

---

# Retrieval-Augmented Generation with Graphs (GraphRAG)

---

**Haoyu Han<sup>1,\*</sup>, Yu Wang<sup>2,\*</sup>, Harry Shomer<sup>1</sup>, Kai Guo<sup>1</sup>, Jiayuan Ding<sup>5</sup>, Yongjia Lei<sup>2</sup>, Mahantesh Halappanavar<sup>3</sup>, Ryan A. Rossi<sup>4</sup>, Subhabrata Mukherjee<sup>5</sup>, Xianfeng Tang<sup>6</sup>, Qi He<sup>6</sup>, Zhigang Hua<sup>7</sup>, Bo Long<sup>7</sup>, Tong Zhao<sup>8</sup>, Neil Shah<sup>8</sup>, Amin Javari<sup>9</sup>, Yinglong Xia<sup>7</sup>, Jiliang Tang<sup>1</sup>**

<sup>1</sup>Michigan State University, <sup>2</sup>University of Oregon, <sup>3</sup>Pacific Northwest National Laboratory

<sup>4</sup>Adobe Research, <sup>5</sup>Hippocratic AI, <sup>6</sup>Amazon, <sup>7</sup>Meta, <sup>8</sup>Snap Inc., <sup>9</sup>The Home Depot,

{hanhaoy1, shomerha, guokai1, tangjili}@msu.edu,

{yuwang, yongjia}@uoregon.edu, hala@pnnl.gov, ryanrossi@gmail.com,

{jiayuan, subho}@hippocraticai.com, {xianft, qih}@amazon.com,

{zhua, bolong, yxia}@meta.com, {tong, nshah}@snap.com, amin\_javari@homedepot.com

## Abstract

Retrieval-augmented generation (RAG) is a powerful technique that enhances downstream task execution by retrieving additional information, such as knowledge, skills, and tools from external sources. Graph, by its intrinsic "nodes connected by edges" nature, encodes massive heterogeneous and relational information, making it a golden resource for RAG in tremendous real-world applications. As a result, we have recently witnessed increasing attention on equipping RAG with Graph, i.e., GraphRAG. However, unlike conventional RAG, where the retriever, generator, and external data sources can be uniformly designed in the neural-embedding space, the uniqueness of graph-structured data, such as diverse-formatted and domain-specific relational knowledge, poses unique and significant challenges when designing GraphRAG for different domains. Given the broad applicability, the associated design challenges, and the recent surge in GraphRAG, a systematic and up-to-date survey of its key concepts and techniques is urgently desired. Following this motivation, we present a comprehensive and up-to-date survey on GraphRAG. Our survey first proposes a holistic GraphRAG framework by defining its key components, including query processor, retriever, organizer, generator, and data source. Furthermore, recognizing that graphs in different domains exhibit distinct relational patterns and require dedicated designs, we review GraphRAG techniques uniquely tailored to each domain. Finally, we discuss research challenges and brainstorm directions to inspire cross-disciplinary opportunities. Our survey repository is publicly maintained at <https://github.com/Graph-RAG/GraphRAG/>.

## 1 Introduction

Retrieval-Augmented Generation (RAG), as a powerful technique to improve downstream tasks by retrieving additional information from external data sources, has been successfully applied to various real-world applications [87, 120, 514, 551]. In RAG frameworks, retrievers search for additional knowledge, skills, and tools based on user-defined queries or task instructions. The retrieved content is then refined by an organizer and seamlessly integrated with the original query or instruction, which is further fed into the generator to produce the final answer. For example, when conducting question-answering (QA) tasks, the classic "Retriever-then-Reader" frameworks [191, 196, 468, 562] retrieve external factual knowledge to improve the answer faithfulness, which significantly benefit social goodness and mitigate risks in high-stake scenarios (e.g., medical, legal, financial, and education

---

\*Equal contribution.

consultation [467, 472, 515]). Moreover, recent advancements in large language models (LLMs) have further underscored the power of RAG in enhancing the social responsibility of LLMs, such as mitigating hallucinations [397], enhancing interpretability and transparency [203], enabling dynamic adaptability [360, 419], reducing privacy risks [512, 513], ensuring reliability/robust responses [105, 460], and promoting fair treatment [362].

Building on the unprecedented success of RAG and further considering the ubiquity of graphs in real-world applications [545], recent research has explored the integration of RAG with graph-structured data. Unlike textual or visual data, graph-structured data encodes heterogeneous and relational information through its intrinsic "nodes connected by edges" nature. For example, individuals connected by social relationships of social networks usually exhibit homophily behaviors [291], sequential decision-making steps in plans follow causal dependency [454], and atoms belonging to the same functional group within a molecule possess unique structural properties [103, 508]. Designing the RAG that utilizes relational information requires adapting its core components, such as the retriever and generator, to seamlessly integrate graph-structured data, resulting in GraphRAG. Different from RAG, which predominantly uses semantic/lexical similarity search [104, 120], GraphRAG offers unique advantages in capturing relational knowledge by leveraging graph-based machine learning (e.g., Graph Neural Networks (GNNs)) and graph/network analysis techniques (e.g., Graph Traversal Search and Community Detection [98, 428]). For example, considering the query "What drugs are used to treat epithelioid sarcoma and also affect the EZH2 gene product?" [452], blindly executing the existing BM25 or embedding-based search that relies solely on semantic/lexical similarity ignores relational knowledge encoded in graph structure. In contrast, some GraphRAG methods traverse the graph along the relational path "Disease (Epithelioid Sarcoma) → [indication] → Drug ← [target] ← Gene/Protein (EZH2 gene product)" to retrieve neighbors of Epithelioid Disease following the relation [indication], neighbors of Gene EZH2 following the relation [target], and find their intersected drug [186, 271, 428]. Moreover, some domains involve entities with extremely complex geometry that require dedicated model design to characterize. For example, 3D structures in molecular graphs [52, 445] and hierarchical tree structures commonly found in product taxonomies (e.g., on Amazon [529]), in document sections (e.g., when using Adobe Acrobat [537]), and social networks (e.g., at Snap [277]) requires carefully designed graph encoders (or, more precisely, geometric encoders) with appropriate expressiveness to capture structural nuances [277, 527]. Simply verbalizing node texts and feeding them into LLMs cannot express complex geometric information and becomes infeasible given the exponentially growing textual descriptions as neighborhood layers expand.

Despite the above advantages of GraphRAGs over RAGs, designing appropriate GraphRAGs faces unprecedented challenges due to the following differences in graph-structured data:

- **Difference 1 - Unified versus (vs.) Diverse-Formatted Information:** Unlike conventional RAG, where semantic information can be uniformly represented as a 2D grid of image patches or a 1D sequence of textual corpora, graph-structured data often encompass diverse formats and are stored in heterogeneous sources [4, 26, 434]. For example, document graphs embed entities as sentence chunks [98, 428], knowledge graphs store graph information as triplets or paths [38], and molecule graphs consist of higher-order structures (e.g., cellular complexes) [26], as shown in Figure 1. Some graph data may even be multimodal (e.g., text-attributed graphs include both structural and textual attributes, and scene graphs combine structures and vision). Consequently, this diversity necessitates different RAG designs. For retrievers, conventional RAG assumes the target information is indexed in an image or text corpus, which can be uniformly represented as vector embeddings and enable one-size-fits-all embedding-based retrieval. However, retrievers for GraphRAG must consider the concrete format and source of the desired information, making the one-size-fits-all design impractical. When dealing with knowledge graph question-answering, information of nodes, edges, or subgraphs is usually fetched by graph search before embedding matching-based retrieval [419, 492]. This fetching operation is usually conducted by identifying relevant nodes/edges/subgraphs via entity linking, relational matching, and graph search algorithms (e.g., Breadth-First Search, Depth-First Search, Monte Carlo Tree Search, and A\* search) [395, 419, 570], which is unachievable if solely through deep learning-based embedding similarity search. Furthermore, the design of the retriever should ensure sufficient geometric expressiveness to capture structural nuances. For instance, when retrieving APIs from a plan graph to accomplish specific goals [355, 356, 454], it is essential to equip the retriever with directional awareness. This enables the execution of APIs with resource dependencies in the correct order, preventing conflicts and avoiding invalid operations. Similarly, designing expressive retrievers capable of differentiating high-order subgraph structures, such as 6-cycle benzene versus vs. 4-

star Methane, and 3-star T-junction vs. 4-square road, is essential in drug design for disease treatment [139] and road construction for city planning [209]. Beyond the retriever, the generator also requires specialized designs. When retrieved content includes complex graph structures with textual attributes, simply verbalizing the text of the subgraph and concatenating it into a prompt may obscure critical structural information. In these cases, encoding the graph with graph encoders such as GNNs before integrating it into generation can help preserve structural nuances [134, 244, 434, 443, 456].

- **Difference 2 - Independent vs. Interdependent Information:** In conventional RAG, information is stored and utilized independently. For example, documents are split into chunks, such as individual sentences, paragraphs, or a fixed number of tokens, based on the document context and downstream task [21, 562]. Each chunk is then indexed and stored independently in a vector database. This independence prevents the retrieval from capturing chunk relations, which hinders performance on tasks requiring multi-hop reasoning and long-form planning. However, GraphRAG stores chunks as interconnected nodes with edges denoting their relations, which can benefit retrieval, organization, and generation. For retrieval, these edges could enable multi-hop traversal to capture other chunks that share a logical connection with existing retrieved chunks. Furthermore, the retrieved content can be organized not only by their semantic meaning (e.g., reranking [43, 172, 256]) but also their structural relations (e.g., graph pruning [377, 431]). During the generation phase, squeezing interdependency (e.g., positional encoding [361, 549]) to the generator would encode richer structural signals into the generated content.
- **Difference 3 - Domain Invariance vs. Domain-specific Information:** The relations in graph-structured data are domain-specific. Unlike images and texts, where different domains often share transferable semantics [254, 286], such as textures and grains in images or vocabulary defined by the tokenizer in texts, graph-structured data lacks explicit transferable units. This shared basis in images and texts lays the foundation for designing encoders with geometric invariance and enables the well-known data-scaling law. However, for graph-structured data, the underlying data generation process governing the generated graphs varies significantly across different domains. This variability makes the relational information highly domain-specific, and it is nearly impossible to design a unified GraphRAG applicable to different domains. For example, when predicting the topic of an academic paper, the widely accepted homophily assumption suggests retrieving references from the paper to inform its topic prediction [563]. However, this homophily assumption is not suitable when classifying the role of an airport in a flight network, where hubs are often sparsely distributed across a country with no direct connections [68]. Moreover, even within the same graph from the same domain, different tasks may necessitate distinct GraphRAG designs. For example, when designing an automatic email completion system to optimize communication efficiency in a company, both content relevance and tone coherence should be considered [429]. To ensure the content relevance of the generated emails, one might assume that close emails (i.e., emails from the same conversation thread) share similar content and thus should be retrieved for reference. However, to maintain tone coherence, emails from staff with similar roles might be retrieved, even if they do not share close social relations (e.g., between subordinates and superiors) but instead hold similar structural roles within the company (e.g., as managers of different teams).

Despite the above differences that have driven extensive research in GraphRAG, the current research landscape in this field remains fragmented, with significant variation in concepts, techniques, and datasets across studies. Moreover, current GraphRAG research primarily focuses on knowledge and document graphs as surveyed in Figure 2, often overlooking broader applications in other domains like infrastructure graphs. This imbalance not only hampers the advancement of GraphRAG but also risks creating a "bubble effect" that restricts the scope of future exploration. To address these challenges, we present a comprehensive and up-to-date review of GraphRAG, aiming to unify the GraphRAG framework from the global perspective while also specializing its unique design for each domain from the local perspective. The key contributions of this survey are as follows:

- **A Holistic Framework of GraphRAG:** We propose a holistic framework of GraphRAG consisting of five key components: query processor, retriever, organizer, generator, and graph data source. Within each component, we review representative GraphRAG techniques.
- **Specialization of GraphRAG in different domains:** We categorize GraphRAG designs into 10 distinct domains based on their specific applications, including knowledge graph , document graph , scientific graph , social graph , planning & reasoning graph , tabular graph .

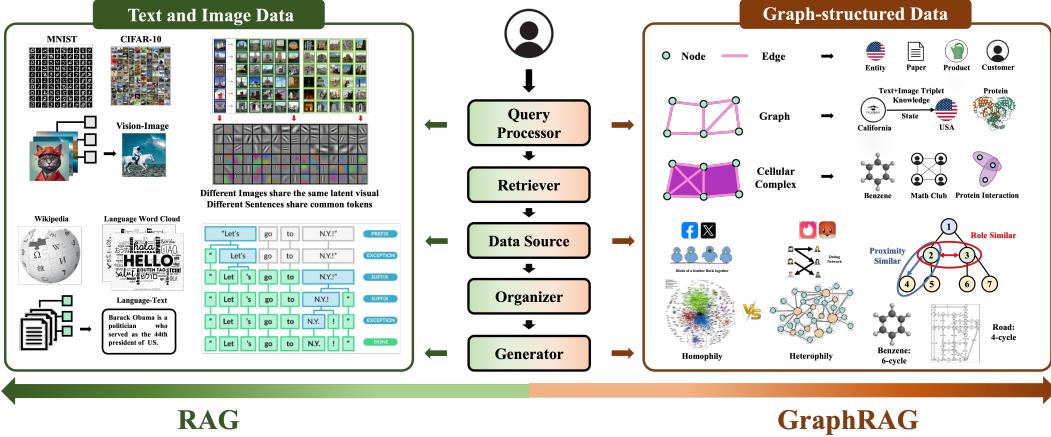


Figure 1: RAG works on text and images, which can be uniformly formatted as 1D sequences or 2D grids with no relational information. In contrast, GraphRAG works on graph-structured data, which encompasses diverse formats and includes domain-specific relational information.

infrastructure graph , biological graph , scene graph , and random graph . For each domain, we review their unique applications and specific graph construction methods. We then summarize the distinctive designs of each component within our proposed holistic GraphRAG framework and collect rich benchmark datasets and tool resources.

- **Challenges and Future Directions:** We highlight the challenges of current GraphRAG research and pinpoint future opportunities for further advancing GraphRAG into the new frontier.

In the following, we highlight the differences between our survey and existing surveys. Despite the urgent need for a systematic overview of GraphRAG, most existing surveys focus on general RAG within the context of i.i.d. data [11, 120, 227, 551, 561]. Before the advent of LLMs, earlier surveys focused on textual RAGs [11, 227]. With the recent unprecedented success achieved by foundational models such as LLMs, various surveys have explored foundational-model-powered RAG in different modalities. Gao et al. [120] group existing RAG approaches into three categories (Naive, Advanced, and Modular RAGs), summarize three core techniques (Retrieval, Generation, and Augmentation), and review evaluation metrics. In parallel, Zhao et al. [551] review representative RAG systems according to their corresponding application and data modality. [561] focuses on reviewing trustworthy concerns and techniques of RAG. However, none of them have a dedicated focus on graph-structured data. To the best of our knowledge, only one very recent study [319] has specifically surveyed RAG in the context of graph-structured data. However, this work mainly focuses on reviewing techniques introduced by graphs under the conventional RAG architecture without specializing in reviewing diverse relations and technical designs for graphs across different domains. In contrast to its holistic review philosophy, we recognize the inherent heterogeneity of graph-structured data and specialize our GraphRAG review across different domains. Specifically, we uncover the fundamental task applications (when to retrieve), graph construction methods and relational rationales (what to retrieve), and GraphRAG techniques (how to retrieve) for each domain. In this way, our survey provides a comprehensive overview of GraphRAG for information retrieval, data mining, and machine learning communities and domain-specific insights that facilitate interdisciplinary research and industrial opportunities.

