRAG improves this process by extracting evidential entity connections, boosting both accuracy and efficiency [72, 125, 186, 212, 213].

(2) Link Prediction. Link prediction identifies potential connections between entities in a graph. GraphRAG improves this task [29, 176] by retrieving and analyzing graph structures to uncover latent relationships, enhancing prediction accuracy.

(3) Dialogue Systems. Dialogue systems engage in natural language conversations, performing tasks like question answering and user assistance. GraphRAG [5] enhances these systems by modeling conversation histories and contextual relationships as graphs, improving response coherence and relevance.

(4) Recommendation. In the context of e-commerce platforms, the purchase relationships between users and products naturally form a network graph. The primary objective of recommendations within these platforms is to predict the future purchasing intentions of users, effectively forecasting the potential connections by leveraging GraphRAG for personalized retrieval [126, 227].

9.2 Application Domains

GraphRAG is widely applied in e-commerce [126, 227, 249], biomedical [38, 98, 116], code and software [117, 142, 175], academic literature [166, 231], legal [100], and other application scenarios [19, 183, 189] for its outstanding ability to integrate structured KGs with NLP, which will be introduced below.

9.2.1 E-Commerce. The primary goal in the e-commerce area involves improving customer shopping experiences and increasing sales through personalized recommendations and intelligent customer services. In this area, historical interactions between users and products can naturally form a graph, which implicitly encapsulates users' behavioral patterns and preference information. However, due to the increasing number of e-commerce platforms and the growing volume of user interaction data, using GraphRAG technology to extract key subgraphs is crucial. In e-commerce, recommendation serves as a core application scenario, primarily encompassing the following typical paradigms: general recommendation [75], sequential recommendation [102], review-based recommendation [112], explainable recommendation [119], and social recommendation [138]. GraphRAG demonstrates unique advantages by retrieving subgraphs or approximate graph patterns related to user-item interactions, which provides deeper semantic support for decisions. For instance, Wang et al. [227] ensemble multiple retrievers under different types or with different parameters to

extract relevant subgraphs, which are then encoded for temporal user action prediction. G-Refer [126] employs a hybrid graph retrieval approach that simultaneously retrieves structural and collaborative filtering information at both path and node levels to enhance explainable recommendation. Beyond recommendation, GraphRAG can also be effectively applied to customer service question answering scenarios. Xu et al. [249] construct a past-issue graph with intra-issue and inter-issue relations. For each given query, subgraphs of similar past issues are retrieved to enhance the system's response quality. These applications demonstrate GraphRAG's unique capability to uncover latent relationships in complex user-item interaction graphs, enabling more accurate, explainable, and context-aware solutions.

9.2.2 Biomedical. Recently, GraphRAG techniques have increasingly been applied in biomedical question answering systems, achieving advanced medical decision-making performance. In this area, each disease is associated with specific symptoms, and every medication contains certain active ingredients that target and treat particular diseases. Some researchers [38, 116, 238] construct KGs for specific task scenarios, while others [98, 235, 253] utilize open source KGs such as CMeKG and CPubMed-KG as retrieval sources. Existing methods generally begin with non-parametric retrievers for initial search, followed by designing methods to filter retrieved content through re-ranking [38, 98, 116, 235, 253]. Additionally, some approaches propose rewriting model inputs using retrieved information to enhance generation effectiveness [116]. These methods highlight GraphRAG's unique strength in transforming complex biomedical knowledge into actionable clinical insights through structured retrieval and evidence-based reasoning.

9.2.3 Code and Software. In software development, interactions such as code dependencies [117], API calls [167], third-party library integrations [113], and service composition [145] naturally form a complex structured knowledge network, which can be modeled as a code graph. GraphRAG leverages this structured representation to enhance code retrieval and generation capabilities, providing intelligent assistance to developers. When implementing specific functionalities, GraphRAG can analyze call chains and dependency relationships within the code graph to retrieve the most relevant functions, modules, or API usage examples, while generating context-aware code suggestions tailored to the project. For instance, CodeRAG [117] constructs a DS-Code Graph that models both dependency relationships and semantic relationships between codes. It then employs an agentic retrieval paradigm to search across the DS-Code Graph, identifying the most relevant source code, code snippets, and APIs for the requirement. GraphCoder [142] constructs a code context graph to capture the relationships among control flow and data dependence and employs two-step graph retrieval to enhance code completion. REPOGRPAH [175] proposes a repo-level graph, which models the relationships among each line of code. Additionally, Alhanahnah et al. [1] propose a GraphRAG-based Chatbot to understand properties about dependencies in a given software package. GraphRAG bridges code structure understanding with intelligent development assistance through graph-based retrieval.

9.2.4 Academic Literature. In the academic literature domain, each literature is authored by one or more researchers and is associated with a field of study. Authors are affiliated with institutions, and there exist relationships among authors, such as collaboration or shared institutional affiliations. These elements can be structured into a graph format. Utilizing GraphRAG on this graph can facilitate academic exploration, including literature mining, predicting potential collaborators for an author, and identifying trends within a specific field, and so on [100]. For instance, ATLANTIC [166] first constructs a document graph based on citation relationships between scientific papers, then employs a joint GNN-LM model for scientific QA and literature classification. While KGP [231] builds a KG from literature similarity and shared keyword relationships, subsequently performing

literature retrieval through iterative search. These approaches demonstrate how GraphRAG unlocks deeper insights into scholarly networks compared to traditional RAG methods, which often overlook relational structures.

9.2.5 Legal. In legal contexts, extensive citation connections exist between cases and judicial opinions, as judges frequently reference previous opinions when making new decisions. This naturally creates a structured graph where nodes represent opinions, opinion clusters, dockets, and courts, and edges encompass relationships such as "opinion-citation," "opinion-cluster," "cluster-docket," and "docket-court" [100]. The application of GraphRAG in legal scenarios could aid lawyers and legal researchers in various tasks such as case analysis and legal consultation. GraphRAG transforms complex legal citation networks into actionable insights, enhancing the efficiency and evidentiary support of legal research and case preparation.

9.2.6 Others. In addition to the above applications, GraphRAG is also applied to other real-world scenarios. For example, Ranade and Joshi [189] use GraphRAG for intelligence report generation, which first constructs an Event Plot Graph and retrieves the critical aspects of the events to aid the generation of intelligence reports. Peng and Yang [183] create a patent-phrase graph and retrieve the ego network of the given patent phrase to assist the judgment of phrase similarity. KAQG [19] applies GraphRAG to the education domain by constructing a KG based on difficulty levels and hierarchical relationships between knowledge points, thereby generating questions with controlled difficulty.

9.3 Benchmarks and Metrics

9.3.1 Benchmarks. The benchmarks used to evaluate the performance of the GraphRAG system can be divided into two categories. The first category is the corresponding datasets of downstream tasks. We summarize the benchmarks and papers tested with them according to the classification in Section 9.1, details of which are shown in Table 3. The second category consists of benchmarks specifically designed for the GraphRAG systems. These benchmarks usually cover multiple task domains to provide a comprehensive test result. For example, STARK [240] benchmarks LLM Retrieval on semi-structured knowledge bases covering three domains, including product search, academic paper search, and queries in precision medicine to access the capacity of current GraphRAG systems. He et al. [76] propose a flexible question-answering benchmark targeting real-world textual graphs, named GraphQA, which is applicable to multiple applications including scene graph understanding, commonsense reasoning, and KG reasoning. Graph Reasoning Benchmark [100] is constructed to facilitate the research of augmenting LLMs with graphs, which contains 1,740 questions that can be answered with the knowledge from 10 domain graphs. CRAG [254] provides a structured query dataset, with additional mock APIs to access information from underlying mock KGs to achieve fair comparison.

9.3.2 Metrics. The evaluation metrics for GraphRAG can be broadly categorized into two main types: downstream task evaluation (generation quality) and retrieval quality.

(1) Downstream Task Evaluation (Generation Quality). In the majority of research studies, downstream task evaluation metrics serve as the primary method for assessing GraphRAG's performance. For example, in KBQA, Hit@1, and F1-score are commonly used to measure the accuracy of answering entities. In addition, many researchers [100, 230, 235, 253] utilize BERT4Score and GPT4Score to mitigate instances where LLMs generate entities that are synonymous with the ground truth but not exact matches. In CSQA, accuracy is the most commonly used evaluation metric. For generative tasks like QA systems, some metrics, such as BLEU, ROUGE-L, and METEOR, are traditionally used

to evaluate the quality of the generated text [249, 253], while others rely on LLMs for assessment [45, 66].

(2) Retrieval Quality Evaluation. While evaluating GraphRAG based on downstream task performance is feasible, directly measuring the accuracy of retrieved content poses challenges. Therefore, many studies employ specific metrics to gauge the precision of retrieved content. For instance, when ground-truth entities are available, retrieval systems face a balance between the quantity of retrieved information and the coverage of answers. Hence, some studies use the recall of answers [120], while others utilize the ratio between answer coverage and the size of the retrieval subgraph to evaluate the performance of the retrieval system [265]. In addition, several studies have explored metrics such as query relevance, diversity, and faithfulness score to respectively assess the similarity between retrieved content and queries, the diversity of retrieved content, and the faithfulness of the information retrieved [166].