Our survey is structured as follows: Section 2 introduces the holistic framework of GraphRAG and introduces representative techniques for its five key components. From Section 3 to 9, we delve into specific domains, reviewing unique task applications, summarizing existing graph construction methods that guide GraphRAG design for that domain, highlighting domain-specific techniques for each of the five components within our proposed holistic framework, and presenting existing GraphRAG resources (e.g., benchmark datasets and tools) used across different domains. Finally, we discuss research challenges and opportunities in Section 10 and conclude our survey in Section 11.

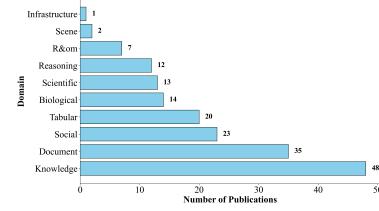


Figure 2: Publications of GraphRAGs in different domains based on surveyed papers

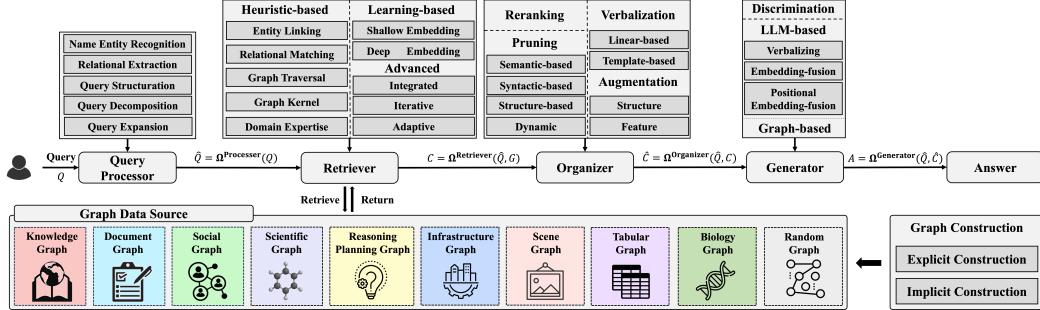


Figure 3: A holistic framework of GraphRAG and representative techniques for its key components.

## 2 A Holistic Framework of GraphRAG

Based on existing literature on GraphRAG, we present a holistic framework of GraphRAG. Next, we introduce the basic problem setting and notation used throughout the whole framework.

### 2.1 Problem Setting and Notations

Following the general setting of RAG, given a graph-structured data source  $G$ , the user-defined query  $Q$  is further sent to query processor  $\Omega^{\text{Processor}}$  to obtain the pre-processed query  $\hat{Q}$ . After that, the retriever  $\Omega^{\text{Retriever}}$  retrieves the content  $C$  from the graph data source  $G$  based on  $\hat{Q}$ . Next, the retrieved content  $C$  is refined by the organizer  $\Omega^{\text{Organizer}}$  to formulate the refined content  $\hat{C}$ . Finally, the refined content  $\hat{C}$  triggers the generator  $\Omega^{\text{Generator}}$  to generate the final answer  $A$ . The above five components are summarized as follows:

- **Query Processor  $\Omega^{\text{Processor}}$ :** Preprocessing the given query  $\hat{Q} = \Omega^{\text{Processor}}(Q)$ .
- **Graph Data Source  $G$ :** Information organized in graph-structured format.
- **Retriever  $\Omega^{\text{Retriever}}$ :** Retrieve the content  $C = \Omega^{\text{Retriever}}(\hat{Q}, G)$  from  $G$  based on the query  $\hat{Q}$ .
- **Organizer  $\Omega^{\text{Organizer}}$ :** Arrange and refine the retrieved content  $\hat{C} = \Omega^{\text{Organizer}}(\hat{Q}, C)$ .
- **Generator:** Generate answers  $A = \Omega^{\text{Generator}}(\hat{Q}, \hat{C})$  to answer query  $Q$ .

Unlike sequential-based textual data and grid-structured image data, graph-structured data encapsulates relational information. To effectively harness this relational information, the above five core components of GraphRAG desire dedicated designs to handle graph-structured input/output and support graph-based operations. For example, in the retriever component, conventional RAG in the Natural Language Processing (NLP) utilizes sparse/dense encoders for index search [196, 468]. In contrast, GraphRAG employs graph traversal methods (e.g., entity linking and BFS/DFS) and graph-based encoders (e.g., Graph Neural Networks (GNNs)) to produce embeddings for retrieval. This motivates us to summarize key innovations and representative designs of GraphRAG for each of the above five components under the holistic GraphRAG framework in the following.

### 2.2 Task Applications and Example Query $Q$

Similar to the general RAG framework where the text-formatted query  $Q$  specifies the question context or the task instruction. Query  $Q$  in GraphRAG could also be in the format of text. For example, in knowledge graph-based question-answering, the query could be "What is the Capital of China?" [272, 395]. In addition, the query could also be in other formats, such as smile strings for molecular graphs [132], or could even be the combination of multiple formats, such as the scene graph along with the text instruction [147]. Table 1 summarizes the common task applications and exemplary queries used in each domain, as well as their representative references.

Table 1: Summary of Task Applications and Exemplary Queries for GraphRAG in each domain.

Domain	Task Applications	Exemplary Query	References
<b>KG QA</b>		Where is the 16th President of the United States?	[492, 530, 493, 381, 110, 185, 78, 158, 522, 347, 272, [34, 456, 431, 380, 312, 308, 395, 181, 205, 118, 164, 72, 270, 406, 289, 251, 106, 452, 200, 443, 225, 244, 483, 232, 425, 182, 568] [62, 312, 389, 439], [246, 119] [566, 567] [200], [330]]
<b>Knowledge</b>	<b>KG Completion</b>	(Louvre, in, ?)	
	<b>Temporal QA</b>	Who was the president of the United States in 2006?	
	<b>Medical Prediction</b>	How long will the patient be spent in the hospital?	
	<b>Fact-Checking</b>	Who was the founder of Apple?	
	<b>Cyber Analysis and Defense</b>	Threat and vulnerability analysis	
<b>Document</b>	<b>Document Summarization</b>	Please summarize the following documents.	[491, 408, 237, 519, 548, 462, 547, 482, 231, 45, 525, 58] [429, 49, 212]
	<b>Text Generation</b>	Please generate an abstract based on this paper and its references.	[90, 501, 544, 249, 137]
	<b>Document Retrieval</b>	Find relevant documents related to Houston.	[538, 259, 518, 461]
	<b>Document Classification</b>	What is the category of the following document?	[307, 410, 147, 428, 108]
	<b>Document QA</b>	What are the main causes of climate change mentioned in the document?	[409, 302, 344, 63, 538, 543]
<b>Scientific</b>	<b>Relational Extraction</b>	What is the relation between 'Surfers Riverwalk' and 'Queensland'?	
	<b>Molecule Generation</b>	Given an input molecule, retrieve exemplar molecules to guide the generation process.	[434, 167]
	<b>Molecule Property Prediction</b>	Lumo is the lowest unoccupied molecular orbital energy. What's the Lumo value of this molecule?	[265]
	<b>Scientific QA</b>	After meals, I feel a bit of stomach reflux. What medication should I take for it?	[484, 185, 443, 225, 175, 449, 78, 318, 366, 449]
	<b>Entity Property Prediction</b>	Morality Assessment, Partisanship Detection	[183, 427]
<b>Social</b>	<b>Text Generation</b>	Review Generation, Social Post Generation	[5, 429, 202, 463, 353, 92, 315]
	<b>Recommendation</b>	Graph-based/Sequential/Conversational Recommendation	[95, 438, 427, 159, 510, 416, 327, 152, 116]
	<b>Social QA</b>	What are the best parks for familial gatherings around Los Gatos?	[511, 198, 452]
	<b>Fake News Detection</b>	The COVID-19 vaccine contains microchips for government tracking.	[331, 226]
	<b>Sequential Plan Retrieval</b>	Please generate an image where a girl is reading a book, and her pose is the same as the boy in "example.jpg"	[356, 355, 372, 454, 555]
<b>Reasoning &amp; Planning</b>	<b>Asynchronous Planning</b>	To make calzones, here are the steps and times required: please calculate the optimal plan for completion	[248]
	<b>Commonsense Reasoning</b>	Infer the stance and generate/retrieve the corresponding commonsense explanation graph	[342]
	<b>Declarative Inference</b>	Given a premise, a hypothesis may be weakened or overturned in light of new evidence	[282]
	<b>Tool Usage</b>	Take Shower -> Walkto(Bathroom) -> Walkto(shower) -> Find(shower) -> TurnTo(shower)	[570, 142]
	<b>Embodied Planning</b>	Cook a potato and put it into the recycle bin	[368, 477]
<b>Tabular</b>	<b>Cell Type Prediction</b>	What is the type of Cell B17?	[188]
	<b>Fraud Detection</b>	Is the following transaction legit or fraudulent?	[335, 364]
	<b>Outlier Detection</b>	Is the data normal or anomalous?	[124, 57, 560]
	<b>CTR Prediction</b>	Given the history and advertisement features, predict the probability that a user will click on a specific ad.	[242, 94, 135]
	<b>Data Imputation</b>	What is the missing value in cell B17?	[557, 497, 451, 341]
<b>Infrastructure</b>	<b>Table Type Classification</b>	What is the category of the table?	[432, 178, 188]
	<b>Table QA</b>	What is the average amount purchased and value purchased for the supplier who supplies the most products?	[532, 531]
	<b>Table Retrieval</b>	Retrieve tables containing information about the currencies of Asian countries	[411, 61]
	<b>Users Behavior Forecasting</b>	pedestrians' crossing actions; lane change maneuvers	[168]
	<b>Visual QA</b>	Is the data normal or anomalous?	
<b>Scene</b>	<b>Embodied Planning</b>	Given the history and advertisement features, predict the probability that a user will click on a specific ad.	
	<b>Gene Imputation</b>	Write 500-word advertisement for this place in the scene graph that would make people want to visit it	[147]
	<b>Clustering</b>	Cook a potato and put it into the recycle bin	[477]
	<b>Multi-omics Prediction</b>	What's the imputed gene value of the unsequenced gene KRAS for all cells?	[334, 161, 414, 469, 440]
	<b>Cell Type Deconvolution</b>	What's the cluster for cell B17?	[64, 502, 117, 60]
<b>Biological</b>	<b>Spatial Domain Identification</b>	What's the protein expression provided by single-cell RNA-seq expression for cell B17?	[84, 371]
	<b>Random</b>	What's the spatial domain for cell B17 in the spatial transcriptomics data?	[154, 91]
<b>Random</b>	<b>Graph Query</b>	What is the clustering coefficient of node P357	[109, 36, 13, 73, 413, 130, 275]

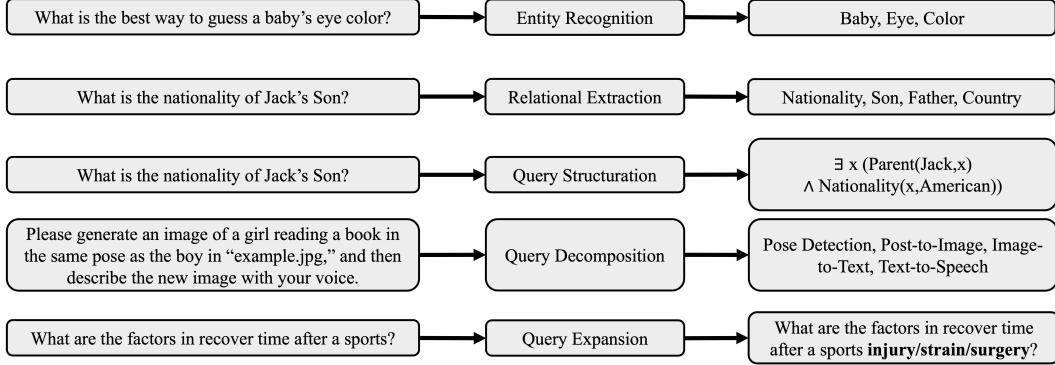


Figure 4: Existing techniques of query processor  $\Omega^{\text{Processor}}$  in GraphRAG.

Table 2: Difference of query processor  $\Omega^{\text{Processor}}$  between RAG and GraphRAG.

Technique	RAG	GraphRAG
Entity Recognition	Extracting mentions in knowledge bases	Extracting mentioned nodes in graphs.
Relational Extraction	Extracting textual relations	Extracting graph edge relations
Query Structuration	Structuring text query to SQL, SPARQL	Structuring text query to GQL
Query Decomposition	Decomposed queries are separate	Decomposed queries are logically related
Query Expansion	Expansion based on semantic knowledge	Expansion based on relational knowledge

### 2.3 Query Processor $\Omega^{\text{Processor}}$

Unlike RAG, where both queries and data sources are purely text-formatted, data sources used in GraphRAG are graph-structured, which raises challenges in bridging text-formatted queries and graph-structured data sources. For example, the information that connects the knowledge graph and the query "Who is Justin Bieber's brother?" is not a specific passage but instead the entity "Justin Bieber" and the relation "brother of". Many techniques are proposed to correctly extract this information from the query, including entity recognition, relational extraction, query structuration, query decomposition, and query expansion. In the following, we first review each of these five query processing techniques within the broader NLP domain, followed by a focused examination of their unique adaptations for GraphRAG.

#### 2.3.1 Name Entity Recognition

Named Entity Recognition (NER) aims to identify mentions of entities from the text that belong to predefined categories, such as persons, locations, or organizations, and it serves as a fundamental component for numerous natural language applications [160, 199, 228, 293]. NER techniques can be broadly categorized into four main approaches: (1) rule-based methods, which rely entirely on handcrafted rules and require no annotated data; (2) unsupervised learning methods, which use unsupervised algorithms without labeled training examples; (3) feature-based supervised learning methods, which depend on supervised algorithms and careful feature engineering; and (4) deep learning approaches, which automatically discover representations needed after (un)-supervised training the deep learning models. Recent LLMs fall into the category of deep learning approaches and have demonstrated unprecedented success for NER. More details about these techniques and their resources can be found in Li et al. [228].

Specifically, in the GraphRAG context, entity recognition primarily uses deep learning techniques (e.g., EntityLinker [395, 493] and LLM-based extraction [186]) to identify entities in queries grounded by nodes in the given graph data sources. This step is vital for applications such as knowledge graph-based question answering [395, 492, 493]. For example, given the question, "What is the best way to guess the color of the eye of the baby?", NER extracts entities such as "baby", "eye", and "color", which correspond to nodes in the knowledge graph and are treated as the seed nodes to initialize the retrieval process thereafter [443, 347]. For more recent GraphRAG research, NER has evolved beyond identifying the entity names but instead their structures. For example, Jin et al. [186] leverages LLMs to recognize node types in the graph, which further guides the retriever to identify nodes that

match the recognized types for next-round exploration. For example, given the question "Who are the authors of ‘Language Models are Unsupervised Multi-task Learners?’?" the initially recognized entity should not only be based on the semantic name "Language Models are Unsupervised Multi-task Learners" but also be based on the type of that entity, which is the paper node in this case. Accurately recognizing the names and structures of entities in GraphRAG reduces cascading errors and provides a solid foundation for subsequent retrieval and generation steps.

### 2.3.2 Relational Extraction

Similar to NER, relational extraction (RE) is a long-standing technique in NLP to identify relations among entities and is widely applied to structured search, sentiment analysis, question answering, summarization, and knowledge graph construction [47, 230, 303]. Recent advances in RE have been largely driven by deep learning techniques, and they can be summarized into three perspectives: text representation, context encoding, and triplet prediction, more details of which can be found in Pawar et al. [317], Nasar et al. [303], Han et al. [141].

For GraphRAG, RE serves two key purposes: constructing graph-structured data sources (e.g., knowledge graphs) by extracting triplets and matching the relations mentioned in the query and the graph data source to guide the graph search. For instance, given a query like "What is the capital of China?", relational extraction identifies the relation "capital of" and searches for corresponding edges via vector similarity in the knowledge graph, which guides the neighborhood selection and graph traversal direction [119, 200, 272, 273].

### 2.3.3 Query Structuration

Query structuration transforms queries into formats tailored to specific data sources and tasks. It often converts natural language queries into structured formats like SQL or SPARQL [181, 238] to interact with relational databases. Recent advancements leverage pre-trained and fine-tuned LLMs to generate structured queries from natural language input to query databases. For graph-structured data, Graph Query Language (GQL) has emerged, such as Cypher, GraphQL, and SPARQL, which enables complex interactions with property graph databases. Additionally, Jin et al. [186] introduced a technique that decomposes complex queries into multiple structured operations, including node retrieval, feature fetching, neighbor checks, and degree assessment, enhancing precision and adaptability in querying.

### 2.3.4 Query Decomposition

Query decomposition [447] aims to split the input query into multiple distinct subqueries, which are used to first retrieve sub-results and aggregate these sub-results together for the final results. In most existing RAG and GraphRAG, decomposed queries usually possess explicit logic connections that can handle complex tasks that require multistep reasoning and planning [248, 316, 355, 372, 477]. For example, a query like "Please generate an image where a girl is reading a book, and her pose is the same as the boy in ‘example.jpg’ then describe the new image with your voice" involves multiple subtasks [477], each of which would be completed by a specific sub-query. In addition, Park et al. [316] enhance the decomposition of the query by building a question graph where each sub-query is represented as a triplet within the graph. These graph-structured sub-queries effectively guide the retriever/generator through multi-step promptings.

### 2.3.5 Query Expansion

Query Expansion enriches a query by adding meaningful terms with similar significance [12], which primarily addresses three challenges: (1) user-submitted queries are ambiguous and relate to multiple topics; (2) queries may be too brief to fully capture user intent; and (3) users are often uncertain about what they are seeking. Generally, it can be categorized into manual query expansion, automatic query expansion, and interactive query expansion. More recently, LLM-based query expansion has been a prominent area due to the creativity of the generated content[54, 173, 221]

Unlike existing methods that mostly focus on textual similarities and overlook relations, QE in GraphRAG augments LLM expansion with structured relations. For example Xia et al. [459] expands the query by leveraging neighboring nodes of the mentioned entities in the query. Alternatively, Wang et al. [406] convert the query into several sub-queries using pre-defined templates.

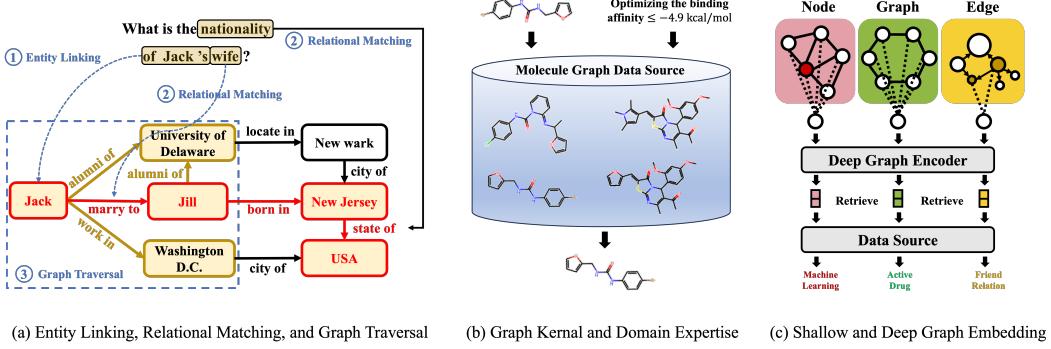


Figure 5: Visualizing representative retrievers used in GraphRAG.

Table 3: Categorizing representative retrievers used in GraphRAG.

Method/Strategy	Input	Output	Description
Entity Linking	Entity Mention	Node	Match query entity and graph node
Relational Matching	Relation Mention	Edge	Match query relation and graph edge
Graph Traversal	Node/Edge	Graph	Expand seed nodes/edges into subgraphs
Graph Kernel	(Sub)Graph	(Sub)Graph	Match query graph and candidate graph
Shallow Embedding	Any	Any	Embedding similarity match query and candidate
Deep Embedding	Any	Any	Embedding similarity match query and candidate
Domain Expertise	Expertise Rule	Any	Match Domain Expertise with nodes/edges/graphs

## 2.4 Retriever $\Omega^{\text{Retriever}}$

After obtaining the processed query  $\hat{Q}$ , the retriever  $\Omega^{\text{Retriever}}$  identifies and retrieves relevant content  $C$  from external graph sources  $G$  to augment the downstream task execution:

$$C = \Omega^{\text{Retriever}}(\hat{Q}, G) \quad (1)$$

Recently, retrievers have been increasingly integrated with LLMs to mitigate hallucination issues [397], address privacy concerns [513], and enhance explainability and dynamic adaptability [360, 419]. While effective, they are predominantly designed for texts and images and not readily transferable to graph-structured data for GraphRAG for two reasons. First, the input/output format of GraphRAG differs significantly from that of traditional RAG. While most retrievers in RAG use NLP tokenizers for encoders and adhere to the "Text-in, Text-out" workflow, the workflow of GraphRAG is more diverse, including "Text-in, Text-out" [395, 428], "Text-in, Graph-out" [454, 569], "Graph-in, Text-out" and "Graph-in, Graph-out" processes [433]. Secondly, retrievers in traditional RAGs do not capture graph structure signals. Methods like BM25 and TF-IDF [337, 333] primarily focus on lexical signals, and deep-learning-based retrievers [196] usually capture semantic signals, both of which overlook the graph structure signals. This motivates us to review existing GraphRAG retrievers, i.e., heuristic-based, learning-based, and domain-specific retrievers, with a particular emphasis on their unique technical design adapted to graph-structured data.

### 2.4.1 Heuristic-based Retriever

Heuristic-based retrievers primarily use predefined rules, domain-specific insights, and hard-coded algorithms to extract relevant information from graph data sources. Their reliance on explicit rules often makes them more time/resource-efficient compared to deep learning models. For instance, simple graph traversal methods like BFS or DFS can be executed in linear time without needing training data. However, this same reliance on fixed heuristics also limits their adaptability to generalize to unseen scenarios. In the following, we review the heuristic-based retrievers commonly used in GraphRAG.

**Entity Linking:** In heuristic-based retrievers, entity linking involves mapping entities identified in the query to corresponding nodes in graph data sources. This mapping forms an initial bridge between the query and the graph, serving as either the retriever by itself or as a foundation for further graph traversal to broaden the scope of the retrieval. The effectiveness of this approach relies on

accurate entity recognition conducted by the query processor and the quality of labeled entities on graph nodes. This technique is commonly applied in knowledge graphs, where Top-K nodes are selected as starting points based on their textual similarity to the query. The similarity metric can be computed using vector embeddings [443, 347] and lexical features [428]. More recently, LLMs have been used as knowledgeable context augmenters to generate mention-centered descriptions as additional input to augment the long-tail entities where their limited training data usually cause the entity linking model to struggle to disambiguate [464].

**Relational Matching:** Relational matching, similar to entity linking, is a heuristic-based retrieval approach designed to identify edges within graph data sources that align with the relations specified in a query. This method is crucial for tasks that focus on identifying relationships among entities in a graph. The matched edges guide the traversal process by indicating which edges to explore next based on the entities and relations encountered in the graph data sources. Similar to entity linking, Top-K edges are selected based on their similarity to each edge in the graph [200, 119].

In addition to the efficiency and simplicity of the above two types of heuristic-based retrievers, another key advantage is their ability to overcome ambiguity. For example, although machine/deep learning-based retrievers are difficult to differentiate semantically/lexically similar entities/relations (e.g., Byte vs. Bit, and President of vs. Resident of), these heuristic methods can easily distinguish them based on pre-defined rules, even in cases where semantic/lexical differences are subtle.

**Graph Traversal:** After performing entity linking and relational matching to identify initial nodes and relations in graph data sources, graph traversal algorithms (e.g., BFS, DFS) can expand this set to uncover additional query-relevant information. However, a core challenge for traversal-based retrieval is the risk of information overload, as the exponentially expanding neighborhood often includes substantial irrelevant content. To address this, current traversal techniques integrate adaptive retrieval and filtering processes, selectively exploring the most relevant neighboring nodes and incrementally refining the retrieved content to minimize noise. This graph traversal is mainly used in GraphRAG for knowledge and document graphs. When traversing on these two types of graphs, many methods extract all paths less than length  $l$  between the nodes identified by entity linking [492, 493, 530, 185, 110], while others consider the  $l$ -hop subgraph around the initial entities [308, 395, 181, 205]. To more efficiently traverse the KG, other methods prune irrelevant paths via the use of a LLM [271, 347, 428, 134] and others use pre-defined rules or templates to traverse the graph [406, 246, 72].

**Graph Kernel:** Compared with the above heuristic-based retrievers for retrieving nodes, edges, and their combined subgraphs, some earlier works (e.g., graph extraction and image retrieval) [448, 218, 123] treat the text and image as the entire graph and use graph-level heuristics such as graph kernels to measure similarity and retrieve. Graph kernels measure pairwise similarities by calculating inner products between graphs, aligning both structural and semantic aspects of the query and the retrieved graphs. Notable examples include the random-walk kernel and the Weisfeiler Leman kernel [357, 403]. The random walk kernel computes similarity by performing simultaneous random walks on two graphs and counting the number of matching paths. The Weisfeiler Leman kernel iteratively applies the Weisfeiler Leman algorithm to produce color distributions of node labels at each iteration and then calculates similarity based on the inner products of these histogram vectors. For example, Wu et al. [448] constructs event graphs of both documents and queries and uses a product graph kernel that counts walks between two graphs to measure the query-document similarity and rank the documents. Lebrun et al. [218] conducts event graph matching by introducing a fast and efficient graph-matching kernel for image retrieval. Similarly, Glavaš and Šnajder [123] translates images into representative attribute structural graphs that capture spatial relations among regions and perform graph kernel based on random walks to derive hash codes for image retrieval.

**Domain Expertise:** The domain-agnostic nature of traditional heuristic-based methods restricts their effectiveness in areas that require specialized expertise. For instance, in drug discovery, chemists typically design drugs by referencing existing molecules with desirable properties rather than constructing molecular structures from scratch. These molecules are selected based on domain knowledge that guides the retrieval of structures with similar characteristics. Following this intuition, many GraphRAG systems incorporate domain expertise to enhance retriever design. Wang et al. [434] develop a hybrid retrieval system that integrates heuristic-based and learning-based retrieval to retrieve exemplar molecules that partially meet the target design criteria.

### 2.4.2 Learning-based Retriever

One significant limitation of heuristic-based retrievers is their over-reliance on pre-defined rules, which limits their generalizability to data that does not strictly adhere to these rules. For example, when confronted with entities that have slight semantic or structural variations, such as "doctor" and "physician", heuristic-based retrievers like entity linking may treat them differently due to their distinct lexical representations, despite their shared underlying meaning. To overcome this limitation, learning-based retrievers have been proposed to capture deeper, more abstract, and task-relevant relations between the query and objects in data sources, which avoid relying solely on hard-coded rules. These retrievers often work by uniformly compressing information of various formats (e.g., texts and images) into embeddings based on machine learning encoders and then fetching relevant information by conducting an embedding-based similarity search. Notably, some entity linking and relational matching methods that employ machine learning encoders to generate embeddings for matching should also be considered as learning-based retrievers.

In conventional RAG, assuming the query  $q$  and data sources that contain  $n$  instances  $S$  are embedded by corresponding encoders as  $\mathbf{q} = \mathcal{F}_q(q) \in \mathbb{R}^d$  and  $\mathbf{S} = \mathcal{F}_S(S) \in \mathbb{R}^{n \times d}$ , we retrieve top- $k$  instances by similarity search according to the pre-defined similarity function  $\phi$  in the embedding space.

$$\mathcal{S}^* = \arg \max_k \phi(\mathbf{q}, \mathbf{S}), \quad (2)$$

Unlike RAGs that use language and vision encoders to embed texts and images, encoders used in GraphRAG retrieval extend beyond independently and identically distributed (i.i.d.) data by embedding nodes, edges, and (sub)graphs. Depending on the input format, the encoder could be a text encoder for query, a graph-based encoder for graph structure, and an integrated text-and-graph encoder for the textual attributed graph [55, 56]. We specifically focus on graph-based encoders. Existing graph-based encoders can be broadly categorized into shallow embedding methods – such as Node2Vec and DeepWalk – and deep embedding methods like Graph Neural Networks (GNNs). Below, we review these two encoders and their unique roles in GraphRAG.

**Shallow Embedding Methods:** Shallow embedding methods [114], like Node2Vec [127] and Role2Vec [5], learn node, edge, and graph embeddings that retain the essential structural information of the original graph. Based on the type of structural information that can be extracted, these methods generally fall into two categories: proximity/role-based embeddings. Proximity-based methods, such as DeepWalk and Node2Vec [127, 321], focus on preserving the proximity of connected nodes, ensuring that nodes close in the graph also remain close in the embedding space. Role-based methods, like Role2Vec and GraphWave [5, 93], generate node embeddings based on their structural roles rather than their proximity relations. In general, these methods initialize each node with a latent embedding vector and conduct unsupervised training to squeeze structural signals derived from graph structure into the embedding. In GraphRAG, proximity-based shallow embeddings can effectively retrieve entities that are proximally close, while role-based embeddings can capture entities that share similar roles. For instance, proximity-based embeddings could be used to retrieve academic papers by fetching papers sharing similar research topics or retrieve reviews from products that are co-purchased with the current product [346, 429]. Meanwhile, role-based embeddings could support tasks like generating company emails by retrieving similar emails based on shared roles or tones [429].

**Deep Embedding Methods:** Although shallow embedding methods incorporate structural signals into learned embeddings for nodes, edges, or entire graphs, they struggle to leverage semantic features—like bag-of-words representations for academic paper retrieval or atomic numbers for molecular retrieval [114]. Additionally, these methods lack inductivity, requiring re-initialization and retraining whenever new nodes, edges, or graphs are added. This limitation significantly reduces their applicability in GraphRAG retrieval tasks as real-world knowledge evolves dynamically where new information continually replaces outdated content, such as in citation networks, social graphs, and knowledge graphs [59, 419, 453]. To address these limitations, deep embedding methods have been proposed, which not only jointly fuse features and graph structures to obtain embeddings for retrieval but also inherent inductive property as the newly coming nodes/edges/graphs share common feature space with the ones during the training phase. One of the most representative and powerful approaches in this category is GNN, which combines the power of message-passing to encode structural signals and feature transformation to extract task-relevant information. Mathematically,  $l^{\text{th}}$ -layer graph convolution can be formulated as:

$$\mathbf{x}_i^l = \gamma_{\Theta_\gamma} \left( \mathbf{x}_i^{l-1} \oplus \sum_{j \in \mathcal{N}_i} \phi_{\Theta_\phi} (\mathbf{x}_i^{l-1}, \mathbf{x}_j^{l-1}, \mathbf{e}_{ij}) \right), \quad \text{Node-level} \quad (3)$$

$$\mathbf{e}_{ij}^l = \gamma_{\Theta_\gamma} \left( \mathbf{e}_{ij}^{l-1} \oplus \sum_{e_{mn} \in \mathcal{N}_{ij}^e} \phi_{\Theta_\phi} (\mathbf{e}_{ij}^{l-1}, \mathbf{e}_{mn}^{l-1}, \mathbf{x}_{e_{ij} \cap e_{mn}}) \right), \quad \text{Edge-level} \quad (4)$$

$$\mathbf{G}^l = \rho_{\Theta_{\rho_G}} (\{\mathbf{x}_i^l, \mathbf{e}_{ij}^l \mid v_i \in \mathcal{V}_G, e_{ij} \in \mathcal{E}_G\}), \quad \text{Graph-level} \quad (5)$$

In node-level graph convolution, each node  $v_i$  adaptively aggregates the embeddings of its neighboring nodes  $\mathcal{N}_i$ , with weights based on edge features via the weighting function  $\phi_{\Theta_\phi}$ . The aggregated neighborhood embeddings are then combined with the node's own embedding from the previous layer  $\mathbf{x}_i^{l-1}$ , using a combination function  $\gamma_{\Theta_\gamma}$ , as shown by Eq (3). Optimizing loss from training downstream tasks would enable the weighting function  $\phi_{\Theta_\phi}$  to prioritize the most important neighbors and enable the combination function  $\gamma_{\Theta_\gamma}$  to balance contributions from the node's neighborhood and its own embedding. Similarly, in edge-level graph convolution, the same aggregation principle applies, but the neighbors of an edge are edges incident to the same ending points of that edge  $\mathcal{N}_{ij}^e$ , as shown by Eq (4). Graph-level embeddings could be obtained by further applying pooling operation  $\rho_{\Theta_\rho}$  over node and edge embeddings, as shown by Eq (5). Following this GNN-based embedding paradigm, various forms of graph knowledge from diverse sources—such as nodes, edges, and (sub)graphs—can be uniformly embedded into vector representations, as shown in Figure 5(c) where we derive embeddings for nodes ( $\mathbf{X}$ ), edges ( $\mathbf{E}$ ), and graphs ( $\mathbf{G}$ ).