9.3.3 Limitations. Despite the existing metrics, evaluating GraphRAG systems comprehensively remains challenging due to several limitations inherent in current methodologies. These challenges are particularly pronounced in assessing the faithfulness of retrieval and the quality of generative outputs for open-ended tasks.

Lack of Standardized Retrieval-Faithfulness Metrics. Current evaluation primarily relies on downstream task performance (e.g., QA accuracy) as an indirect proxy for retrieval quality, lacking direct assessment of the relevance and completeness of retrieved graph elements. The difficulty lies in obtaining reliable ground truth for retrieval. Heuristic methods, such as using shortest paths between entities [192], are common but may introduce inaccuracies and redundancy. While newer benchmarks employ more precise techniques like tree search to construct reasoning paths [180], the existence of multiple valid paths for a single query remains a challenge. Developing benchmarks that can better capture the complex relationship between queries and their potential valid retrievals is crucial for faithful evaluation.

Position Bias in LLM-Based Evaluation for Generative Tasks. For open-ended tasks like summarization, LLM-as-a-judge [114] is increasingly adopted but suffers from position bias [70], where the order of output presentation significantly affects the evaluation result. This undermines the reliability of comparisons. Advancements in evaluation methods are needed, such as leveraging more robust judgment models trained to mitigate position bias, incorporating finer-grained evaluation criteria [118], and utilizing reasoning models that provide explicit chain-of-thought [233] to enhance judgment transparency.

9.4 GraphRAG in Industry

In this section, we mainly focus on industrial GraphRAG systems. These systems are characterized by their reliance on industrial graph database systems or their focus on large-scale graph data, details of which are as follows.

— *GraphRAG (by Microsoft)*[8]: The system uses LLMs to construct entity-based KGs and pre-generate community summaries of related entity groups, which enables the capture of both local and global relationships within a document collection, thereby enhancing QFS task [45]. The project can also utilize open source RAG toolkits for rapid implementation, such as LlamaIndex,[9] LangChain,[10] and so on.

[8] https://github.com/microsoft/graphrag.
[9] https://docs.llamaindex.ai/en/stable/examples/indexstructs/knowledgegraph/KnowledgeGraphDemo.html.
[10] https://python.langchain.com/docs/use_cases/graph.

— *Nano-GraphRAG*[11]: Nano-GraphRAG is a lightweight and flexible implementation of Microsoft GraphRAG, designed to simplify the usage and customization process of GraphRAG. It retains the core functionality of GraphRAG while offering a more streamlined code structure, a better user experience, and higher customizability.
— *GraphRAG (by NebulaGraph)*[12]: The project is the first industrial GraphRAG system, which is developed by NebulaGraph Corporation. The project integrates LLMs into the NebulaGraph database, which aims to deliver more intelligent and precise search results.
— *GraphRAG (by Antgroup)*[13]: The framework is developed on the foundation of several AI engineering frameworks such as DB-GPT, KG engine OpenSPG, and graph database TuGraph.
— *KAG (by Antgroup)*[14]: KAG is a logical reasoning QA framework based on the OpenSPG engine and LLMs, designed to build logical reasoning solutions for vertical domain knowledge bases. KAG supports features such as logical reasoning, multi-hop factual QA, and so on.
— *Fast-GraphRAG (by Circlemind)*[15]: Fast-GraphRAG is an agent-driven Fast-GraphRAG framework designed for interpretable, high-precision graph retrieval.
— *NallM (by Neo4j)*[16]: The NaLLM (Neo4j and LLMs) framework integrates Neo4j graph database technology with LLMs. It aims to explore and demonstrate the synergy between Neo4j and LLMs, focusing on three primary use cases: natural language interface to a KG, creating a KG from unstructured data, and generate reports using both static data and LLM data.
— *LLM Graph Builder (by Neo4j)*[17]: It is a project developed by Neo4j for automatically constructing KGs, suitable for the GraphRAG's Graph Database Construction and Indexing phase. The project primarily utilizes LLMs to extract nodes, relationships, and their properties from unstructured data and utilizes the LangChain framework to create structured KGs.

10 Future Prospects

While GraphRAG technology has made substantial strides, it continues to face enduring challenges that demand comprehensive exploration. This section will delve into the prevalent obstacles and outline prospective avenues for future research in the field of GraphRAG.

10.1 Dynamic and Adaptive Graphs

Most GraphRAG methods [45, 55, 107, 108, 152, 255] are built upon static databases; however, as time progresses, new entities and relationships inevitably emerge [27, 58, 242, 245, 246]. Rapidly updating these changes is both promising and challenging. Incorporating updated information is crucial for achieving better results and addressing emerging trends that require current data. Developing efficient methods for dynamic updates and real-time integration of new data will significantly enhance the effectiveness and relevance of GraphRAG systems.

10.2 Multi-Modality Information Integration

Most KGs primarily encompass textual information, thereby lacking the inclusion of other modalities such as images, audio, and videos, which hold the potential to significantly enhance the overall quality and richness of the database [234]. The incorporation of these diverse modalities could provide a more comprehensive and nuanced understanding of the stored knowledge. However,

[11] https://github.com/gusye1234/nano-graphrag.
[12] https://www.nebula-graph.io/posts/graph-RAG.
[13] https://github.com/eosphoros-ai/DB-GPT.
[14] https://github.com/OpenSPG/KAG.
[15] https://github.com/circlemind-ai/fast-graphrag.
[16] https://github.com/neo4j/NaLLM.
[17] https://github.com/neo4j-labs/llm-graph-builder.

the integration of such multi-modal data presents considerable challenges. As the volume of information increases, the graph's complexity and size grow exponentially, rendering it increasingly difficult to manage and maintain. This escalation in scale necessitates the development of advanced methodologies and sophisticated tools to efficiently handle and seamlessly integrate the diverse data types into the existing graph structure, ensuring both the accuracy and accessibility of the enriched KG.

10.3 Scalable and Efficient Retrieval Mechanisms

The application of GraphRAG in industrial scenarios faces significant scalability challenges. While current GraphRAG implementations demonstrate promising results on small-scale KGs (typically containing thousands of entities) [45], they encounter substantial difficulties when applied to real-world industrial KGs that often encompass millions to billions of entities. This scalability gap manifests in two critical aspects: computational complexity of graph traversal algorithms grows exponentially with graph size and retrieval latency increases dramatically, making real-time applications challenging. To address these limitations, future research should focus on developing hybrid or hierarchical retrieval architectures, as well as investigating distributed graph processing frameworks that can parallelize retrieval operations. Additionally, novel indexing structures specifically optimized for billion-scale KGs may need to be developed to enable efficient subgraph retrieval while maintaining the semantic richness that makes GraphRAG valuable.

10.4 Combination with Graph Foundation Model

Recently, graph foundation models [56, 154, 157], which can effectively address a wide range of graph tasks, have achieved significant success. Deploying these models to enhance the current GraphRAG pipeline is an essential problem. The input data for graph foundation models are inherently graph-structured, enabling them to handle such data more efficiently than LLM models. Integrating these advanced models into the GraphRAG framework could greatly improve the system's ability to process and utilize graph-structured information, thereby enhancing overall performance and capability.

10.5 Lossless Compression of Retrieved Context

In GraphRAG, the retrieved information is organized into a graph structure containing entities and their interrelations. This information is then transformed into a sequence that can be understood by LLMs, resulting in a very long context. There are two issues with inputting such long contexts: LLMs cannot handle very long sequences [82], and extensive computation during inference can be a hindrance for individuals. To address these problems, lossless compression of long contexts is crucial. This approach removes redundant information and compresses lengthy sentences into shorter, yet meaningful ones. It helps LLMs capture the essential parts of the context and accelerates inference. However, designing a lossless compression technique is challenging. Current works [55, 108] make a tradeoff between compression and preserving information. Promising solutions include: (1) text compression to retain only salient content while maintaining semantic coherence [91, 179] and (2) advanced graph pruning techniques [95, 150, 230] that eliminate redundant nodes/edges without breaking reasoning paths. The development of such techniques remains an open research challenge with substantial practical implications.

10.6 Standard Benchmarks

The absence of standardized and multifaceted benchmarks remains a significant impediment to the fair evaluation and comparable progress of GraphRAG systems. Most existing benchmarks are narrowly focused on static KGs [214, 259] and simple question answering [11], failing to

represent the diversity of real-world applications that involve dynamic [58], domain-specific [98, 166, 249], or text-rich graphs [100]. To enable more rigorous and holistic evaluation, future benchmark efforts could prioritize several key dimensions. First, expanding the variety of graph types and supporting multiple tasks could significantly enhance the scope and practical relevance of GraphRAG evaluations. Second, benchmarks should provide not only final answer annotations but also ground-truth retrieval elements to enable direct assessment of intermediate retrieval quality, separate from generator capabilities. Finally, for open-ended generation tasks, it is essential to establish robust LLM-as-a-judge [114] protocols incorporating position-bias-resistant judgment models, fine-grained evaluation criteria, and reasoning-aware protocols to ensure faithful, relevant, and verifiable output assessment. Such comprehensive benchmarks would drive the development of more generalizable, robust, and trustworthy GraphRAG systems.