Having obtained these node/edge/graph-level embeddings further enables us to create embeddings for different types of structures ( $\mathbf{S}$ ) by combining these sub-structure embeddings according to specific configurations for each structure. For instance, if the retrieved subgraph is a path within a knowledge graph, we can aggregate the embeddings of the nodes and relations along that path to form a cohesive path embedding.<sup>2</sup> Eventually, the resulting embeddings for different structures can be utilized either during the training phase to optimize query alignment or during the testing phase to enable similarity-based neural search. For example, GNN-RAG [347] uses a GNN to perform retrieval, where a separate round of message passing is performed for each query. The query  $\hat{Q}$  is incorporated into the message passing by, including its embedding in the message computation. A set of “candidate” nodes is chosen which have a probability of being relevant greater than some threshold. The shortest path from the query nodes to each candidate node is retrieved as context. Liu et al. [251] consider the use of a conditional GNN [163] where only the linked entities from the query are initialized to a non-zero representation. The candidate nodes are chosen in a similar manner to [347]. A single path is then retrieved for each candidate node and is extracted by backtracking until we reach a query node. REANO [106] encodes the query information into an edge-specific attention weight, conditional on the query. After aggregation, the top k triples most similar to the query are chosen as context.

#### 2.4.3 Advanced Retrieval Strategies

Real-world queries are often complex and encode multi-aspect intentions, possess structure patterns, and desire multi-hop reasoning that the aforementioned basic retrievers struggle to address. For example, answering "What is the name of the fight song of the university whose main campus is in Lawrence, Kansas, and whose branch campuses are in the Kansas City metropolitan area?" demands multi-hop reasoning to identify the university based on location and retrieve information about its fight song [145, 399]. Similarly, a query like "What are the main themes in the dataset?" requires understanding the product community structure, retrieving themes for each community, and aggregating the identified themes together to summarize the main theme [98]. Furthermore, when asking "Who is the most impactful research scholar in deep learning?" the answer could vary depending on multiple aspects [536], such as the number of citations, the volume of published papers, or the number of co-authors. Accurately addressing such queries requires a deeper understanding of the underlying data distribution to discern which aspect the query prioritizes. To address these highly complex queries, advanced retrieval strategies have been proposed, and we review them as follows:

---

<sup>2</sup>Incorporating structural signals may be necessary, a consideration to be addressed in future work.

**Integrated Retrieval:** Integrated retrieval combines various types of retrievers to capture relevant information by balancing their strengths and weaknesses. Typically, integrated retrieval approaches are categorized according to which individual retrievers are used in combination, with notable examples including neural-symbolic retrieval [83, 220, 428] and multimodal retrieval [69, 266].

Since the knowledge stored in graph-structured data exists mostly in symbolic format, neural-symbolic retrieval is a natural choice for the integrated retrieval strategy in GraphRAG. This strategy interleaves rule-based patterns for retrieving symbolic knowledge with neural-based signals for retrieving more abstract and deep knowledge. For example, Luo et al. [272], Wen et al. [443] first expands the neighbors based on the knowledge of the symbolic knowledge graph and then performs path retrieval using neural matching. In contrast, Mavromatis and Karypis [289] first utilizes GNNs to retrieve seed entities (neural retrieval) and then extract the shortest paths from seed entities (symbolic retrieval). Similarly, Tian et al. [395], Yasunaga et al. [492, 493], Wang et al. [427], Luo et al. [272] fetch the k-hop neighborhood of the entities mentioned in the current question-answering pair and the session of user-generated items as the answer candidates (symbolic retrieval) and compute attention between the query and the extracted subgraph to differentiate candidate relevance (neural retrieval).

**Iterative Retrieval:** Iterative retrieval is a multistep process where consecutive retrieval operations share common dependencies such as causal, resource, and temporal dependency. These dependencies can be implicitly characterized by the retrieval order in RAG [399, 481] or explicitly modeled as a graph structure in GraphRAG [145, 454]. Consequently, iterative retrieval is primarily utilized in GraphRAG to capture these dependencies. For example, KGP [419] alternates between generating the next piece of evidence for the question and selecting the most promising neighbor. ToG [381] starts by identifying initial entities and then iteratively expands reasoning paths until enough information is gathered to answer the question. StructGPT [181] pre-defines graph interfaces and prompts LLMs to iteratively invoke these interfaces until sufficient information is collected.

**Adaptive Retrieval:** While retrieved external knowledge offers benefits, it also introduces risks. If the generator already possesses sufficient internal knowledge for a task, the retrieved external information may be unnecessary or even conflicting [42, 473]. Specifically, when internal knowledge fully covers the necessary information, retrieval becomes redundant and may introduce contradictions. To mitigate this, knowledge checking has been proposed in RAG systems [176, 189, 412, 490]. This approach allows the system to adaptively assess when and how much external information is needed. By equipping the retriever with this adaptability, RAG can provide more intelligent, flexible, and context-aware responses, fostering better harmony between internal and external knowledge sources.

One of the adaptive retrievals in GraphRAG is designed by considering different reasoning depths for different queries, i.e., too few hops of graph traversal might overlook critical reasoning relations, while too many can introduce unnecessary noise. Guo et al. [134], Wu et al. [455] address this by training models to predict the required number of hops for a given query and retrieving the relevant graph content accordingly. No existing works focus on resolving knowledge conflicts in GraphRAG, and therefore, we leave this discussion to future work.

## 2.5 Organizer

After retrieving the relevant content  $C$  from external graph data sources, which may be in the format of entities, relations, triplets, paths or subgraphs, the organizer  $\Omega^{\text{Organizer}}$  processes this content in conjunction with the processed query  $\hat{Q}$ . The aim is to post-process and refine the retrieved content to better adapt it for generator consumption, thereby further improving the quality of the downstream content generation. Formally, the organizer is represented as follows:

$$\hat{C} = \Omega^{\text{Organizer}}(\hat{Q}, C) \quad (6)$$

In GraphRAG, the need for fine-grained organization and refinement of retrieved content is driven by several key reasons. Firstly, when the retrieved contents are subgraphs, their heterogeneous format of knowledge in terms of node/edge features and graph structures becomes more likely to include irrelevant and noisy information, which poses significant difficulty for LLM to digest and thus compromises the generation quality. This raises the desire for graph pruning techniques to polish the retrieved subgraph and remove task-irrelevant knowledge. Secondly, LLMs have been widely demonstrated to possess attention biases toward certain positions of relevant information within the retrieved context [43]. Therefore, the exponentially growing neighbors as the receptive

field enlarges (i.e., the number of hops increases) in the retrieved subgraphs would also exponentially increase the amount of context length in the prompt and dilute the focus of LLMs on the task-relevant knowledge [358]. This poses a new requirement for the graph-based reranking mechanism to prioritize the most important content within the retrieved graph. Thirdly, the retrieved content might be incomplete in terms of both the semantic content and the structural content, which necessitates graph augmentation for the enhancement. Finally, the retrieved content is often a graph, which not only possesses semantic content information but also owns its unique structure. This complex structural content is not easily consumed by LLMs that are trained by next-token prediction coupled with linearized prompting, which requires structure-aware verbalization techniques to reorganize. We will formally review each of the above-motivated organizer techniques in the following sections.

### 2.5.1 Graph Pruning

In GraphRAG, the retrieved graph can be large and potentially contain a significant amount of noisy and redundant information. For example, when graph traversal methods are applied in retrieval, the size of the retrieved subgraph exponentially increases with the number of hops. Large subgraph sizes not only increase computational costs but can also reduce generation quality due to the inclusion of noisy information. In contrast, if the number of hops is too small, the retrieved subgraph may be too small to include crucial knowledge required by tasks. To achieve a better trade-off between the size of the retrieved subgraph and the amount of its encoded task-relevant information, various graph pruning methods have been proposed to reduce the size of subgraphs by removing irrelevant nodes and edges while preserving the essential information.

- **Semantic-based pruning:** Semantic-based pruning focuses on reducing the graph size by removing nodes and edge relations that are semantically irrelevant to the query. For example, QA-GNN [492] prunes irrelevant nodes with low relevance scores by encoding the query context and node labels using LLMs, followed by a linear projection. GraphQA [389] further removes clusters of nodes with the lowest relevance to the query. KnowledgeNavigator [133] scores the relations in the retrieved graph based on the query and prunes irrelevant relations to reduce graph size. Additionally, Gao et al. [118] partition the retrieved subgraph into smaller subgraphs and then ranks them with only the top-k smaller subgraphs retained for generations. G-Retriever [147] defines a semantic score for each retrieved node and edge, then refines the graph by solving the prize-collecting Steiner tree problem to construct a more compact and relevant subgraph.
- **Syntactic-based pruning:** Syntactic-based pruning removes irrelevant nodes from a syntactic perspective. For instance, Su et al. [377] leverages dependency analysis to generate a parsing tree of the context and then filters the retrieved nodes based on their span distance from the parsing tree.
- **Structure-based pruning:** Structure-based pruning methods focus on pruning the retrieved graph based on its structural properties. For example, RoK [431] filters out reasoning paths in the subgraph by calculating the average PageRank score for each path. Other works, such as Jiang et al. [181] and He et al. [144], also leverage PageRank to extract the most relevant entities.
- **Dynamic pruning:** Unlike the aforementioned methods, which typically prune the graph once, dynamic pruning removes noisy nodes dynamically during training. For example, JointLK [382] uses attention weights to recursively remove irrelevant nodes at each layer, keeping only a fixed ratio of nodes. Similarly, DHLK [430] filters out nodes with attention scores below a certain threshold dynamically during the learning process.

### 2.5.2 Reranker

The performance of LLMs can be influenced by the position of relevant information within the context, whether it appears at the beginning, middle, or end [43]. Additionally, LLMs' generation is impacted by the order in which in-context knowledge is provided, with later documents contributing less than earlier ones [172, 256]. While retrieved information is typically ordered by relevance scores during the retrieval process, these scores are often based on coarse-grained rankings across a large set of candidates. Enhancing reordering solely among the retrieved information at a fine-grained level, a process known as re-ranking, is essential to achieve optimal downstream performance. For example, Li et al. [234] rerank retrieved triples using a pre-trained cross-encoder. Jiang et al. [185] and Liu et al. [252] employ pre-trained reranker models to rerank retrieved paths. Yu et al. [498] train a GNN to rerank the retrieved passages. Liao et al. [246] order the paths by the time they occurred, giving more emphasis to recent paths.

### 2.5.3 Graph Augmentation

Graph augmentation aims to enrich the retrieved graph to either enhance the content or improve the robustness of the generator. This process can involve adding supplementary information to the retrieved graph, sourced from external data or knowledge embedded within LLMs. There are two main categories of methods:

- **Graph Structure Augmentation:** Graph structure augmentation methods involve adding new nodes and edges to the retrieved graph. For instance, GraphQA [389] augments the retrieved subgraph by incorporating noun phrase chunk nodes extracted from the context. Moreover, Yasunaga et al. [492] and Taunk et al. [389] treat the query as a node, integrating it into the retrieved graph to create direct connections between the query and relevant information. Tang et al. [388] augment the graph structure based on pretrained diffusion models.
- **Graph Feature Augmentation:** Graph feature augmentation methods focus on enriching the features of the nodes and edges in the graph. Since the original features might be lengthy or sparse, data augmenters can be employed to summarize or provide additional details for these features. For example, Once [258] uses LLMs as Content Summarizers, User Profilers, and Personalized Content Generators in recommendation systems. Similarly, LLM-Rec [276] and KAR [458] apply various prompting techniques to enrich node features, making them more informative for downstream tasks.

Additionally, some graph augmentation techniques focus solely on the retrieved graph itself, such as randomly dropping nodes, edges, or features to improve model robustness. Ding et al. [86] provide a systematic review of these data augmentation methods.

### 2.5.4 Verbalizing

Verbalizing refers to converting retrieved triples, paths or graphs into natural language that can be consumed by LLMs. There are two main approaches to verbalization: linear verbalization and model-based verbalization.

Linear verbalization methods typically convert graphs into text using predefined rules. The primary techniques for linear verbalization include:

- **Tuple-based:** These methods place the different pieces of retrieved information and order them in a tuple [14, 309]. For example, when performing retrieval on a KG, many methods retrieve a set of facts. A single fact is verbalized in the generation prompt as the tuple (entity 1, relation 1, entity 2) [308, 395]. For a set of facts, we first sort them in a specific order, and then verbalize them one at a time as an individual tuple. Each piece of information is typically separated by line in the prompt. Note that the same logic can be applied to paths, nodes, and so on.
- **Template-based:** These methods verbalize paths or graphs using predefined templates to generate more natural text. For example, LLaGA [49] proposes some templates such as Hop-Field Overview Template to convert graph into sequence. For KGs, several methods [134, 244] convert individual facts into natural text. For example, Guo et al. [134] convert a fact (entity 1, relation, entity 2) to text using the template “*The {relation} of {entity 1} is/are: {entity 2}*”.

Model-based verbalization methods typically use fine-tuned models or LLMs to convert input facts into coherent and natural language. These methods generally fall into two categories:

- **Graph-to-text verbalization:** These methods focus on converting retrieved graphs into natural language while preserving all the information. For instance, Koncel-Kedziorski et al. [211] and Wang et al. [421] leverage graph transformers to generate text from knowledge graph. Ribeiro et al. [336] evaluate several pretrained language models for graph-to-text generation, while Wu et al. [455], and Agarwal et al. [3] fine-tune LLMs to transform graphs into sentences, ensuring a faithful representation of the graph content in textual form.
- **Graph Summarization:** In contrast to Graph-to-Text Verbalization, which retains all details, Graph Summarization methods aim to generate concise summaries based on the retrieved graph and the query. EFSum [208] proposes two approaches: one directly prompts LLMs to summarize the retrieved facts and query, while the other fine-tunes LLMs specifically for summarization tasks. CoTKR [457], on the other hand, alternates between two operations: Reasoning, where it

decomposes the question, generates a reasoning trace, and identifies the specific knowledge needed for the current step; and Summarization, where it summarizes the relevant knowledge from the subgraph that retrieved based on the current reasoning trace.