10.7 Broader Applications

Current GraphRAG applications primarily focus on common tasks such as customer service systems [249], recommendation systems [37, 126], and KBQA [55]. Extending GraphRAG to broader applications involves incorporating more complex techniques across diverse domains. In financial services [3], GraphRAG can be utilized for fraud detection through time series analysis of transactional patterns [146], risk assessment via dynamic process discovery of market behaviors [14], and personalized financial advice by analyzing transactional data, market trends, and customer profiles. The automotive industry could leverage GraphRAG for autonomous driving [18, 133], where it could retrieve and reason over complex scene graphs to enhance real-time decision-making in dynamic environments. Industrial production could employ GraphRAG for risk detection by processing sensor network graphs and maintenance logs to predict potential failures [132]. Smart cities and IoT [206] applications present additional opportunities, where GraphRAG could analyze interconnected urban data graphs to optimize resource allocation and infrastructure management. Furthermore, GraphRAG could potentially be leveraged in reverse to enhance KGs, such as KG representation [129, 151, 204] and KG completion [115, 264]. Expanding GraphRAG to these diverse and complex domains will enhance its utility and impact, providing more sophisticated and targeted solutions across various fields.

11 Conclusion

In summary, this survey offers a comprehensive retrospective of GraphRAG technology, systematically categorizing and organizing its fundamental techniques, training methodologies, and application scenarios. GraphRAG significantly enhances the relevance, accuracy, and comprehensiveness of IR by leveraging pivotal relational knowledge derived from graph datasets, thereby addressing critical limitations associated with traditional RAG approaches. Furthermore, as GraphRAG represents a relatively nascent field of study, we delineate the benchmarks, analyze prevailing challenges, and illuminate prospective future research directions within this domain.

Acknowledgment

This work is supported by Ant Group through Ant Research Intern Program.

References

[1] Mohannad Alhanahnah, Yazan Boshmaf, and Benoit Baudry. 2024. DepsRAG: Towards managing software dependencies using large language models. arXiv:2405.20455. Retrieved from https://arxiv.org/abs/2405.20455

[2] Nicholas Alonso and Beren Millidge. 2024. Mixture-of-PageRanks: Replacing long-context with real-time, sparse GraphRAG. arXiv:2412.06078. Retrieved from https://arxiv.org/abs/2412.06078

[3] Muhammad Arslan and Christophe Cruz. 2024. Business-RAG: Information extraction for business insights. In *Proceedings of the 21st International Conference on Smart Business Technologies (ICSBT '24)*, 88–94.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference on the Semantic Web and 2nd Asian Semantic Web Conference (ISWC '07 + ASWC '07)*, Lecture Notes in Computer Science, Vol. 4825, 722–735.

[5] Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. 2023. Direct fact retrieval from knowledge graphs without entity linking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL '23)*, Long Papers, Vol. 1, 10038–10055.

[6] Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. arXiv:2306.04136. Retrieved from https://arxiv.org/abs/2306.04136

[7] Jinheon Baek, Soyeong Jeong, Minki Kang, Jong C. Park, and Sung Ju Hwang. 2023. Knowledge-augmented language model verification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP '23)*, 1720–1736.

[8] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, 1533–1544.

[9] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20), the 32nd Innovative Applications of Artificial Intelligence Conference (IAAI '20), the 10th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI '20)*, 7432–7439.

[10] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*, 1247–1250.

[11] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Largescale simple question answering with memory networks. arXiv:1506.02075. Retrieved from https://arxiv.org/abs/1506.02075

[12] William Brannon, Wonjune Kang, Suyash Fulay, Hang Jiang, Brandon Roy, Deb Roy, and Jad Kabbara. 2024. ConGraT: Self-supervised contrastive pretraining for joint graph and text embeddings. arXiv:2305.14321. Retrieved from https://arxiv.org/abs/2305.14321

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, Vol. 33, 1877–1901.

[14] Rujian Cai, Chao Zheng, Jian Wang, Duantengchuan Li, Chong Wang, and Bing Li. 2024. Reinforcement learning-based streaming process discovery under concept drift. In *Proceedings of the 36th International Conference on Advanced Information Systems Engineering (CAiSE '24)*, Lecture Notes in Computer Science, Vol. 14663, Springer, 55–70.

[15] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI '10)*, 1306–1313.

[16] Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen, and Huang, Yang Yang. 2023. GraphLLM: Boosting graph reasoning ability of large language model. arXiv:2310.05845. Retrieved from https://arxiv.org/abs/2310.05845

[17] Abir Chakraborty. 2024. Multihop question answering over knowledge graphs using large language models. arXiv:2404.19234. Retrieved from https://arxiv.org/abs/2404.19234

[18] Cheng Chang, Jingwei Ge, Jiazhe Guo, Zelin Guo, Binghong Jiang, and Li Li. 2025. Driving-RAG: Driving scenarios embedding, search, and RAG applications. arXiv:2504.04419. Retrieved from https://arxiv.org/abs/2504.04419

[19] Ching Han Chen and Ming Fang Shiu. 2025. KAQG A knowledge-graph-enhanced RAG for difficulty-controlled question generation. arXiv250507618. Retrieved from https://arxiv.org/abs/2505.07618

[20] Huajun Chen. 2024. Large knowledge model: Perspectives and challenges. arXiv231202706. Retrieved from https://arxiv.org/abs/2312.02706.

[21] Jiaoyan Chen, Yuxia Geng, Zhuo Chen, Jeff Z. Pan, Yuan He, Wen Zhang, Ian Horrocks, and Huajun Chen. 2023. Zero-shot and few-shot learning with knowledge graphs: A comprehensive survey. *Proceedings of the IEEE* 111, 6 (2023), 653–685.

[22] Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. arXiv:2410.23875. Retrieved from https://arxiv.org/abs/2410.23875

[23] Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. 2024. LLaGA: Large language and graph assistant. In *Proceedings of the 41st International Conference on Machine Learning (ICML '24)*.

[24] Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. ReTraCk: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the Joint Conference of the 59th*

Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL '21), 325–336.

[25] Weijie Chen, Ting Bai, Jinbo Su, Jian Luan, Wei Liu, and Chuan Shi. 2024. KG-retriever: Efficient knowledge indexing for retrieval-augmented large language models. arXiv:2412.05547. Retrieved from https://arxiv.org/abs/2412.05547

[26] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (LLMs) in learning on graphs. *ACM SIGKDD Explorations Newsletter* 25, 2 (2023), 42–61.

[27] Keyuan Cheng, Gang Lin, Haoyang Fei, Yuxuan Zhai, Lu Yu, Muhammad Asif Ali, Lijie Hu, and Di Wang. 2024. Multi-hop question answering under temporal knowledge editing. arXiv:2404.00492. Retrieved from https://arxiv.org/abs/2404.00492

[28] Hyeong Kyu Choi, Seunghun Lee, Jaewon Chu, and Hyunwoo J. Kim. 2023. NuTrea: Neural tree search for context-guided multi-hop KGQA. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023 (NeurIPS '23)*.

[29] Nurendra Choudhary and Chandan K. Reddy. 2024. Complex logical reasoning over knowledge graphs using large language models. arXiv:2305.01157. Retrieved from https://arxiv.org/abs/2305.01157

[30] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-Won Hwang, and Wei Wang. 2017. KBQA: Learning question answering over QA corpora and knowledge bases. *Proceedings of the VLDB Endowment* 10, 5 (2017), 565–576.

[31] Yuanfei Dai, Shiping Wang, Neal N. Xiong, and Wenzhong Guo. 2020. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics* 9, 5 (2020), 750.

[32] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *Proceedings of the 6th International Conference on Learning Representations (ICLR '18)*.

[33] DeepSeek-AI. 2024. DeepSeek-Coder-V2: Breaking the barrier of closed-source models in code intelligence. arXiv240611931. Retrieved from https://arxiv.org/abs/2406.11931.

[34] DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv250112948. Retrieved from https://arxiv.org/abs/2501.12948.

[35] DeepSeek-AI. 2025. DeepSeek-V3 technical report. arXiv241219437. Retrieved from https://arxiv.org/abs/2412.19437.

[36] Mohammad Dehghan, Mohammad Ali Alomrani, Sunyam Bagga, David Alfonso-Hermelo, Khalil Bibi, Abbas Ghaddar, Yingxue Zhang, Xiaoguang Li, Jianye Hao, Qun Liu, et al. 2024. EWEK-QA : Enhanced web and efficient knowledge graph retrieval for citation-based question answering systems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL '24)*, Long Papers, Vol. 1, 14169–14187.

[37] Yashar Deldjoo, Zhankui He, Julian J. McAuley, Anton Korikov, Scott Sanner, Arnau Ramisa, René Vidal, Maheswaran Sathiamoorthy, Atoosa Kasirzadeh, and Silvia Milano. 2024. A review of modern recommender systems using generative models (Gen-RecSys). In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*, 6448–6458.

[38] Julien Delile, Srayanta Mukherjee, Anton Van Pamel, and Leonid Zhukov. 2024. Graph-based retriever captures the long tail of biomedical knowledge. arXiv:2402.12352. Retrieved from https://arxiv.org/abs/2402.12352

[39] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI '18), the 30th Innovative Applications of Artificial Intelligence (IAAI '18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI '18)*, 1811–1818.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Long and Short Papers, Vol. 1, 4171–4186.

[41] Guanting Dong, Rumei Li, Sirui Wang, Yupeng Zhang, Yunsen Xian, and Weiran Xu. 2023. Bridging the KB-text gap: Leveraging structured knowledge-aware pre-training for KBQA. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*, 3854–3859.

[42] Junnan Dong, Qinggang Zhang, Xiao Huang, Keyu Duan, Qiaoyu Tan, and Zhimeng Jiang. 2023. Hierarchy-aware multi-hop question answering over knowledge graphs. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*. ACM, 2519–2527.

[43] Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. 2023. SimTeG: A frustratingly simple approach improves textual graph learning. arXiv:2308.02565. Retrieved from https://arxiv.org/abs/2308.02565

[44] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. arXiv:2407.21783. Retrieved from https://arxiv.org/abs/2407.21783

[45] 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:2404.16130. Retrieved from https://arxiv.org/abs/2404.16130

[46] Carl Edwards, ChengXiang Zhai, and Heng Ji. 2021. Text2Mol: Cross-modal molecule retrieval with natural language queries. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP '21)*, 595–607.

[47] Hady ElSahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon S. Hare, Frédérique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC '18)*.

[48] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on RAG meeting LLMs: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*, 6491–6501.

[49] Wenqi Fan, Shijie Wang, Jiani Huang, Zhikai Chen, Yu Song, Wenzhuo Tang, Haitao Mao, Hui Liu, Xiaorui Liu, Dawei Yin, et al. 2024. Graph machine learning in the era of large language models (LLMs). arXiv:2404.14928. Retrieved from https://arxiv.org/abs/2404.14928

[50] Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. 2024. DARA: Decomposition-alignment-reasoning autonomous language agent for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics (ACL '24)*. Association for Computational Linguistics, 3406–3432.

[51] Jinyuan Fang, Zaiqiao Meng, and Craig MacDonald. 2024. REANO: Optimising retrieval-augmented reader models through knowledge graph generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL '24)*, Long Papers, Vol. 1, 2094–2112.

[52] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*. OpenReview.net.

[53] Chao Feng, Xinyu Zhang, and Zichu Fei. 2023. Knowledge solver: Teaching LLMs to search for domain knowledge from knowledge graphs. arXiv:2309.03118. Retrieved from https://arxiv.org/abs/2309.03118

[54] Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP '20)*, 1295–1309.

[55] Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. 2020. A survey on complex question answering over knowledge base: Recent advances and challenges. arXiv:2007.13069. Retrieved from https://arxiv.org/abs/2007.13069

[56] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. 2024. Towards foundation models for knowledge graph reasoning. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*. OpenReview.net.

[57] Hanning Gao, Lingfei Wu, Po Hu, Zhihua Wei, Fangli Xu, and Bo Long. 2022. Graph-augmented learning to rank for querying large-scale knowledge graph. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (AACL/IJCNLP '22)*, Long Papers, Vol. 1, 82–92.

[58] Yifu Gao, Linbo Qiao, Zhigang Kan, Zhihua Wen, Yongquan He, and Dongsheng Li. 2024. Two-stage generative question answering on temporal knowledge graph using large language models. In *Findings of the Association for Computational Linguistics (ACL '24)*, 6719–6734.

[59] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. arXiv:2312.10997. Retrieved from https://arxiv.org/abs/2312.10997

[60] Aashish Ghimire, James Prather, and John Edwards. 2024. Generative AI in education: A study of educators' awareness, sentiments, and influencing factors. arXiv:2403.15586. Retrieved from https://arxiv.org/abs/2403.15586

[61] Yu Gu, Sue Kase, Michelle Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond I.I.D.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021 (WWW '21)*, 3477–3488.

[62] Yu Gu and Yu Su. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING '22)*. International Committee on Computational Linguistics, 1718–1731.

[63] Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. GPT4Graph: Can large language models understand graph structured data? An empirical evaluation and benchmarking. arXiv:2305.15066. Retrieved from https://arxiv.org/abs/2305.15066

[64] Kai Guo, Harry Shomer, Shenglai Zeng, Haoyu Han, Yu Wang, and Jiliang Tang. 2025. Empowering GraphRAG with knowledge filtering and integration. arXiv:2503.13804. Retrieved from https://arxiv.org/abs/2503.13804

[65] Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. 2024. KnowledgeNavigator: Leveraging large language models for enhanced reasoning over knowledge graph. arXiv:2312.15880. Retrieved from https://arxiv.org/abs/2312.15880

[66] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and fast retrieval-augmented generation. arXiv241005779. Retrieved from https://arxiv.org/abs/2410.05779

[67] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. arXiv240514831. Retrieved from https://arxiv.org/abs/2405.14831

[68] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to memory: Non-parametric continual learning for large language models. arXiv250214802. Retrieved from https://arxiv.org/abs/2502.14802

[69] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 1024–1034.

[70] Haoyu Han, Harry Shomer, Yu Wang, Yongjia Lei, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, and Jiliang Tang. 2025. RAG vs. GraphRAG: A systematic evaluation and key insights. arXiv:2502.11371. Retrieved from https://arxiv.org/abs/2502.11371

[71] Zhen Han, Yue Feng, and Mingming Sun. 2023. A graphguided reasoning approach for open-ended commonsense question answering. arXiv:2303.10395. Retrieved from https://arxiv.org/abs/2303.10395

[72] Ching Nam Hang, Pei-Duo Yu, and Chee Wei Tan. 2025. TrumorGPT: Graph-based retrieval-augmented large language model for fact-checking. *IEEE Transactions on Artificial Intelligence* 6, 11 (2025), 3148–3162.

[73] Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM '21)*, 553–561.

[74] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*.

[75] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*, 639–648.

[76] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. arXiv:2402.07630. Retrieved from https://arxiv.org/abs/2402.07630

[77] Michael Himsolt. 1996. *GML: Graph Modelling Language*. University of Passau.

[78] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, 782–792.

[79] Yubin Hong, Chaofan Li, Jingyi Zhang, and Yingxia Shao. 2025. FG-RAG: Enhancing query-focused summarization with context-aware fine-grained graph RAG. arXiv:2504.07103. Retrieved from https://arxiv.org/abs/2504.07103

[80] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. GRAG graph retrieval-augmented generation. arXiv240516506. Retrieved from https://arxiv.org/abs/2405.16506

[81] Yucheng Hu and Yuxing Lu. 2024. RAG and RAU: A survey on retrieval-augmented language model in natural language processing. arXiv:2404.19543. Retrieved from https://arxiv.org/abs/2404.19543

[82] Zhiyuan Hu, Yuliang Liu, Jinman Zhao, Suyuchen Wang, Yan Wang, Wei Shen, Qing Gu, Anh Tuan Luu, See-Kiong Ng, Zhiwei Jiang, et al. 2024. LongRecipe: Recipe for efficient long context generalization in large language models. arXiv:2409.00509. Retrieved from https://arxiv.org/abs/2409.00509

[83] Ziniu Hu, Yichong Xu, Wenhao Yu, Shuohang Wang, Ziyi Yang, Chenguang Zhu, Kai-Wei Chang, and Yizhou Sun. 2022. Empowering language models with knowledge graph reasoning for open-domain question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP '22)*, 9562–9581.

[84] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. arXiv:2311.05232. Retrieved from https://arxiv.org/abs/2311.05232

[85] Manzong Huang, Chenyang Bu, Yi He, and Xindong Wu. 2025. How to mitigate information loss in knowledge graphs for GraphRAG: Leveraging triple context restoration and query-driven feedback. arXiv:2501.15378. Retrieved from https://arxiv.org/abs/2501.15378

[86] Yizheng Huang and Jimmy Huang. 2024. A survey on retrieval-augmented text generation for large language models. arXiv:2404.10981. Retrieved from https://arxiv.org/abs/2404.10981

[87] Yongfeng Huang, Yanyang Li, Yichong Xu, Lin Zhang, Ruyi Gan, Jiaxing Zhang, and Liwei Wang. 2023. MVP-tuning: Multi-view knowledge retrieval with prompt tuning for commonsense reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*, Long Papers, Vol. 1, 13417–13432.

[88] Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. (Comet-) Atomic 2020: On symbolic and neural commonsense knowledge graphs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21), 33rd Conference on Innovative Applications of Artificial Intelligence (IAAI '21), the 11th Symposium on Educational Advances in Artificial Intelligence (EAAI '21)*, 6384–6392.

[89] Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume (EACL '21)*, 874–880.

[90] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. 2021. A survey on locality sensitive hashing algorithms and their applications. arXiv:2102.08942. Retrieved from https://arxiv.org/abs/2102.08942

[91] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP '23)*, 13358–13376.