## 2.6 Generator

The generator aims to produce the desired output for specific tasks based on the query and the retrieved information. These tasks can range from discrimination tasks (e.g., node/edge/graph classification) to generation tasks (e.g., KG-based question answering) and graph generation (e.g., molecular generation). Due to the uniqueness of different tasks, different generators are often desired. We categorize generators into three main types: Discriminative-based Generators, which leverage models like GNNs and Graph Transformers for tasks like classification; LLM-based Generators, which utilize the capabilities of LLMs to generate answers for text-based tasks; and Graph-based Generators, which generate new graphs using generative models such as diffusion models. Next, we provide a detailed illustration of these generators.

### 2.6.1 Discrimination-based Generator

Discrimination-based generators focus on discriminative and regression tasks, which can typically be modeled as graph tasks, such as node, edge, or graph classification and regression. Models designed for graph data, such as GNNs and Graph Transformers, are widely used as discrimination-based generators. The choice of GNN depends on the graph type and task. For instance, GCN [206], GraphSAGE [138], and GAT [402] are typically applied to homogeneous graphs, whereas models like RGCN [350] and HAN [423] are used for heterogeneous graphs, and HGNN [111] and Hyper-Attention [16] are suitable for hypergraphs. Additionally, graph transformers [296, 354] have gained popularity for their ability to capture global dependencies. Additionally, different training strategies, such as (semi-)supervised learning [279] and graph contrastive learning [192, 264], are employed depending on the specific requirements of the task.

### 2.6.2 LLM-based Generator

LLMs have demonstrated remarkable capabilities in understanding and generating natural language across a wide range of tasks. However, LLMs are inherently designed to process sequential data, while the retrieved information in GraphRAG is typically structured as graphs. Although various GraphRAG organizers, such as verbalization methods, convert retrieved graph information into text, these transformations may result in the loss of important graph structure information, which could be crucial for certain tasks. To take advantage of the ability of LLMs, many research efforts have been proposed to feed the graph information into LLMs, and we summarize them into the following categories:

- **Verbalizing:** Verbalizing aims to convert the retrieved information in GraphRAG into sequences that can be processed by LLMs. These methods are detailed in Section 2.5.4.
- **Embedding-fusion:** Embedding-fusion integrates graph embeddings and text embeddings within LLMs. The graph embeddings can be obtained using GNNs or Graph Transformers[36]. To align graph embeddings with text embeddings, a domain projector is typically learned to map graph embeddings to the text embedding space. Embedding fusion can occur at different layers of LLMs. For example, He et al. [147] feed the projected graph embeddings through the self-attention layers of LLMs, while Tian et al. [395] prepend the projected graph embeddings with the text tokens. [9] fuse the text and projected graph embeddings before the prediction layers of LLMs. Additionally, LLMs can either be fine-tuned along with the domain projector using methods such as LoRA, or the LLM can remain fixed, training only the graph embedding model and domain projector.
- **Positional embedding-fusion:** Directly converting the graph into sequences by Verbalization may lose graph structure information, which can be crucial in some tasks. Positional embedding-fusion aims to add the position of nodes in the retrieved graph to the LLMs. GIMLET[549], as a unified graph-text model, employs a generalized position embedding to encode both graph structures and textual instructions as unified tokens. LINKGPT [148] leverages the pairwise encoding in LPFormer [361] to encode the pairwise information between two nodes.

### 2.6.3 Graph-based Generator

In the scientific graph domain, GraphRAG generators often go beyond LLM-based methods due to the need for accurate structure generation. RetMol [433] is particularly versatile because it can work with various encoder and decoder architectures, supporting multiple generative models and molecule representations. For example, generators can be transformer-based or utilize Graph VAE architectures. Huang et al. [167] highlight the use of a diffusion model, specifically the 3D molecular diffusion model IRDIF. In the generation process, SE(3)-equivariance is achieved through architectures like Equivariant Graph Neural Networks (EGNNs) [349], which ensure that the geometric properties of molecular structures remain invariant to spatial transformations such as rotation, translation, and reflection at each step. Incorporating SE(3)-equivariance into the diffusion model guarantees that the generated molecular structures maintain geometric consistency under these transformations. For KGs, multiple works [110, 492, 389] use a GNN to generate the answer. The GNNs used in these works are conditional on the query, thereby making the final predictions relevant to it.

## 2.7 Graph Datasources

We have conducted a comprehensive review of the primary techniques applied in the initial four model-centric components of GraphRAG—namely, the query processor, retriever, organizer, and generator. However, even with the best configurations of these components, a GraphRAG system may still fall short of optimal performance if the underlying graph data sources, from which external knowledge is retrieved, are not meticulously curated. This also underscores the recent significant shift in AI research from a model-centric to a data-centric perspective, where enhancing data quality and relevance becomes equally, if not more, crucial for achieving superior results. Adopting this data-centric perspective, the following section provides an overview of existing GraphRAG research on constructing graph data sources from a high-level perspective, with a detailed discussion of domain-specific graph construction methods reserved for the subsequent domain-specific section.

- **Explicit Construction:** Explicit construction refers to building graphs based on explicit and predefined relationships in the data. This method is widely adopted across various domains. For example, molecule graphs are constructed from the connections between atoms; knowledge graphs are formed based on explicit relationships between entities; citation graphs are built by linking papers through citation relationships; and recommendation graphs model interactions between users and items.
- **Implicit Construction:** Implicit construction is used when there are no explicit relationships between nodes, but instead, implicit connections can be derived. For instance, word co-occurrence in a document can suggest shared semantic information, and feature interaction in Tabular data can indicate the correlation between features. Graphs can explicitly model these connections, which might be beneficial to the downstream tasks.

After the graph is constructed, there are also several ways to formally represent graphs.

- **Adjacency matrix:** The adjacency matrix is one of the most popular ways to denote a graph. Specifically, the adjacency matrix  $\mathbf{A} \in \mathbb{R}^{\|\mathcal{V}\| \times |\mathcal{V}|}$  denotes the graph connections among nodes in  $\mathcal{V}$ , where  $|\mathcal{V}|$  is the number of nodes.
- **Edge list:** The edge list represents each edge in the graph, typically in the form of tuples or triples, such as  $(i, j)$  or  $(i, r, j)$ , where  $i$  and  $j$  are nodes, and  $r$  is the relation between nodes  $i$  and  $j$ .
- **Adjacency list:** The adjacency list is a node-centric representation where each node is associated with a list of its neighbors. It is typically represented as a dictionary  $\{i : \mathcal{N}_i\}$ , where  $\mathcal{N}_i$  is the neighbor list of node  $i$ .
- **Node Sequence:** A node sequence transforms a graph into a sequence of nodes in either an irreversible or reversible manner. Most serialization methods are irreversible and do not allow for complete recovery of the original graph structure. For example, there are also some serialization methods that are reversible which can recover the whole graph structure. For example, Zhao et al. [552] propose serializing graphs using Eulerian paths by first applying Eulerization to the graph. Besides, if the graph establishes a tree structure, the BFS/DFS can also serialize the graph in a reversible manner.
- **Nature language:** With the growing popularity of LLMs for processing text-based information, various methods have been developed to describe graphs using natural language.

Note that the above-mentioned data structures can only represent basic graphs without support for complex scenarios such as multi-relational edges or edge attributes. For instance, using an adjacency matrix to represent a multi-relational attributed graph requires an expanded structure:  $\mathbf{A} \in \mathbb{R}^{|V| \times |V| \times |\mathcal{R}|}$ , where  $\mathcal{R}$  denotes the set of possible relationships. Here,  $\mathbf{A}_{i,j,r}$  represents the weight of the edge connecting node  $i$  and node  $j$  under relation  $r$ .

Selecting an appropriate graph representation is essential for task-specific requirements. For example, Ge et al. [122] finds that the order of graph descriptions significantly impacts LLMs' comprehension of graph structures and their performance across different tasks.

### 3 Knowledge Graph

A knowledge graph is a structured database that connects entities through well-defined relationships. It can either encompass a broad spectrum of general knowledge, such as the widely recognized Google Knowledge Graph [53, 261], or delve deeply into specialized domains, like the BioASQ dataset [400] for biomedical reasoning. The diverse information contained in a knowledge graph – represented as entities, relationships, paths, and subgraphs – serves as a valuable resource for enhancing various downstream tasks across different sectors, including question-answering [395, 428, 493], commonsense reasoning [169], fact-checking [201], recommender systems [131], drug discovery [28], healthcare [37], and fraud detection [287].

#### 3.1 Application Tasks

This section reviews representative applications that GraphRAG on KGs is used for.

- **Question-answering:** Question-answering (QA) can focus on a single domain or span across global knowledge. Typically, a query in text format is given, such as "What is the best way to predict a baby's eye color?" or "Were there fossil fuels in the ground when humans evolved?" [395] – the answer can be a sentence generated by a large language model (LLM), a selected text span from relevant documents, or even a specific choice in a multiple-choice QA scenario. In all these contexts, GraphRAG leverages knowledge graphs to retrieve relevant information, providing the necessary context or supporting facts to generate accurate answers.
- **Fact-Checking:** Fact-checking is to verify the truthfulness of statements by cross-referencing them with reliable sources of information. GraphRAG enhances this task by querying a knowledge graph to retrieve relevant facts and relational structures that either support or refute the given claim. GraphRAG identifies discrepancies or confirmations within the data by mapping the statement onto the knowledge graph, providing a thorough and evidence-based validation process.
- **Knowledge Graph Completion:** Knowledge graph completion is the task of predicting new facts to enhance the comprehensiveness of the graph and infer missing facts [535]. GraphRAG addresses this task by retrieving structural knowledge around the triplets for inference, supplying essential structural knowledge, and enhancing the LLM inference.
- **Cybersecurity Analysis and Defense:** Cybersecurity Analysis and Defense aims to analyze and respond to vulnerabilities, weaknesses, attack patterns, and threat tactics. With the increasing complexity and volume of cybersecurity data, GraphRAG has been proposed to provide cybersecurity analysis with more comprehensive insights into potential attack vectors and mitigation strategies [330].

#### 3.2 Knowledge Graph Construction

We discuss how KGs are typically constructed. For each type of construction technique, we give examples of common KG databases. How a KG is constructed is important, as it can affect both its usefulness and function in different downstream tasks. We describe the main techniques below:

- **Manual construction:** Some KGs are constructed manually via human annotation. WikiData [404] is a KG that uses crowd-sourced efforts to gather a variety of knowledge. Each entity corresponds to a page in the Wikipedia encyclopedia. Another KG, the Unified Medical Language System (UMLS) [25], contains biomedical facts collected from numerous sources.

- **Rule-based construction:** Many traditional approaches use rule-based techniques for constructing the graph. This takes the form of custom parsers and manually defined rules used to extract facts from raw text. Note that these parsers can differ depending on the source of the text. Prominent examples include ConceptNet [373], which links different words together via assertions, and Freebase [27] which contains a wide variety of general facts. REANO [106] extracts the entities and relations from a set of passages using traditional entity recognition (ER) and relation extraction (RE) methods, respectively. This includes SpaCy [151] for ER and TAGME [112] for RE. To extract the facts that connect two entities via a relation, they use DocuNet [526].
- **LLM-based construction:** Recently, work has explored how LLMs can be used to construct KGs from a set of documents. In such a way, the LLM can automatically extract the entities and relations and link those together to form facts in the given text. Of note is that no ground-truth KG exists for these methods. Rather, they simply use a KG as a way to organize and represent a set of documents. For example, CuriousLLM [487] considers passages in the text as entities and determines whether two entities should be connected based on their encoded textual similarity. On the other hand, Cheng et al. [58] uses a manually-defined prompt to convert a piece of text into a KG. Graph-RAG [98] first divides each document into chunks and then uses an LLM to detect all the entities in each chunk, including their name, type, and description. To identify the relation between any two entities, both entities and a description of their relationship are passed to an LLM. An LLM is then used again to summarize the content of each entity and relation to arrive at their final title. Lastly, AutoKG [40] uses a combination of LLM embeddings and clustering techniques to construct a KG from a set of texts.

### 3.3 Retriever

Real-world facts in KGs can provide grounded information for generative models, enhancing the reliability of the model output. Given the structured nature of KGs, they are naturally well-suited for retrieval. The goal is for a given question or query to retrieve either relevant facts<sup>3</sup> or entities that can help answer that question. Multiple considerations need to be considered during retrieval, including the type of facts we want to retrieve, the efficiency, and the amount of facts retrieved. In general, retrieval of KGs has two stages: identifying seed entities and retrieving facts or entities. We describe both below.

**Identifying seed entities:** The first step in retrieving the relevant facts for a given query is to identify a set of “seed entities”, which we’ll refer to as  $V_{\text{seed}}$ . Seed entities are the initial entities that are chosen to be highly relevant to the original query. Given such, we expect that triples that contain any of these entities or are nearby in the graph should provide helpful context. Multiple techniques exist for identifying the seed entities. Some works [181, 200, 251, 380, 522] assume that we are given a set of initial entities for each query. However, most works [110, 381, 443, 530, 308, 493] attempt to extract the entities from the query. One approach is through entity extraction [6], which uses methods specifically designed for extracting entities from a given text. Most works only extract entities from the original query. Another common approach is to extract a set of entities that are semantically similar to the original query [443, 347]. HyKGE [185] first generates a hypothesis and extracts entities from the original query and the hypothesis. Similarly, in order to reduce the possibility of hallucination, Guo et al. [134] uses an LLM to generate two similar questions and retrieves all entities found in the original and generated questions. In a similar vein, RoK [431] first uses chain of thoughts reasoning to expand the original query, extracting the seed entities from the expanded query.

**Retrieval Methods:** The outcome of the previous step provides us with a set of entities that are related in some capacity to the query. These entities are then leveraged to retrieve a set of facts or entities that can aid us in answering the query. We summarize the core retrieval methods below.

- **Traversal-based retriever:** These methods traverse the graph and extract paths to aid in answering a specific question. Given the set of seed entities,  $V_{\text{seed}}$ , Yasunaga et al. [492, 493], Zhang et al. [530] extract all paths up to length two between the entities in  $V_{\text{seed}}$ , resulting in a final entity set  $V$ . They further augment  $V$  by including all triples that connect any two entities in  $V$ . Given  $V$ , both [492, 530] only keep the top  $k$  entities by relevance scores. This is calculated by training a separate model that takes the text embedding of the query and entity as input and outputs how relevant

---

<sup>3</sup>Throughout this paper, we will refer to facts as triples or edges, interchangeably

the entity is to the query. For Yasunaga et al. [493], if  $|V| > 200$ , they randomly sample 200 entities. Sun et al. [381] use a version of beam-search to explore the KG. Jiang et al. [185], Feng et al. [110] extract all paths of length  $k \leq 2$  between seed entities. Alternatively, LARK [62] retrieves all facts that lie on paths  $\leq k$  in length starting from the seed entities. Delile et al. [78] first extract the shortest paths connecting all seed entities. They further prioritize some entities over others by considering the recency, importance, and relevance to the query of their associated text. OREOLM [158] traverse  $k$  hops from the seed entities, contextualizing the importance of each relation and entity to a path via a learnable  $d$ -dimensional embedding and it's LM-encoded representation. Zhang et al. [522] introduce a trainable retriever that traverses the graph starting from each seed entity. They also train a model to score each newly visited edge, only keeping a portion of them. KG-RAG [347] works in a similar manner, scoring each edge by its relevance and similarity to the query via a dense retriever. They then use an LLM to decide which paths to explore in the next step. RoG [271] uses instruction tuning to fine-tune an LLM to generate useful relation paths, which can be retrieved from the KG. KnowledgeNavigator [134] first uses the query to predict the expected number of hops,  $h_Q$  needed to traverse in the retrieval stage. It then traverses  $h_Q$  hops starting from the seed entities, using an LLM to score and prune irrelevant nodes. Wu et al. [456] operate in a similar manner; however, they choose which paths to traverse based solely on the relations. Furthermore, when scoring a path, all relations that lie on that path are considered when computing the score. RoK [431] considers a different approach, using the Personalized PageRank (PPR) score to identify useful paths. They further augment these paths by including the 1-hop neighbors of the seed entities. PullNet [380] assumes that each entity has an associated set of documents. Given a single seed entity, PullNet, traverses  $k$  hops, where in each iteration it extracts the facts for the newly observed entities. It also extracts any entities that are contained in documents associated with an entity found in the traversal. Furthermore, for each entity, only the top N facts are used, which are ranked via similarity to the query. KG-R3 [312] uses MINERVA [75], a reinforcement learning approach to mining paths between entities, to retrieve a set of important paths between both entities in the fact. Wang et al. [428] use an LLM to traverse the graph starting from the seed entities. At each iteration in the traversal, we choose the next node to visit by prompting an LLM. Specifically, given the information already collected in the traversal, the LLM is prompted to generate the remaining information needed to correctly answer the question. The neighboring node that best matches the required information is chosen as the next node to visit. They further instruction-tune the LLM.