[92] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP '23)*, 9237–9251.

[93] Jinhao Jiang, Kun Zhou, Ji-Rong Wen, and Wayne Xin Zhao. 2022. Great truths are always simple: A rather simple knowledge encoder for enhancing the commonsense reasoning capacity of pre-trained models. In *Findings of the Association for Computational Linguistics (NAACL '22)*, 1730–1741.

[94] Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. 2024. KG-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. arXiv:2402.11163. Retrieved from https://arxiv.org/abs/2402.11163

[95] Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023. UniKGQA: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *Proceedings of the 11th International Conference on Learning Representations (ICLR '23)*.

[96] Kelvin Jiang, Dekun Wu, and Hui Jiang. 2019. FreebaseQA: A new factoid QA data set matching trivia-style question-answer pairs with freebase. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '19)*, Long and Short Papers, Vol. 1, 318–323.

[97] Xinke Jiang, Rihong Qiu, Yongxin Xu, Wentao Zhang, Yichen Zhu, Ruizhe Zhang, Yuchen Fang, Xu Chu, Junfeng Zhao, and Yasha Wang. 2024. RAGraph: A general retrieval-augmented graph learning framework. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*.

[98] Xinke Jiang, Ruizhe Zhang, Yongxin Xu, Rihong Qiu, Yue Fang, Zhiyuan Wang, Jinyi Tang, Hongxin Ding, Xu Chu, Junfeng Zhao, et al. 2024. HyKGE A hypothesis knowledge graph enhanced framework for accurate and reliable medical LLMs responses. arXiv:2312.15883. Retrieved from https://arxiv.org/abs/2312.15883

[99] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs a comprehensive survey. arXiv:2312.02783. Retrieved from https://arxiv.org/abs/2312.02783

[100] Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, et al. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics (ACL '24)*, 163–184.

[101] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. What disease does this patient have? A largescale open domain question answering dataset from medical exams. arXiv:2009.13081. Retrieved from https://arxiv.org/abs/2009.13081

[102] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM '18)*, 197–206.

[103] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-Tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP '20)*, 6769–6781.

[104] Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics (EMNLP '23)*, 9410–9421.

[105] Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023. FactKG: Fact verification via reasoning on knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Long Papers, Vol. 1, 16190–16206.

[106] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR '17)*.

[107] Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: methods, challenges and solutions. In *Proceedings of 13th International Joint Conference on Artificial Intelligence (IJCAI '21)*, 4483–4491.

[108] Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 11 (2023), 11196–11215.

[109] Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL '20)*, 969–974.

[110] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP '21)*, 3045–3059.

[111] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-Tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020 (NeurIPS '20)*.

[112] Duantengchuan Li, Ceyu Deng, Xiaoguang Wang, Zhifei Li, Chao Zheng, Jing Wang, and Bing Li. 2024. Joint inter-word and inter-sentence multi-relation modeling for summary-based recommender system. *Information Processing and Management* 61, 2 (2024), 103631.

[113] Duantengchuan Li, Yuxuan Gao, Zhihao Wang, Hua Qiu, Pan Liu, Zhuoran Xiong, and Zilong Zhang. 2024. Homogeneous graph neural networks for third-party library recommendation. *Information Processing and Management* 61, 6 (2024), 103831.

[114] Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. 2025. From generation to judgment opportunities and challenges of LLM-as-a-judge. arXiv241116594. Retrieved from https://arxiv.org/abs/2411.16594

[115] Duantengchuan Li, Fobo Shi, Xiaoguang Wang, Chao Zheng, Yuefeng Cai, and Bing Li. 2024. Multi-perspective knowledge graph completion with global and interaction features. *Information Sciences* 666 (2024), 120438.

[116] Dawei Yang, Shu Tan, Zhen Baik, Jae Young Yun, Sukwon Lee, LiJoseph Chacko, Aaron Hou, Bojian Duong-Tran, Duy Ding, Ying Liu, et al. 2024. DALK dynamic co-augmentation of LLMs and KG to answer Alzheimer's disease questions with scientific literature. arXiv:2405.04819. Retrieved from https://arxiv.org/abs/2405.04819

[117] Jia Li, Xianjie Shi, Kechi Zhang, Lei Li, Ge Li, Zhengwei Tao, Jia Li, Fang Liu, Chongyang Tao, and Zhi Jin. 2025. CodeRAG: Supportive code retrieval on bigraph for real-world code generation. arXiv:2504.10046. Retrieved. Retrieved from https://arxiv.org/abs/2504.10046

[118] Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2024. Generative judge for evaluating alignment. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*. OpenReview.net.

[119] Lei Li, Yongfeng Zhang, and Li Chen. 2023. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems* 41, 4 (2023), 103:1–103:26.

[120] Mufei Li, Siqi Miao, Pan Li. 2024. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. arXiv:2410.20724. Retrieved from https://arxiv.org/abs/2410.20724

[121] Shiyang Li, Yifan Gao, Haoming Jiang, Qingyu Yin, Zheng Li, Xifeng Yan, Chao Zhang, and Bing Yin. 2023. Graph reasoning for question answering with triplet retrieval. In *Findings of the Association for Computational Linguistics (ACL '23)*, 3366–3375.

[122] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP '21)*, Long Papers, Vol. 1, 4582–4597.

[123] Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2024. A survey of graph meets large language model: Progress and future directions. arXiv:2311.12399. Retrieved from https://arxiv.org/abs/2311.12399

[124] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023. Large language models in finance: A survey. In *Proceedings of the 4th ACM International Conference on AI in Finance (ICAIF '23)*, 374–382.

[125] Yihao Li, Ru Zhang, and Jianyi Liu. 2024. An enhanced prompt-based LLM reasoning scheme via knowledge graph-integrated collaboration. In *Artificial Neural Networks and Machine Learning and 33rd International Conference on Artificial Neural Networks (ICANN '24)*, Lecture Notes in Computer Science, Vol. 15020, 251–265.

[126] Yuhan Li, Xinni Zhang, Linhao Luo, Heng Chang, Yuxiang Ren, Irwin King, and Jia Li. 2025. G-Refer: Graph retrieval-augmented large language model for explainable recommendation. In *Proceedings of the ACM on Web Conference 2025 (WWW '25)*, 240–251.

[127] Zhuoyang Li, Liran Deng, Hui Liu, Qiaoqiao Liu, and Junzhao Du. 2024. UniOQA A unified framework for knowledge graph question answering with large language models. arXiv:2406.02110. Retrieved from https://arxiv.org/abs/2406.02110

[128] Zijian Li, Qingyan Guo, Jiawei Shao, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. 2024. Graph neural network enhanced retrieval for question answering of LLMs. arXiv:2406.06572. Retrieved from https://arxiv.org/abs/2406.06572

[129] Zhifei Li, Qi Zhang, Fangfang Zhu, Duantengchuan Li, Chao Zheng, and Yan Zhang. 2023. Knowledge graph representation learning with simplifying hierarchical feature propagation. *Information Processing and Management* 60, 4 (2023), 103348.

[130] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP '19)*, 2829–2839.

[131] Bill Yuchen Lin, Ziyi Wu, Yichi Yang, Dong-Ho Lee, and Xiang Ren. 2021. RiddleSense: Reasoning about riddle questions featuring linguistic creativity and commonsense knowledge. In *Findings of the Association for Computational Linguistics (ACL/IJCNLP '21)*, 1504–1515.

[132] Ke Lin, Duantengchuan Li, Yanjie Li, Shiyu Chen, and Xindong Wu. 2024. FHCPL: An intelligent fixed-horizon constrained policy learning system for risk-sensitive industrial scenario. *IEEE Transactions on Industrial Informatics* 20, 4 (2024), 5794–5804.

[133] Ke Lin, Yanjie Li, Shiyu Chen, Duantengchuan Li, and Xinyu Wu. 2024. Motion planner with fixed-horizon constrained reinforcement learning for complex autonomous driving scenarios. *IEEE Transactions on Intelligent Vehicles* 9, 1 (2024), 1577–1588.

[134] Guangyi Liu, Yongqi Zhang, Yong Li, and Quanming Yao. 2024. Explore then determine: A GNN-LLM synergy framework for reasoning over knowledge graph. arXiv:2406.01145. Retrieved from https://arxiv.org/abs/2406.01145

[135] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2024. One for all: Towards training one graph model for all classification tasks. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*.

[136] H. Liu and P. Singh. 2004. ConceptNet-a practical commonsense reasoning tool-kit. *BT Technology Journal* 22, 4 (2004), 211–226.

[137] Haochen Liu, Song Wang, Yaochen Zhu, Yushun Dong, and Jundong Li. 2024. Knowledge graph-enhanced large language models via path selection. In *Findings of the Association for Computational Linguistics (ACL '24)*, 6311–6321.

[138] Hai Liu, Chao Zheng, Duantengchuan Li, Zhaoli Zhang, Ke Lin, Xiaoxuan Shen, Neal N. Xiong, and Jiazhang Wang. 2022. Multi-perspective social recommendation method with graph representation learning. *Neurocomputing* 468 (2022), 469–481.

[139] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S. Yu, et al. 2024. Towards graph foundation models: A survey and beyond. arXiv:2310.11829. Retrieved from https://arxiv.org/abs/2310.11829

[140] Lei Liu, Xiaoyan Yang, Junchi Lei, Xiaoyang Liu, Yue Shen, Zhiqiang Zhang, Peng Wei, Jinjie Gu, Zhixuan Chu, Zhan Qin, et al. 2024. A survey on medical large language models: Technology, application, trustworthiness, and future directions. arXiv:2406.03712. Retrieved from https://arxiv.org/abs/2406.03712

[141] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173.

[142] Wei Liu, Ailun Yu, Daoguang Zan, Bo Shen, Wei Zhang, Haiyan Zhao, Zhi Jin, and Qianxiang Wang. 2024. GraphCoder: Enhancing repository-level code completion via code context graph-based retrieval and language model. arXiv:2406.07003. Retrieved from https://arxiv.org/abs/2406.07003

[143] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL '22)*, Short Papers, Vol. 2, 61–68.

[144] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian. Zhilin Yang, and Jie Tang. 2023. GPT understands, too. arXiv:2103.10385. Retrieved from https://arxiv.org/abs/2103.10385

[145] Yongqiang Liu, Bing Li, Jian Wang, Duantengchuan Li, and Yutao Ma. 2022. Multi-information fusion based few-shot web service classification. *Future Generation Computer Systems* 130 (2022), 231–240.

[146] Yu Liu, Duantengchuan Li, Jian Wang, Bing Li, and Bo Hang. 2024. MDLR: A multi-task disentangled learning representations for unsupervised time series domain adaptation. *Information Processing & Management* 61, 2 (2024), 103638.

[147] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692. Retrieved from https://arxiv.org/abs/1907.11692

[148] Pei-Chi Lo and Ee-Peng Lim. 2023. Contextual path retrieval: A contextual entity relation embedding-based approach. *ACM Transactions on Information Systems* 41, 1 (2023), 1:1–1:38.

[149] Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL '19)*, Long Papers, Vol. 1, 3449–3460.

[150] Dan Luo, Jiawei Sheng, Hongbo Xu, Lihong Wang, and Bin Wang. 2023. Improving complex knowledge base question answering with relation-aware subgraph retrieval and reasoning network. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '23)*, 1–8.

[151] Haoran Luo, Haihong E., Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, et al. 2025. HyperGraphRAG: Retrieval-Augmented Generation via Hypergraph-Structured Knowledge Representation. arXiv:2503.21322. Retrieved from https://arxiv.org/abs/2503.21322.

[152] Haoran Luo, Haihong E., Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, et al. 2024. ChatKBQA: A Generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the Association for Computational Linguistics (ACL '24)*, 2039–2056.

[153] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*.

[154] Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. 2025. GFM-RAG: Graph foundation model for retrieval augmented generation. arXiv:2502.01113. Retrieved from https://arxiv.org/abs/2502.01113

[155] Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, and Jian Guo. 2024. Think-on-Graph 2.0: Deep and interpretable large language model reasoning with knowledge graph-guided retrieval. arXiv:2407.10805. Retrieved from https://arxiv.org/abs/2407.10805

[156] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. arXiv230514283. Retrieved from https://arxiv.org/abs/2305.14283

[157] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. 2024. Position: Graph foundation models are already here. In *Proceedings of the 41st International Conference on Machine Learning (ICML '24)*.

[158] Qiheng Mao, Zemin Liu, Chenghao Liu, Zhuo Li, and Jianling Sun. 2024. Advancing graph representation learning with large language models a comprehensive survey of techniques. arXiv:2402.05952. Retrieved from https://arxiv.org/abs/2402.05952

[159] Shengyu Mao, Yong Jiang, Boli Chen, Xiao Li, Peng Wang, Xinyu Wang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2024. RaFe: Ranking feedback improves query rewriting for RAG. arXiv:2405.14431. Retrieved from https://arxiv.org/abs/2405.14431

[160] Costas Mavromatis and George Karypis. 2022. ReaRev: Adaptive reasoning for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics (EMNLP '22)*, 2447–2458.

[161] Costas Mavromatis and George Karypis. 2024. GNNRAG: Graph neural retrieval for large language model reasoning. arXiv:2405.20139. Retrieved from https://arxiv.org/abs/2405.20139

[162] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2381–2391.

[163] Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP '16)*, 1400–1409.

[164] Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL '19)*, Long Papers, Vol. 1, 845–854.

[165] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TU-Dataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ '20)*.

[166] Sai Munikoti, Anurag Acharya, Sridevi Wagle, and Sameera Horawalavithana. 2023. ATLANTIC: Structure-aware retrieval-augmented language model for interdisciplinary science. arXiv:2311.12289. Retrieved from https://arxiv.org/abs/2311.12289

[167] Siyu Nan, Jian Wang, Neng Zhang, Duantengchuan Li, and Bing Li. 2025. DDASR: Deep diverse API sequence recommendation. *ACM Transactions on Software Engineering and Methodology* 34, 6 (2025), 162:1–162:39.

[168] Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. 2021. Named entity recognition and relation extraction: State-of-the-art. *ACM Computing Surveys* 54, 1 (2021), 1–39.

[169] Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M. Mulvey, H. Vincent Poor, Qingsong Wen, and Stefan Zohren. 2024. A survey of large language models for financial applications: Progress, prospects and challenges. arXiv:2406.11903. Retrieved from https://arxiv.org/abs/2406.11903

[170] Zhijie Nie, Zhangchi Feng, Mingxin Li, Cunwang Zhang, Yanzhao Zhang, Dingkun Long, and Richong Zhang. 2025. When text embedding meets large language model: A comprehensive survey. arXiv:2412.09165. Retrieved from https://arxiv.org/abs/2412.09165

[171] Byungkook Oh, Seungmin Seo, and Kyong-Ho Lee. 2018. Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 257–266.

[172] Yasumasa Onoe, Michael J. Q. Zhang, Eunsol Choi, and Greg Durrett. 2021. CREAK: A dataset for commonsense reasoning over entity knowledge. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1 (NeurIPS Datasets and Benchmarks '21)*.

[173] OpenAI. 2024. GPT-4 technical report. arXiv230308774. Retrieved from https://arxiv.org/abs/2303.08774.

[174] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022 (NeurIPS '22)*.

[175] Siru Ouyang, Wenhao Yu, Kaixin Ma, Zilin Xiao, Zhihan Zhang, Mengzhao Jia, Jiawei Han, Hongming Zhang, and Dong Yu. 2025. RepoGraph: Enhancing AI software engineering with repository-level code graph. In *Proceedings of the 13th International Conference on Learning Representations (ICLR '25)*.

[176] Vardaan Pahuja, Boshi Wang, Hugo Latapie, Jayanth Srinivasa, and Yu Su. 2023. A retrieve-and-read framework for knowledge graph link prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*, 1992–2002.

[177] Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, et al. 2023. Large language models and knowledge graphs: Opportunities and challenges. *Transactions on Graph Data and Knowledge* 1, 1 (2023), 2:1–2:38.

[178] Shirui Pan, Linhao Luo, Yufei Wang, Chen, Chen Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36, 7 (2024), 3580–3599.

[179] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics (ACL '24)*, 963–981.

[180] Boci Peng, Yongchao Liu, Xiaohe Bo, Jiaxin Guo, Yun Zhu, Xuanbo Fan, Chuntao Hong, and Yan Zhang. 2025. M^3GQA: A multi-entity multi-hop multi-setting graph question answering benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL '25)*, Long Papers, Vol. 1, Association for Computational Linguistics, 30594–30620.

[181] Boci Peng, Yongchao Liu, Xiaohe Bo, Sheng Tian, Baokun Wang, Chuntao Hong, and Yan Zhang. 2024. Subgraph retrieval enhanced by graph-text alignment for commonsense question answering. In *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference (ECML PKDD '24)*, 39–56.

[182] Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. 2024. Large language model based long-tail query rewriting in taobao search. In *Companion Proceedings of the ACM on Web Conference 2024 (WWW '24)*, 20–28.

[183] Zhuoyi Peng and Yi Yang. 2024. Connecting the dots: Inferring patent phrase similarity with retrieved phrase graphs. In *Findings of the Association for Computational Linguistics (NAACL '24)*, 1877–1890.

[184] Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. 2022. QALD-9-plus: A multilingual dataset for question answering over DBpedia and Wikidata translated by native speakers. In *Proceedings of the 16th IEEE International Conference on Semantic Computing (ICSC '22)*, 229–234.

[185] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick S. H. Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2021. KILT: A benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '21)*, 2523–2544.

[186] Zhixiao Qi, Yijiong Yu, Meiqi Tu, Junyi Tan, and Yongfeng Huang. 2023. FoodGPT: A large language model in food testing domain with incremental pre-training and knowledge graph prompt. arXiv:2308.10173. Retrieved from https://arxiv.org/abs/2308.10173

[187] Zile Qiao, Wei Ye, Yong Jiang, Tong Pengjun Xie, Weiping Fei Huang, and Shikun Zhang. 2024. MoLi supportiveness-based knowledge rewriting for retrieval-augmented language modeling. arXiv:2406.08116. Retrieved from https://arxiv.org/abs/2406.08116

[188] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* 21 (2020), 140:1–140:67.