- **Subgraph-based retriever:** These methods extract a subgraph of size  $k$  around each of the seed entities. Facts that contain one of the seed entities or are nearby, should be highly relevant to answering the question. Furthermore, they may actually contain the answer itself. Each of [308, 395, 181, 205] extract either the one or two hop subgraph around each seed entity. The final set of facts is the union of each individual subgraph. Gao et al. [118] propose to first extract the subgraph containing the seed entity and potential answers using the method in Sun et al. [379]. This is then partitioned into a set of smaller subgraphs. Then, they design a framework to rank the subgraphs, keeping the top  $k$  subgraphs for generation. For a question-choice pair, MVP-Tuning [164] considers the triple that contains the highest number of seed and choice entities. They further augment this by extracting the top  $k$  most similar questions in the dataset using BM25 [338], and extract the triples for each of them.
- **Rule-based retriever:** These methods use pre-defined rules or templates to extract paths from the graph. GenTKG [246] considers a temporal KG, where they first extract logical rules from the KG, and use the top  $k$  rules to extract paths in a given time interval for the seed entities. Both [72, 270] generate queries using SPARQL, which are then used to retrieve import paths. KEQING [406] decomposes the original query into  $k$  sub-queries using an LLM fine-tuned via LoRA [153]. For each sub-query, they find the most similar question templates, which are predefined. For each template, they further pre-define a set of logical chains, which are then used to extract matching paths for the seed entities in the sub-query from the KG.
- **GNN-based retriever:** GNN-RAG [289] trains a GNN for the retrieval task. A separate round of message passing is done for each query  $q$ , which is incorporated in the message computation along with the relation and entity representations. The GNN is then trained as in the node classification task, where the correct answer entity for  $q$  has a label of 1 and 0 otherwise. During inference, the entities with probability above some threshold are treated as candidate answers, and the shortest path from the seed entity is extracted. Liu et al. [251] use a conditional GNN [163] for retrieval, where for each query, only the seed entity (they assume there is only one) is initialized to a non-zero

representation based on the LLM-encoded query. They then run  $L$  rounds of message passing where after each layer  $l$  only the top-K new edges are kept, resulting in a set of entities  $C_q^l$ . This is determined by a learnable attention weight, which prunes the other edges from the graph. The final set of candidate entities is the union of candidate entities at each layer  $l$ ,  $C_q = \cup_{l=1}^L C_q^l$ . It is optimized in a similar manner to [289]. For each candidate entity, they retrieve an evidence chain by backtracking from the entity until it reaches the seed entity, choosing those edges with the highest attention weight. REANO [106] initializes the entity and relation representations via the mean-pooled representations of all mentions of that entity/relation in the texts, encoded by T5. They then run a GNN, which includes an attention weight that considers the relevance of a given triple to the original question (also encoded by T5). After running the GNN, they retrieve the top K triples in the KG that are most relevant to the question, where relevance is defined via the dot product between the triple encoded by the GNN and the question.

- **Similarity-based retriever:** STaRK [452] considers the vector similarity of the query to each entity. Each entity embeds both the textual and relational information together. They further consider multi-vector similarity, where the entities are encoded using multiple vectors. This is done by chunking the textual and relational information of each entity, with each chunk being embedded into its own vector. Both REALM [567] and EMERGE [566] extract the entities most similar to the query. While REALM only retrieves the entities themselves, EMERGE further retrieves the 1-hop subgraph around each entity.
- **Relation-based retriever:** Kim et al. [200] propose a general framework for reasoning on KGs using LLMs. They first use an LLM to segment the original query into a set of  $i \in I$  sub-sentences, where each sub-sentence  $S_i$  has an associated set of entities  $\mathcal{E}_i$ . For each sub-sentence, they further use an LLM to retrieve the top- $k$  most relevant relations  $\mathcal{R}_{i,k}$ . Given the set of  $k$  relations, for each sub-sentence, they retrieve all triples that contain a relation in  $\mathcal{R}_{i,k}$  and whose entities are in  $\bigcup_{i \in I} S_i$ . GenTKGQA [119] focuses on temporal KG QA. Like [200], they retrieve the top- $k$  relations for the query. They then retrieve all facts that contain one of the top  $k$  relations and satisfy the temporal constraints.
- **Fusion-based retriever:** These techniques consider a combination of different retrieval techniques. Mindmap [443] considers extracting some paths  $\leq k$  hops from the seed entity and the 1-hop subgraph of each seed. These two extracted components are combined into one subgraph. DALK [224] uses a procedure similar to Mindmap, where they extract both paths and the 1-hop subgraph around each seed entity. However, they argue that this procedure often results in the retrieval of redundant or unnecessary information. To remove these facts, they use an LLM to rank the retrieved facts given both the original question and the subgraph. Only the Top- $k$  most relevant facts are kept. UniOQA [244] considers two branches for retrieving. The first is a translator, which is a fine-tuned LLM that generates the answer in a CQL format. the second is a searcher that retrieves the 1-hop subgraph around the seed entities. When determining the answer, answers from the translator are prioritized over those from the searcher. KG-Rank [483] considers ranking all triples in the 1-hop neighborhood of the seed entities via the similarity of the relation to the query, the similarity of each triple to the encoded output of  $a = \text{LLM}(q)$ , and an MMR ranking [35] that uses the similarity score. Only the top-ranked triples are kept. GrapeQA [389] extends [492] by further including a set of “extra nodes”, which are the common neighbors of the entities retrieved via a path-based retriever. They further introduce a clustering-based method for pruning entities that may be irrelevant to the query. SubgraphRAG [232] considers both GNN and textual information. For the GNN, they consider initializing the node representations using a one-hot encoding to differentiate between seed entities and others. A GNN is then run for  $L$  layers, resulting in the final representation  $s_v$  for a node  $v$ . To retrieve the relevant triples, they consider first concatenating the final node representations for each triple  $(h, r, t)$  such that  $z_\tau = [s_h, s_t]$ . The probability of choosing this triple is then given by  $p(h, r, t) = \text{MLP}([z_q, z_h, z_r, z_t, z_\tau])$ , where  $z_q, z_h, z_r, z_t$  are the encoded textual representations of the query and the triple  $(h, r, t)$ , respectively. Only the top K triples are chosen.
- **Agent-based retriever:** These techniques use LLM agents to retrieve facts from the KG. KnowEdGPT [425] defines a set of tools for searching over a KG. Given a query, they generate a piece of code to search over the KG that considers the seed entities. The code is then executed over the KG to find the correct answer. KG-Agent [182] focuses on fine-tuning an LLM to generate the SQL code for retrieving the correct answer. Using a set of tools, they extract a set of paths that contain the seed entities. KnowAgent [568] first identifies the relevant actions for the query via a planning module. Using these actions, they then generate a set of paths that are used for generation.

**Other retrievers:** KICGPT [439] is concerned with the task of knowledge graph completion, where given a partial fact  $(h, r, *)$ , we want to predict the correct entity  $\hat{e}$ . KICGPT retrieves the entities by first scoring all possible entities using a traditional KG embedding score function. That is, for a score function  $f(\cdot)$  and a partial fact  $(h, r, *)$ , they compute the set of scores  $\{f(h, r, e) \forall e \in \mathcal{V}\}$ . They use RotatE [383] for the function  $f(\cdot)$ , a popular approach. Only the top  $k$  entities by score are retrieved. To supplement their knowledge, they also retrieve all triples with (a) the same relation as the query and (b) all triples that contain the entity  $h$  in the query. These are referred to as the analogous and supplement triple pools, respectively.

### 3.4 Organizer

In this subsection, we describe how the retrieved knowledge is organized for generation. More concretely, this is how the information is formatted when given to the generator. Note that not every method necessarily has an explicit organizer. We summarize the common methods below:

- **Tuple-based organizer:** These methods consider each piece of retrieved information as an ordered triple. For example, it would include a triple in the generation prompt as “(entity 1, relation 1, entity 2)”. Similarly, a path of length  $m$  is given by “(entity 1, relation 1, entity 2, relation 2, …, entity  $m$ )”. The entities and relations are usually represented either as their names or IDs. Each triple or path is usually listed on a separate line. Many works append the retrieved paths to the original query as additional context [381, 185, 289, 347, 271, 62, 251, 431, 568]. Other works that retrieve facts instead of paths operate in a similar manner, where instead they append the triples [308, 567, 72, 246, 483, 181, 200, 119]. Some methods [566, 395] consider only including the retrieved entities as the context. Given a set of facts, KG-R3 [312] first lists all entities and then relations, i.e., “(entity 1, entity 2, …, entity  $m$ , …, relation 1, …, relation  $m - 1$ )”. Delile et al. [78] consider a KG where each entity has an associated chunk of text. Each text chunk for an entity is considered as a different piece of information to be included in the context. Both [395, 119] represent each entity and relation as an embedding, which is the combination of the LLM and GNN embedding. Liu et al. [251] further include the probability of each path containing the correct answer given by the GNN model. MVP-Tuning [164] considers combining multiple facts that share the same subject and relation to remove redundant information. That is, for a subject-relation pair (subject, relation), they denote the facts for  $k$  possible objects as “subject relation {object 1, …, object  $k$ }”. KG-Agent [182] stores the current KG information and the historical reasoning programs in lists.
- **Text organizer:** Wu et al. [456] verbalize the retrieved subgraph by passing each triple to the LLM and prompting it to convert it to a text representation. MindMap Wen et al. [443] uses a similar procedure for subgraphs, where each is organized as a path before being passed to the LLM. Some methods use a set of pre-defined templates to verbalize the triples or paths [134, 244, 205]. Wang et al. [406] experiment with verbalizing either via an LLM or pre-defined question templates, finding that LLM-based verbalizing works better for ChatGPT while template-based works better for LLaMA [398]. KICGPT [439] uses a combination of data preprocessing and LLM prompting to convert the triples to text. StaRK [452] uses an LLM to synthesize each entity with its relational and textual information. Note that they use some pre-defined templates that depend on the specific task. CoTKR [457] uses an LLM to summarize and then re-write a subgraph of facts for a question through a “knowledge rewriter”. To train the rewriter, preference alignment is used, which optimizes the rewriter’s output to match our preferred output. First,  $k$  representations of the retrieved subgraph are produced, with ChatGPT choosing the best and work representations as the most and least preferred solutions.
- **Other organizer:** There are some exceptions to the previous classification. KnowledGPT [425] represents the information in the form of a python class format. They also experiment with including additional information like the entity description and entity-aspect information.
- **Re-Ranking:** Some methods also *re-rank* the information in a specific order. This is done as the order of information can have a subtle impact on LLM performance. Delile et al. [78] order the text chunks of each entity based on the impact (measured by # of citations of the parent paper) and the recency. Dai et al. [72] sort the triples by the relevancy score to the triple. Choudhary and Reddy [62] attempt to order the paths in a logical matter, such that for a given path, the subsequent paths build upon it. Yang et al. [483] re-rank the retrieved triples using a task-specific Cross-Encoder [187]. STAARK [452] considers re-ranking the retrieved entities using an LLM. The

LLM is given the relational and textual information of each, and is asked to give it a score from 0 to 1, which is then used for re-ranking. GenTKG [246] orders the paths by the time they occurred, further including the time with each. KICGPT [439] ranks all entities using the score of the KG embedding score function, keeping only the top  $k$  entities. KICGPT re-rank the entities using in-context learning, where they prompt the LLM with examples from the analogy and supplement pool, as prior knowledge to aid the LLM in how to re-rank the entities.

### 3.5 Generator

In this section we describe how the retrieved and organized data is used to generate a final response to the query. We categorize these generators according to the type of methods used to create these responses.

- **LLM-based generator:** The vast majority of works use a LLM to generate the response. The input to the LLM is the original query and retrieved and organized context, formatted using a specific template. The most commonly used LLMs include ChatGPT [310], Gemini [390], Mistral [180], Gemma [391], among others. For open-source models where the weights are publicly available, fine-tuning is sometimes used to modify the weights for a specific task [244, 568]. This is often done through LoRA [153], which allows for efficient fine-tuning.
- **GNN-based generator:** Some methods use graph neural networks (GNNs) [207] to conduct the generation. Yasunaga et al. [492], Taunk et al. [389], Feng et al. [110] extract both the language and GNN embeddings for each potential answer (i.e., entity) conditional on the query. The probability of a single entity being the answer is then learnt based on the fusion of the two types of embeddings.
- **Other generators:** Zhang et al. [530], Yasunaga et al. [493], Hu et al. [158] formulate the prediction as a masked language modeling (MLM) problem. The goal is to predict the correct value (i.e., entity) for the masked token which answers the query. To do so, they fine-tune RoBERTa [263] language model. KG-R3 [312] scores the potential answer entities by performing cross-attention the representations of the query and each individual entity. PullNet [380] uses GraftNet [379] to score the different entities. Gao et al. [118] first selects the correct subgraph by computing the cosine similarity between the query and subgraph representations. For the subgraph with the highest similarity, it's fed to GraftNet [379] to select the most probable entity. REANO [106] passes the encoded triples and their associated text passages to the T5 decoder. The task is framed as a classification problem, where the goal is to assign the highest probability to the triple with the correct answer.

### 3.6 Resources and Tools

In this section, we list common tools and KGs that are used in graph RAG systems. For each, we give a brief description and a link to the project.

#### 3.6.1 Data Resources

- **Freebase**<sup>4</sup> [27] is an encyclopedic KG that contains a large variety of general and basic facts.
- **ConceptNet**<sup>5</sup> [373] is a semantic graph, where the links in the graph are used to describe the meaning of different words or ideas.
- **WikiData**<sup>6</sup> [404] is a crowdsourced knowledge base that functions as a structured analog to the Wikipedia encyclopedia.

#### 3.6.2 Tools

- **Graph RAG**<sup>7</sup> [98] is an official open-source implementation of the Graph RAG [98] framework. It can further be installed via the *graphrag* python package.

---

<sup>4</sup><https://developers.google.com/freebase>

<sup>5</sup><https://conceptnet.io/>

<sup>6</sup>[https://www.wikidata.org/wiki/Wikidata:Main\\_Page](https://www.wikidata.org/wiki/Wikidata:Main_Page)

<sup>7</sup><https://github.com/microsoft/graphrag>

- **LangChain**<sup>8</sup> is an open-source framework for using LLMs with various components and applications, including RAG, where using RAG on KGs is supportive.

## 4 Document Graph

A document graph typically models the connections between different documents or various granularity of documents. It is widely observed in real-world scenarios [351, 367, 471], such as hyperlinks connecting different websites and citations linking one paper to another. Additionally, a document graph's relationships between sentences and entities can explicitly capture semantic and syntactic contextual information. The structural information in documents can serve as a valuable resource for GraphRAG, aiding LLMs in various tasks. For example, in the retrieval process in RAG, the document containing the answer may have less apparent connections with the question, such as in multi-hop question answering [288]. However, we can identify these relevant documents through their connections to other documents whose context is strongly aligned with the question's context [90]. In this section, we will systematically review the document graph.