[189] Priyanka Ranade and Anupam Joshi. 2023. FABULA: Intelligence report generation using retrieval-augmented narrative construction. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM '23)*, 603–610.

[190] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP '19)*, 3980–3990.

[191] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: towards deep graph convolutional networks on node classification. In *Proceedings of the 8th International Conference on Learning Representations (ICLR '20)*.

[192] Andrey Sakhovskiy, Mikhail Salnikov, Irina Nikishina, Aida Usmanova, Angelie Kraft, Cedric Möller, Debayan Banerjee, Junbo Huang, Longquan Jiang, Rana Abdullah, et al. 2024. TextGraphs 2024 shared task on text-graph representations for knowledge graph question answering. In *Proceedings of TextGraphs-17: Graph-Based Methods for Natural Language Processing*, 116–125.

[193] Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '19), the 31st Innovative Applications of Artificial Intelligence Conference (IAAI '19), the 9th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI '19)*, 3027–3035.

[194] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. SocialIQA: Commonsense reasoning about social interactions. arXiv:1904.09728. Retrieved from https://arxiv.org/abs/1904.09728

[195] Bhaskarjit Sarmah, Benika Hall, Rohan Rao, Sunil Patel, Stefano Pasquali, and Dhagash Mehta. 2024. HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. arXiv:2408.04948. Retrieved from https://arxiv.org/abs/2408.04948

[196] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*.

[197] Apoorv Saxena, Aditay Tripathi, and Partha P. Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL '20)*, 4498–4507.

[198] Priyanka Sen, Alham Fikri Aji, and Amir Saffari. 2022. Mintaka: A complex, natural, and multilingual dataset for end-to-end question answering. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING '22)*, 1604–1619.

[199] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics (EMNLP '23)*. Association for Computational Linguistics, 9248–9274.

[200] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv2402.03300. Retrieved from https://arxiv.org/abs/2402.03300

[201] Ahsan Shehzad, Feng Xia, Shagufta Abid, Ciyuan Peng, Shuo Dongyu Zhang, and Karin Verspoor. 2024. Yu graph transformers: A survey. arXiv:2407.09777. Retrieved from https://arxiv.org/abs/2407.09777

[202] Wei Shen, Jianyong Wang, and Jiawei Han. 2014. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering* 27, 2 (2014), 443–460.

[203] Zhili Shen, Chenxin Diao, Pavlos Vougiouklis, Pascual Merita, Shriram Piramanayagam, Damien Graux, Dandan Tu, Zeren Jiang, Ruofei Lai, Yang Ren, et al. 2024. GeAR: Graph-enhanced agent for retrieval-augmented generation. arXiv:2412.18431. Retrieved from https://arxiv.org/abs/2412.18431

[204] Fobo Shi, Duantengchuan Li, Xiaoguang Wang, Bing Li, and Xindong Wu. 2025. TGformer: A graph transformer framework for knowledge graph embedding. *IEEE Transactions on Knowledge and Data Engineering* 37, 1 (2025), 526–541.

[205] Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje F. Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge bases. arXiv:2210.12925. Retrieved from https://arxiv.org/abs/2210.12925

[206] S. Srivastava, M. D. Jain, H. Jain, K. Jaroli, V. J. Mayank Patel, and L. Khan. 2020. IOT monitoring bin for smart cities. In *3rd Smart Cities Symposium (SCS '20)*, Vol. 2020, IET, 533–536.

[207] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*, 697–706.

[208] Haitian Sun, Tania Bedrax-Weiss, and William W. Cohen. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP '19)*, 2380–2390.

[209] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4231–4242.

[210] Hao Sun, Yang Li, Liwei Deng, Bowen Li, Binyuan Hui, Binhua Li, Yunshi Lan, Yan Zhang, and Yongbin Li. 2023. History semantic graph enhanced conversational KBQA with temporal information modeling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL '23)*, Long Papers, Vol. 1, 3521–3533.

[211] Hao Sun, Xiao Liu, Yeyun Gong, Yan Zhang, Daxin Jiang, Linjun Yang, and Nan Duan. 2023. Allies: Prompting large language model with beam search. In *Findings of the Association for Computational Linguistics (EMNLP '23)*, 3794–3805.

[212] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*.

[213] Lei Sun, Zhengwei Tao, Youdi Li, and Hiroshi Arakawa. 2024. ODA: Observation-driven agent for integrating LLMs and knowledge graphs. In *Findings of the Association for Computational Linguistics (ACL '24)*, 7417–7431.

[214] Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '18)*, Long Papers, Vol. 1, 641–651.

[215] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '19)*, Long and Short Papers, Vol. 1, 4149–4158.

[216] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. GraphGPT: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*, 491–500.

[217] Dhaval Taunk, Lakshya Khanna, Siri Venkata Pavan Kumar Kandru, Vasudeva Varma, Charu Sharma, and Makarand Tapaswi. 2023. GrapeQA: GRaph augmentation and pruning to enhance question-answering. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23)*, 1138–1144.

[218] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP '15)*, 1499–1509.

[219] Hugo Touvron, Louis Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288. Retrieved from https://arxiv.org/abs/2307.09288

[220] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 5998–6008.

[221] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. arXiv:1710.10903. Retrieved from https://arxiv.org/abs/1710.10903

[222] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM* 57, 10 (2014), 78–85.

[223] Chaojie Wang, Yishi Xu, Zhong Peng, Chenxi Zhang, Bo Chen, Xinrun Wang, Lei Feng, and Bo An. 2023. keqing: Knowledge-based question answering is a nature chain-of-thought mentor of LLM. arXiv:2401.00426. Retrieved from https://arxiv.org/abs/2401.00426

[224] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023 (NeurIPS '23)*.

[225] Jinqiang Wang, Huansheng Ning, Yi Peng, Qikai Wei, Daniel Tesfai, Wenwei Mao, Tao Zhu, and Runhe Huang. 2024. A survey on large language models from general purpose to medical applications: Datasets, methodologies, and evaluations. arXiv:2406.10303. Retrieved from https://arxiv.org/abs/2406.10303

[226] Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-driven CoT: Exploring faithful reasoning in LLMs for knowledge-intensive question answering. arXiv:2308.13259. Retrieved from https://arxiv.org/abs/2308.13259

[227] Ruijie Wang, Zheng Li, Danqing Zhang, Qingyu Yin, Tong Zhao, Bing Yin, and Tarek F. Abdelzaher. 2022. RETE: Retrieval-enhanced temporal event forecasting on unified query product evolutionary graph. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*, 462–472.

[228] Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S. Yu, and Qingsong Wen. 2024. Large language models for education: A survey and outlook. arXiv:2403.18105. Retrieved from https://arxiv.org/abs/2403.18105

[229] Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. 2023. KnowledGPT: Enhancing large language models with retrieval and storage access on knowledge bases. arXiv:2308.11761. Retrieved from https://arxiv.org/abs/2308.11761

[230] Yuqi Wang, Boran Jiang, Yi Luo, Dawei He, Peng Cheng, and Liangcai Gao. 2024. Reasoning on efficient knowledge paths: Knowledge graph guides large language model for domain question answering. arXiv:2404.10384. Retrieved from https://arxiv.org/abs/2404.10384

[231] Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa F. Siu, Ruiyi Zhang, and Tyler Derr. 2024. Knowledge graph prompting for multi-document question answering. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI '24), 36th Conference on Innovative Applications of Artificial Intelligence (IAAI '24), 14th Symposium on Educational Advances in Artificial Intelligence (EAAI '14)*, 19206–19214.

[232] Yaoke Wang, Yun Zhu, Wenqiao Zhang, Yueting Zhuang, Liyunfei Liyunfei, and Siliang Tang. 2024. Bridging local details and global context in text-attributed graphs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP '24)*, 14830–14841.

[233] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed. H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022 (NeurIPS '22)*.

[234] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. 2019. MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19)*, 1437–1445.

[235] Yilin Wen, Zifeng Wang, and Jimeng Sun. 2024. MindMap: Knowledge graph prompting sparks graph of thoughts in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL '24)*, Long Papers, Vol. 1, 10370–10388.

[236] Zhihao Wen and Yuan Fang. 2023. Augmenting low-resource text classification with graph-grounded pre-training and prompting. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*, 506–516.