### 4.1 Application Tasks

Document graphs are beneficial for a wide range of tasks. In this subsection, we will review the tasks where document graphs can play a significant role. Although document graphs have not yet been fully integrated with LLMs, they still offer great potential for enhancing the capabilities of LLMs in various applications.

- **Multi-document Summarization(MDS):** Multi-document summarization aims to condense the contents of multiple documents into a cohesive summary. Summarizing an entire corpus can involve large volumes of text, often exceeding the context window limitations of LLMs. Document graphs can help compress the corpus by extracting key components and their relationships, proving highly beneficial for MDS [491, 408, 237, 519, 462, 515, 482, 231, 45]. These graphs also provide different levels of granularity for summarization through hierarchical clustering [98].
- **Text generation:** Text generation focuses on producing coherent and meaningful text. While text-based RAG models have been widely used to generate more reliable text, document graphs can further enhance this process by retrieving similar documents using graph topology. For instance, when writing the abstract of a paper, access to its cited papers in the related work section can significantly improve writing efficiency, as these referenced papers often contain relevant knowledge and context [429, 313].
- **Document Retrieval:** Document retrieval aims to find a list of documents relevant to a given query, which is a key task in Information Retrieval (IR). The exact query terms may not always appear together in the candidate document; however, by leveraging the connections between documents, we can retrieve related documents through those documents that are strongly linked to the query. Thus, it becomes essential to consider document-level relationships in the retrieval process. Several works [90, 501, 544, 249] have leveraged document graphs with various granularities to improve document retrieval and ranking. Rather than retrieving the entire document, graphs can also be used to retrieve specific segments, such as chunks.
- **Document Classification:** Document classification is a fundamental task in natural language processing. Traditional methods often focus on the locality of words, limiting their ability to capture long-distance and non-consecutive word interactions. Moreover, these methods typically focus on individual documents, overlooking the relationships between them, where connected documents often exhibit homophily, meaning they are more likely to share similar labels. Building a graph can help enhance document classification by leveraging both local and global relationships between words within a single document [538, 259] and by utilizing document-level relationships [518, 461].
- **Question Answering:** Question answering aims to provide answers to questions based on information from documents and is a fundamental task for RAG. However, documents used for question answering can be long, and traditional methods often focus on local structures within these documents, neglecting their global structure, which is crucial for long-range understanding. Additionally, some multi-hop questions require reasoning across multiple documents, necessitating the use of

---

<sup>8</sup>[https://python.langchain.com/v0.1/docs/use\\_cases/graph/constructing/](https://python.langchain.com/v0.1/docs/use_cases/graph/constructing/)

document-level relationships. Humans often consolidate scattered information into structured knowledge to streamline the reasoning process and make more accurate judgments, in line with cognitive load theory [384, 243]. Graph-based methods are well-suited for this task by constructing word-level document graphs [307, 410], utilizing document-level relationships [147, 428] and leveraging hierarchical interactions [108].

- **Relation Extraction:** Relation extraction aims to extract semantic relationships between entities in text, which often requires local, global, syntactic, and semantic dependencies, especially in the case of document-level relation extraction. Graphs have been proven to be helpful for document-level relation extraction [409, 302, 344, 63, 558] by capturing these dependencies more effectively.

In addition to the aforementioned tasks, graphs have also been shown to be helpful in other areas, such as fake news detection [155], coherence assessment [260], and machine translation [471].

## 4.2 Document Graph Construction

Different tasks may require different types of document graphs, such as document-level, sentence-level, or word-level graphs. Therefore, the method of obtaining the document graph plays a crucial role. There are primarily two approaches to constructing a document graph: explicit construction and implicit construction.

- **Explicit Construction.** In real-world scenarios, many documents have explicit connections. Examples include web pages linked through hyperlinks, academic papers citing other works, and social media posts connected by reposts, comments, and interactions. In these cases, it is natural to connect the documents to build a document graph, as the connected documents often share semantic relationships [320]. For instance, Asai et al. [10] constructed a Wikipedia graph based on hyperlinks between Wikipedia articles. Li et al. [240, 239] follow the same way to leverage hyperlinks to construct graphs. Yu et al. [498] built a graph using an external knowledge graph (KG), where each node represents a retrieved passage mapped to entities within the KG, and two passage nodes are connected if their mapped entities are linked in the KG. These connections have been leveraged to pretrain large language models (LLMs), enhancing their performance across various tasks [494, 572, 157].
- **Implicit Construction.** Despite the presence of explicit connections between documents, different components within those documents also exhibit important semantic and syntactic relationships. The structure of these components can be highly beneficial for various tasks in natural language processing [450, 294]. For example, semantic parsing graphs, such as Abstract Meaning Representation (AMR) graphs, can be leveraged to enhance information extraction [546, 162] and text summarization [88]. Additionally, syntactic parsing trees are exploited to understand grammatical structure and resolve ambiguities [533].

The implicit construction of document graphs is highly diverse, adapting to the specific requirements of different tasks. For instance, nodes in document graphs can represent various granularity, such as words, entities, sentences, text segments, paragraphs, documents, or topics. In addition, document graphs often exhibit heterogeneity, where edges connect different types of nodes. Next, we will detail the construction of different types of edges:

- *Word-word edge:* The word-word edges connect words with semantic or syntactic relations or dependencies. There are various methods to establish these connections, including word co-occurrence within a sliding window [408, 501, 518, 249, 76], dependency parsing graphs [470, 386, 471, 516, 326, 401] generated by NLP parsing tools [219, 216], Abstract Meaning Representation [476, 546, 162, 88, 475] and semantic graphs [90, 370, 313] based on different representations [17, 359]. Additional techniques include coreference resolution [471, 344, 222, 82, 369], word embedding similarity [410, 415] and using large language models (LLMs) to extract relationships between words [98]. For entities, the construction process is similar to that for words, though entity extraction methods are often required to first identify entities [303, 6, 317].
- *Word-Sentence edge:* Word-sentence edges connect words to sentences based primarily on the relationship of belonging. These edges are constructed by linking words to the sentences they appear in [408, 155, 63], with edge weights often measured using term frequency-inverse document frequency (TF-IDF). Besides, Ramesh et al. [332] connects the entities with passage titles via hyperlinks with existing out-of-box entity linkers.

- *Sentence-Sentence edge*: Sentence-sentence edges connect sentences based on their semantic similarity or relationships. For example, these edges can be constructed through sentence interactions [491, 284, 165], similarities between TF-IDF representations [237, 45], BM25 [297], sentence embeddings [260, 410, 525], Part of speech (Pos) feature and N-Gram feature [482]. Additionally, Zheng and Kordjamshidi [556] construct a Semantic Role Labeling(SRL) graph using AllenNLP-SRL model [359]. This allows for connecting sentences within long documents or across different documents, which is particularly useful for LLMs with limited context length.
- *Sentence-document edge*: Sentence-document edges are constructed by linking sentences to the specific documents they belong to [392].
- *Document-document edge*: Document-document edges connect documents based on their similarities. These edges can be constructed when entities in one document are referenced or shared by another [392, 90], through document clustering [410], embedding similarity [237], topic similarity [525] or structural similarity [260].

Document graphs typically exhibit heterogeneity, consisting of multiple types of edges. In addition, several hierarchical graphs have been proposed [444, 540], where different levels of abstraction are captured. The graph structure can also be dynamic or updated during the learning process [408, 302, 410, 259]. Moreover, the node in the document graph can also be the response of LLMs. For example, GoR [520] connects the history responses of LLMs with the responding chunks of documents for long-context summarization.

### 4.3 Retriever

The retrievers for document graphs typically follow the general retriever design, as described in Section 2.4. However, there are some special retrieval methods as follows:

- **Pre-Retrieval:** As introduced earlier, there are many scenarios where building a document graph is necessary. However, constructing a fine-grained graph for a large volume of documents can be inefficient and unnecessary. The pre-retrieval aims to first retrieve relevant documents based on the query and then construct a graph. For example, Thayaparan et al. [392] use pre-trained GloVe vectors to first extract relevant sentences and then construct a graph based on the retrieved sentences. Zheng and Kordjamshidi [556], Yu et al. [498] also construct graphs based on the retrieved information.
- **Graph similarity-based retriever:** Graph similarity aims to measure the similarity between two graphs. If both the query and the retrieval data source are graphs, calculating the graph similarity is essential to retrieve relevant information. For instance, [544] leverages the General Maximum Common Subgraph (GMCS) method to retrieve related graphs.
- **Iterative Retriever:** In certain tasks, such as multi-step question answering, the node containing the answer might not be directly similar to the query. Iterative retrievers address this by first retrieving nodes related to the query and then using the retrieved information to iteratively retrieve subsequent nodes. For example, Wang et al. [428], Zhang et al. [541] utilize iterative retrieval methods for multi-step question-answering tasks, and Ma et al. [278] iteratively retrieve the documents based on the existing knowledge graph. Asai et al. [10] train a Recurrent Neural Network (RNN) to recurrently retrieve relevant information with the Bayesian Personalized Ranking (BPR) loss.
- **Topology-based Retriever:** Various topological relationships in a graph can be used to measure different types of similarity. For example, proximity-based topological similarity measures the structural distance between two nodes, while role-based topological similarity assesses the similarity of the roles nodes play within the graph. These similarities can also be leveraged in the retrieval process [429].

### 4.4 Organizer

In this section, we describe the organizers for document graphs. GraphRAG on document graphs is still in its early stage, and many works do not incorporate explicit organizers, as shown in Section 2.5. We summarize the existing methods below:

- **Graph Pruning:** Graph Pruning refines the retrieved subgraph to reduce irrelevant information and improve computational efficiency. For example, Hemmati and Ghassem-Sani [149] prune graphs

based on the local clustering coefficient, while Zhang et al. [539] employs path-centric pruning to incorporate off-path information. Li et al. [229] dynamically drop irrelevant nodes during decoding, and Angelova and Weikum [8] prune edges based on a similarity threshold. Additionally, Edge et al. [98] uses community detection algorithms to create distinct communities that are then fed into the generators.

- **Reranking:** Reranking methods aim to reorder retrieved information to facilitate the generation. For example, Yu et al. [498], Zhang et al. [541], and Dong et al. [90] use GNNs to rerank retrieved passages. Li et al. [239] perform listwise reranking to reorder passages expanded via the graph structure.

#### 4.5 Generator

In this section, we summarize commonly used generators for document graphs. Various methods are applied within document graphs, depending on the input format. Some approaches use the entire graph as input, in which case GNNs and Graph Transformers are commonly employed. Other methods take individual sentences as input, where RNNs or (Large) Language Models are suitable. Additionally, some works leverage both graph and text as inputs, requiring integrated methods that can process multimodal data effectively. We detail each category in the following:

- **GNN-based generator:** Numerous works model tasks as graph-related problems, leveraging GNNs as generators. Conventional GNNs, such as GCN [206], GraphSAGE [138], and GAT [402], are widely adopted in various studies [320, 392, 491, 300, 313, 90]. When graphs contain edge relations, Relational Graph Convolutional Networks (R-GCNs) are often used to capture these relationships effectively [76, 297]. In addition, some works incorporate graph contrastive learning techniques to enhance performance [462, 524, 149].
- **Graph Transformer-based generator:** Graph transformers [504], which capture global information across the graph, are used to encode graph structures for various tasks [518, 292]. These models leverage the transformer’s self-attention mechanism to capture dependencies beyond local neighborhoods, making them well-suited for tasks requiring global context.
- **RNN-based generator:** Recurrent Neural Networks (RNNs), such as LSTMs, are popular for processing sequences. Therefore, RNNs are employed when the input is in text form [369].
- **LLM-based generator:** LLM-based generators typically transform the retrieved subgraph into text before using large language models (LLMs) for generation [428, 98]. Various models are used in this approach. For instance, BERT [81] has been applied by [401, 233, 541] to support generation tasks, while Yu et al. [498] and Ju et al. [191] leverage the T5 model [329], and Chen et al. [48] use RoBERTa [263].
- **Integrated Generator:** Some works leverage both graph and text data simultaneously for generation, using integrated generators that combine graph models with text generation models. For example, Xu et al. [476] and Wang et al. [410] employ a combination of RoBERTa and GCN, while Ramesh et al. [332] fuse GNN with T5 to harness the strengths of both graph structure and language model capabilities.

In addition to the aforementioned generators, some approaches embed graphs directly into text generation models. For example, Li et al. [237] proposes replacing traditional self-attention layers in transformers with graph-informed self-attention, enabling the model to integrate graph structure directly into the generation process.

#### 4.6 Resources and Tools

The data resources for document GraphRAG can include any type of document, and thus, we do not provide a comprehensive list here. Instead, in the following, we introduce several tools specifically designed or commonly used for GraphRAG on document graphs:

- **CoreNLP:**<sup>9</sup> The Stanford CoreNLP natural language processing toolkit offers a comprehensive set of natural language processing tools, including token and sentence boundaries, parts of speech, named entities, numeric and time values, dependency and constituency parses, coreference,

---

<sup>9</sup><https://github.com/stanfordnlp/CoreNLP>

sentiment, quote attributions, and relations. These tools are valuable for constructing document graphs.

- **spaCy**:<sup>10</sup> The spaCy is an advanced natural language processing library known for its speed and neural network models, which are optimized for tasks such as tagging, parsing, named entity recognition, text classification, part-of-speech tagging, dependency parsing, sentence segmentation, lemmatization, morphological analysis, and entity linking.
- **BLINK**:<sup>11</sup> BLINK is an entity linking python library that uses Wikipedia as the target knowledge base.
- **OpenIE**<sup>12</sup>: Open Information Extraction (OpenIE) is a tool for extracting structured information from text. It is valuable for generating triples from unstructured text, which can be used as nodes and edges in document graphs.
- **CogComp NLP**<sup>13</sup>: Developed by the Cognitive Computation Group, this suite includes tools for natural language processing tasks such as named entity recognition, sentiment analysis, and coreference resolution.
- **GraphRAG** [98]<sup>14</sup>: GraphRAG is a data pipeline and transformation suite that is designed to extract meaningful, structured data from unstructured text using the power of LLMs. The process involves extracting a knowledge graph out of raw text, building a community hierarchy, generating summaries for these communities, and then leveraging these structures when performing RAG-based tasks.
- **LangChain**<sup>15</sup>: LangChain is a framework for developing applications powered by LLMs, with use cases in document analysis and summarization, RAG, chatbots, and code analysis. Notably, LangChain supports Graph Transformers, which convert documents into graph-structured formats, making it highly suitable for processing document graphs.
- **Neo4j**<sup>16</sup>: Neo4j is a graph database platform offering comprehensive tools for storing, visualizing, managing, and querying graph data. It includes an LLM Graph Builder, which can extract graphs with LLMs. It also provides GraphRAG demos to demonstrate how to implement a LLMs and RAG system with Neo4j.
- **LlamaIndex** [253]<sup>17</sup>: LlamaIndex is a data framework designed to support the development of applications based on LLMs. It allows developers to seamlessly integrate data sources, ranging from various file formats to applications and databases, with LLMs. LlamaIndex features an efficient data retrieval and query interface, enabling developers to input any LLM prompt and receive context-rich, knowledge-enhanced outputs. Notably, it includes a Property Graph Index that facilitates the building, modeling, storage, and querying of graphs.
- **Haystack**<sup>18</sup>: Haystack is an end-to-end framework for building applications powered by LLMs, Transformer models, vector search, and more. It supports a variety of use cases, including RAG, document search, question answering, and answer generation. Haystack enables the orchestration of state-of-the-art embedding models and LLMs into custom pipelines for creating comprehensive NLP applications. Additionally, it supports Neo4j as a DocumentStore, making it suitable for document graph storage and query operations.