[237] Sondre Wold, Lilja Øvrelid, and Erik Velldal. 2023. TextTo-KG alignment: Comparing current methods on classification tasks. arXiv230602871. Retrieved from https://arxiv.org/abs/2306.02871

[238] Junde Wu, Jiayuan Zhu, and Yunli Qi. 2024. Medical graph RAG towards safe medical large language model via graph retrieval-augmented generation. arXiv:2408.04187. Retrieved from https://arxiv.org/abs/2408.04187

[239] Shangyu Wu, Ying Xiong, Yufei Cui, Haolun Wu, Can Chen, Ye Yuan, Lianming Huang, Xue Liu, Tei-Wei Kuo, Nan Guan, et al. 2024. Xue Retrieval-augmented generation for natural language processing: A survey. arXiv:2407.13193. Retrieved from https://arxiv.org/abs/2407.13193

[240] Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. 2024. STaRK: Benchmarking LLM retrieval on textual and relational knowledge bases. arXiv:2404.13207. Retrieved from https://arxiv.org/abs/2404.13207

[241] Taiqiang Wu, Xingyu Bai, Weigang Guo, Weijie Liu, Siheng Li, and Yujiu Yang. 2023. Modeling fine-grained information via knowledge-aware hierarchical graph for zero-shot entity retrieval. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining (WSDM '23)*, 1021–1029.

[242] Yuxia Wu, Yuan Fang, and Lizi Liao. 2024. Retrieval augmented generation for dynamic graph modeling. arXiv:2408.14523. Retrieved from https://arxiv.org/abs/2408.14523

[243] Yike Wu, Nan Hu, Sheng Bi, Guilin Qi, Jie Ren, Anhuan Xie, and Wei Song. 2023. Retrieve-rewrite-answer: A KG-to-text enhanced LLMs framework for knowledge graph question answering. arXiv:2309.11206. Retrieved from https://arxiv.org/abs/2309.11206

[244] Han Xie, Da Zheng, Jun Ma, Houyu Zhang, Vassilis N. Ioannidis, Xiang Song, Qing Ping, Sheng Wang, Carl Yang, Yi Xu, et al. 2023. Graph-aware language model pre-training on a large graph corpus can help multiple graph applications. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*, 5270–5281.

[245] Siheng Xiong, Yuan Yang, Faramarz Fekri, and James Clayton Kerce. 2023. TILP: Differentiable learning of temporal logical rules on knowledge graphs. In *Proceedings of the 11th International Conference on Learning Representations (ICLR '23)*.

[246] Siheng Xiong, Yuan Yang, Ali Payani, James Clayton Kerce, and Faramarz Fekri. 2024. TEILP: Time prediction over knowledge graphs via logical reasoning. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI '24), 36th Conference on Innovative Applications of Artificial Intelligence (IAAI '24), 14th Symposium on Educational Advances in Artificial Intelligence (EAAI '14)*, 16112–16119.

[247] Tianyang Xu, Haojie Zheng, Chengze Li, Haoxiang Chen, Yixin Liu, Ruoxi Chen, and Lichao Sun. 2025. NodeRAG: Structuring graph-based RAG with heterogeneous nodes. arXiv250411544. Retrieved from https://arxiv.org/abs/2504.11544

[248] Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Jun Zhao, and Kang Liu. 2024. Generate-on-graph: Treat LLM as both agent and KG for incomplete knowledge graph question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP '24)*, 18410–18430.

[249] Zhentao Xu, Mark Jerome Cruz, Matthew Guevara, Tie Wang, Manasi Deshpande, Xiaofeng Wang, and Zheng Li. 2024. Retrieval-augmented generation with knowledge graphs for customer service question answering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*, 2905–2909.

[250] Christos Xypolopoulos, Guokan Shang, Xiao Fei, Giannis Nikolentzos, Hadi Abdine, Iakovos Evdaimon, Michail Chatzianastasis, Giorgos Stamou, and Michalis Vazirgiannis. 2024. Graph linearization methods for reasoning on graphs with large language models. arXiv:2410.19494. Retrieved from https://arxiv.org/abs/2410.19494

[251] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 Technical report. arXiv:2505.09388. Retrieved from https://arxiv.org/abs/2505.09388

[252] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 Technical report. arXiv:2407.10671. Retrieved from https://arxiv.org/abs/2407.10671

[253] Rui Yang, Haoran Liu, Edison Marrese-Taylor, Qingcheng Zeng, Yuhe Ke, Wanxin Li, Lechao Cheng, Qingyu Chen, James Caverlee, Yutaka Matsuo, et al. 2024. KG-Rank: Enhancing large language models for medical QA with knowledge graphs and ranking techniques. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing (BioNLP@ACL '24)*, 155–166.

[254] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, et al. 2024. CRAG – comprehensive RAG benchmark. arXiv240604744. Retrieved from https://arxiv.org/abs/2406.04744

[255] Mohammad Yani and Adila Alfa Krisnadhi. 2021. Challenges, techniques, and trends of simple knowledge graph question answering: A survey. *Information* 12, 7 (2021), 271.

[256] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '21)*, 535–546.

[257] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: (EACL '24)*, 1955–1973.

[258] Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL '22)*, Long Papers, Vol. 1, 6032–6043.

[259] Wen-Tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL '16)*.

[260] Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *Proceedings of the 11th International Conference on Learning Representations (ICLR '23)*.

[261] Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2022. KG-FiD: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL '22)*, Long Papers, Vol. 1, 4961–4974.

[262] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of retrieval-augmented generation: A survey. arXiv:2405.07437. Retrieved from https://arxiv.org/abs/2405.07437

[263] Han-Cheng Yu, Yu-An Shih, Kin-Man Law, Kai-Yu Hsieh, Yu-Chen Cheng, Hsin-Chih Ho, Zih-An Lin, Wen-Chuan Hsu, and Yao-Chung Fan. 2024. Enhancing distractor generation for multiple-choice questions with retrieval augmented pretraining and knowledge graph integration. In *Findings of the Association for Computational Linguistics (ACL '24)*, 11019–11029.

[264] Hairong Zhang, Jiaheng Si, Guohang Yan, Boyuan Qi, Pinlong Cai, Song Mao, Ding Wang, and Botian Shi. 2025. RAKG: Document-level retrieval augmented knowledge graph construction. arXiv:2504.09823. Retrieved from https://arxiv.org/abs/2504.09823

[265] Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL '22)*, Long Papers, Vol. 1, 5773–5784.

[266] Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. 2024. GraphTranslator: Aligning graph model to large language model for open-ended tasks. In *Proceedings of the ACM on Web Conference 2024 (WWW '24)*, 1003–1014.

[267] Nan Zhang, Prafulla Kumar Choubey, Alexander Fabbri, Gabriel Bernadett-Shapiro, Rui Zhang, Prasenjit Mitra, Caiming Xiong, and Chien-Sheng Wu. 2024. SiReRAG: Indexing similar and related information for multihop reasoning. arXiv:2412.06206. Retrieved from https://arxiv.org/abs/2412.06206

[268] Qinggang Zhang, Junnan Dong, Hao Chen, Daochen Zha, Zailiang Yu, and Xiao Huang. 2024. KnowGPT: Knowledge graph based prompting for large language models. arXiv:2312.06185. Retrieved from https://arxiv.org/abs/2312.06185

[269] Taolin Zhang, Dongyang Li, Qizhou Chen, Chengyu Wang, Longtao Huang, Hui Xue, Xiaofeng He, and Jun Huang. 2024. R^4: Reinforced retriever-reorder-responder for retrieval-augmented large language models. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI '24) Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS '24)*, Frontiers in Artificial Intelligence and Applications, Vol. 392. IOS Press, 2314–2321.

[270] Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D. Manning, and Jure Leskovec. 2022. GreaseLM: Graph REASoning enhanced language models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR '22)*.

[271] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI '18), the 30th Innovative Applications of Artificial Intelligence (IAAI '18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI '18)*, 6069–6076.

[272] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. GraphText: Graph reasoning in text space. arXiv:2310.01089. Retrieved from https://arxiv.org/abs/2310.01089

[273] 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:2402.19473. Retrieved from https://arxiv.org/abs/2402.19473

[274] Yanxin Zheng, Wensheng Gan, Zefeng Chen, Zhenlian Qi, Qian Liang, and Philip S. Yu. 2024. Large language models for medicine: A survey. arXiv:2405.13055. Retrieved from https://arxiv.org/abs/2405.13055

[275] Yuqi Zhou, Sunhao Dai, Zhanshuo Cao, Xiao Zhang, and Jun Xu. 2024. Length-induced embedding collapse in transformer-based models. arXiv241024200. Retrieved from https://arxiv.org/abs/2410.24200

[276] Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Taotao Wang, Runyuan He, Yongwei Zhang, Sicong Liang, Xilin Liu, Yuchi Ma, et al. 2025. In-depth analysis of graph-based RAG in a unified framework. arXiv:2503.04338. Retrieved from https://arxiv.org/abs/2503.04338

[277] Yun Zhu, Haizhou Shi, Xiaotang Wang, Yongchao Liu, Yaoke Wang, Boci Peng, Chuntao Hong, and Siliang Tang. 2025. GraphCLIP: Enhancing transferability in graph foundation models for text-attributed graphs. In *Proceedings of the ACM on Web Conference 2025 (WWW '25)*, 2183–2197.

[278] Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. 2024. Efficient tuning and inference for large language models on textual graphs. arXiv:2401.15569. Retrieved from https://arxiv.org/abs/2401.15569

Received 11 January 2025; revised 7 September 2025; accepted 8 September 2025