## 5 Scientific Graph

Scientific graphs refer to graph-structured data used in domains such as drug discovery [433, 167, 265] and biomedicine [484, 255, 78, 185, 30, 443], both of which are common application areas for GraphRAG. Therefore, in this section, scientific graphs specifically refer to molecular graphs and medical graphs.

<sup>10</sup><https://github.com/explosion/spaCy>

<sup>11</sup><https://github.com/facebookresearch/BLINK>

<sup>12</sup><https://github.com/dair-iitd/OpenIE-standalone>

<sup>13</sup><https://github.com/CogComp/cogcomp-nlp>

<sup>14</sup><https://github.com/microsoft/graphrag>

<sup>15</sup><https://www.langchain.com/>

<sup>16</sup><https://neo4j.com/>

<sup>17</sup><https://www.llamaindex.ai/>

<sup>18</sup><https://haystack.deepset.ai/>

In recent years, significant progress has been made in the development of artificial intelligence for science [1, 193, 393, 283]. Machine learning (ML) and deep neural network technologies are increasingly driving scientific discovery from experimental data. Notably, generative models such as Large Language Models (LLMs) have achieved remarkable success in working with scientific graph data, including molecular graphs and biomedical graphs. In molecular graphs, for example, atoms serve as nodes, while chemical bonds represent edges, capturing the structure of molecules. AI technologies can handle both prediction and generation tasks on these graphs, driving advancements in fields like drug discovery.

Despite the impressive capabilities of generative models like LLMs, several challenges still hinder their applications to scientific domains. One of the most prominent challenges is the lack of domain-specific expertise. In fields like drug discovery, molecule generation is crucial, yet traditional generative models often struggle with producing incorrect or scientifically invalid structures. In medical question-answering (QA) tasks, incorrect answers, hallucinations, and limited interpretability are frequently encountered. To address these challenges, recent works have proposed leveraging external knowledge databases to enhance generation via GraphRAG. GraphRAG improves accuracy by retrieving relevant scientific graphs from extensive databases to guide the generation or answering process. This approach ensures scientific validity by incorporating known valid graph structures, leverages existing knowledge for practical applications, and accelerates the generation process by narrowing the search space.

## 5.1 Application Tasks

Scientific graphs are beneficial for a wide range of tasks. In this subsection, we will review some representative tasks where scientific graphs can play a significant role.

- **Molecule generation:** Molecule generation refers to the process of creating or designing new molecular structures, often using generative models [433, 167]. It plays a crucial role in fields like drug discovery. The use of scientific graphs, especially molecular graphs, can enhance the rationality and accuracy of generated molecular structures in molecular generation. Typically, an inquiry such as a molecule is given to retrieve the most relevant molecular structures to guide molecular generation.
- **Molecule property prediction:** Molecule property prediction refers to the use of computational methods to estimate the physical, chemical, or biological properties of molecules based on their structure [265]. It has proven highly effective in accelerating the drug discovery process while significantly reducing associated costs. The use of scientific graphs, especially molecular graphs, can enhance the accuracy of prediction. To achieve this, a query molecule is provided, and similar molecules are identified as demonstrations to improve the prediction.
- **Question answering:** Question answering (QA) in the scientific domain refers to the use of computational methods to provide accurate and context-specific answers to scientific questions [484, 185, 443, 225, 175, 449, 366]. This involves retrieving or generating information from scientific literature, databases, or other resources to address complex, domain-specific queries, such as "After meals, I feel a bit of stomach reflux. What medication should I take for it?" In graphRAG, the scientific literature is usually converted into a knowledge graph to provide a basis for answering questions or to enrich the query.

## 5.2 Scientific Graph Construction

In GraphRAG, the choice and construction of data sources are crucial and typically include both public and private datasets. Public datasets are often derived from widely recognized resources, such as molecular databases like PubChem [204], ChEMBL [121], and ZINC [170], as well as biomedical literature and data sources like PubMed [269] and ClinicalTrials [509]. These datasets provide a broad foundation of information across various domains, offering reliable and authoritative references for the model. On the other hand, private datasets contain user-specific information, such as medical records from hospitals or confidential clinical trial data [449]. These datasets are highly confidential and unique, enabling GraphRAG models to offer personalized and proprietary knowledge support.

For chemistry, molecules can be represented in four main forms: 1D SMILES (Simplified Molecular Input Line Entry System) [433, 265], 2D molecular graphs [274], 3D molecular graphs with coordinates [167], and text captions describing molecular structures [78]. 1D SMILES is a linear string

generated through depth-first search (DFS) on the molecular graph, following specific rules. 2D molecular graphs represent atoms as nodes and bonds as edges, visually showing the connectivity of atoms. 3D molecular graphs incorporate spatial coordinates for each atom, reflecting the molecule's structure in three-dimensional space, which is crucial for tasks such as molecular docking and reaction prediction. Text captions, on the other hand, provide a natural language description of the molecular structure, which can be used in tasks that involve textual data interpreting chemical structures from text. Scientific graphs can typically be constructed as follows:

- **Text-based construction:** Text-based graph construction is the most commonly used method, which can transform textual scientific knowledge into knowledge graphs. Delile et al. [78] believe that text captions suffer from issues of information redundancy and imbalance. By constructing text as a knowledge graph, it is possible to rebalance the retrievable information and reduce redundancy. Building a knowledge graph typically involves two key steps:
  - *Entity Extraction:* This step involves identifying and extracting key entities from the text, such as domain-specific terms (e.g., chemical compounds, genes).
  - *Relationship Extraction:* After identifying the entities, the next step is to extract the relationships between these entities, determining how they are linked in the given context (e.g., "A is related to B"). This step forms the structural backbone of the knowledge graph.
- **SMILES-based construction:** Many chemical databases store data in analytical descriptor formats such as SMILES [434]. To model graphs, SMILES representations need to be transformed into graph structures. SMILES-based construction utilizes a library such as RDKit [217] to read the SMILES notation, create a molecular object, and extract atomic and bonding information to construct a 2D graph. Each atom in the molecule is represented as a node, and each bond is represented as an edge between nodes. The atoms carry features such as atom type, degree, charge, and aromaticity, while the bonds can include bond type (single, double, etc.) and whether they are part of a ring.
- **3D graph construction:** Huang et al. [167] introduce IRDIFF, an interaction-based retrieval-augmented 3D molecular diffusion model designed for target-specific molecule generation. For pretraining, PMINet is utilized to capture interactive structural context information with binding affinity signals, leveraging the PDBbind v2016 dataset. This dataset offers 3D protein structures and a substantial set of experimentally validated protein-ligand complexes. These protein structures are primarily obtained via techniques like X-ray crystallography, nuclear magnetic resonance (NMR), or cryo-electron microscopy, containing detailed atomic coordinates, bond angles, and secondary structure information.

### 5.3 Retriever

The retriever is responsible for locating relevant information based on the input query. In GraphRAG, this information typically consists of graph-structured data. The retriever can be roughly classified as heuristic-based retriever and deep learning-based retriever.

- **Heuristic-based retriever:** A heuristic-based retriever employs predefined rules, algorithms, and heuristics to identify and retrieve relevant knowledge from graph-structured data sources. Heuristic-based retrieval can be categorized into several types as below:
  - *Similarity-based retriever:* Wang et al. [433] retrieve exemplar molecules based on their similarity to the input molecule, using cosine similarity as the metric. MindMap [443] encodes the entities from the query and the external knowledge graph into dense embeddings using BERT, and then retrieves the entity set with the highest similarity scores.
  - *Matching-based retriever:* A matching-based method provides an alternative approach, enabling LLMs to generate evidence-based responses by leveraging comprehensive private data. Specifically, Wu et al. [449] identify the most relevant graph through a top-down matching process. Once the relevant content is identified, the LLM generates an intermediate response with its assistance, enhancing both the transparency and interpretability of the results.
  - *Knowledge graph-based retriever:* A knowledge graph-based method can effectively identify and retrieve the necessary information to respond to a query. Pelletier et al. [318], Li et al. [225] use named entity recognition and relation extraction to connect user queries with relevant entities in the knowledge graph, thereby uncovering interpretable and actionable insights from

existing biomedical knowledge. This approach significantly enhances the transparency and utility of predictive models. Delile et al. [78] map the text chunks to the knowledge graph, then utilize graph distances to find the chunks most relevant to the user’s question. In addition, this work introduces a scoring metric that balances the data by giving each concept mapped along the shortest path an equal opportunity. This metric prioritizes text chunks based on both their recency and their impact. HyKGE [185] first queries the LLM to generate a hypothetical output and extracts entities from both the output and the query. Then, HyKGE retrieves reasoning chains between any two anchor entities in an existing knowledge graph, such as CMeKG [30] and CPubMed-KG, and feeds the reasoning chains along with the query into the LLM. KG-Rank [484] identifies entities within the query and retrieves the related triples or sub-graphs from the KG to gather factual information.

- *Fusion-based retriever:* Soman et al. [366] begins by retrieving the relevant node in the knowledge graph based on vector similarity to the query’s entity. Then, it retrieves the context triples (Subject, Predicate, Object) linked to this node within the knowledge graph.
- **Deep Learning-based Retriever:** Deep learning-based retrievers can extract relevant knowledge to guide the generation process, such as the generation of molecules or proteins. Specifically, Huang et al. [167] uses a pre-trained protein-molecule interaction network named PMINet to extract interactive structural context information between the target protein and ligands in the reference pool to guide the generation of target-aware ligands. Jeong et al. [175] utilize the off-the-shelf MedCPT retriever, a tool specifically tailored for retrieving documents in response to biomedical queries, capable of retrieving up to ten relevant pieces of evidence for each input. DALK [225] filters out noise and retrieves the most relevant knowledge by utilizing the ranking capabilities of LLMs.

#### 5.4 Organizer

Basically, there are two types of the organizer: embedding-based and query-based organizers according to how the retrieved knowledge is utilized.

- **Query-based organizer.** The simplest and most direct organizer is a query-based organizer, which integrates the retrieved information with input queries to generate responses. Specifically, HyKGE [185] enhances the reasoning process by incorporating external knowledge into the large language model. In this framework, the retrieved reasoning chains are fed into the LLM alongside the original query. This enables the model to ground its responses in structured, contextually relevant information, improving both the accuracy and depth of the generated output. Wu et al. [449] prompt the LLM to answer the question by providing the retrieved entity names and their relationships in a concatenated form. This approach involves retrieving relevant entities and the relationships between them from an external knowledge source and structuring this information as a cohesive input for the LLM. DALK [225] integrates the query and retrieves knowledge and then feeds it into LLMs for reasoning and getting the predicted answer. Similarly, KG-Rank [484] combines the re-ranked triplets with the task prompt and inputs them into LLMs for answer generation. In contrast to the methods mentioned above, Soman et al. [366] begin with context pruning. Specifically, this approach refines the retrieved context by selecting only the most semantically relevant elements needed to answer the query promptly. The input prompt is then combined with this pruning-aware information, producing an enriched prompt that is fed into the LLM for text generation. Delile et al. [78] develop a data rebalancing mechanism to ensure that each entity relevant to a question has an equal opportunity of being represented while also highlighting recent significant discoveries. This rebalanced knowledge is then combined with the query prompt and provided as input to the LLM.
- **Embedding-based organizer.** Embedding-based organizer integrates the retrieved information with input embedding to generate responses. Specifically, Wang et al. [433] uses a lightweight, trainable standard cross-attention mechanism to fuse the embedding of the input molecule and the retrieved example molecules. Similarly, Huang et al. [167] uses a trainable cross-attention mechanism to fuse the enhanced embeddings of the retrieved example ligands and the generated molecules.

#### 5.5 Generator

The generator is a core component of the GraphRAG model and is responsible for producing the final output by integrating retrieved evidence with the input data. Generators can be broadly classified into

three categories according to the type of generated models they use: transformer-based generators, diffusion model-based generators, and large language models-based generators.

- **Transformer-based generator:** RetMol [433] utilizes the Megatron version of the molecule generative model Chemformer for drug discovery. Specifically, the Chemformer model is a Transformer-based model that can be efficiently applied to tasks in chemistry.
- **Diffusion-based generator:** Huang et al. [167] introduce a novel interaction-based retrieval-augmented diffusion model (IRDIFF) for structure-based drug design. Specifically, IRDIFF is able to generate molecules that bind strongly to the target pocket by utilizing protein-molecule interaction data between reference proteins and the target protein to guide the diffusion model.
- **LLM-based generator:** Some approaches leverage large language models, such as LLaMA2, LLaMA3, GPT-4, and Gemini, etc [449, 175, 318, 265, 225, 366]. For instance, MedGraphRAG [449] utilizes GraphRAG to improve the answering capabilities of models such as LLaMA2, LLaMA3, Gemini, and GPT-4 for medical question answering. MolecularGPT [265] utilizes GraphRAG to improve GPT-3’s predictive capabilities for molecular property prediction. DALK [225] employs GraphRAG to boost GPT-3.5-turbo’s ability to answer questions related to Alzheimer’s Disease. HyKGE [185] utilizes GraphRAG to enhance the medical question-answering capabilities of GPT-3.5 and Baichuan13B.

## 5.6 Resources and Tools

In this section, we provide an overview of common data sources and tools utilized in graph RAG systems within the scientific domain, along with a brief description of each project.

### 5.6.1 Data Resources

The public datasets are commonly from well-known resources, such as molecular databases:

- **PubChem** [204]: PubChem encompasses a wide range of chemical data, including 2D and 3D structures, chemical and physical properties, bioactivity, pharmacology, toxicology, drug targets, metabolism, safety guidelines, associated patents, and scientific literature. Most of its entries pertain to small molecules, with a primary emphasis on those containing fewer than 100 atoms and 1,000 bonds.
- **ChEMBL** [121]: It is an openly accessible database containing detailed information on drugs, drug-like small molecules, and their bioactivity. This curated resource stands out for its comprehensive coverage of the drug discovery process, encompassing data on more than 2.2 million compounds and over 18 million records documenting their effects on biological systems. ChEMBL provides insights into the interactions between small molecules and their protein targets, along with data on how these compounds influence cellular and organismal functions. It also includes information on ADMET (absorption, distribution, metabolism, excretion, and toxicity) profiles. The database stores two-dimensional structures, calculated molecular properties (such as logP, molecular weight, and Lipinski’s Rule of Five parameters), and bioactivity data like binding affinities and pharmacological effects.
- **ZINC** [170]: The ZINC dataset is a curated collection of commercially available chemical compounds designed specifically for virtual screening purposes. It offers over 230 million purchasable compounds in 3D formats that are ready for docking, as well as more than 750 million compounds available for analog searches. Each molecule is adjusted to biologically relevant protonation states and is annotated with properties such as molecular weight, calculated LogP, and rotatable bonds. The library includes vendor and purchasing details, making it compatible with several widely used docking software programs. Compounds are provided in multiple protonation states and tautomeric forms within certain constraints, and some formats even offer multiple conformations per molecule. The ZINC database is available for free download<sup>19</sup> in multiple common file formats, including SMILES, mol2, 3D SDF, and DOCK flexibase.

There are also biomedical data sources as below:

---

<sup>19</sup><http://zinc.docking.org>

- **PubMed** [269]: PubMed is a freely accessible database that primarily houses the MEDLINE collection, containing references and abstracts in life sciences and biomedicine. Managed by the United States National Library of Medicine (NLM) within the National Institutes of Health, PubMed is part of the Entrez retrieval system. As of May 23, 2023, PubMed contains over 35 million citations and abstracts, with records dating back to 1966 and selectively to 1865, and a few even to 1809. On this date, approximately 24.6 million records include abstracts, and 26.8 million provide links to full-text articles, with around 10.9 million available freely. In the last decade (up to December 31, 2019), PubMed added nearly one million new records annually on average.
- **ClinicalTrials** [509]: ClinicalTrials is a global registry and database that provides information on clinical studies funded by both private and public sources. Managed by the U.S. National Library of Medicine, this resource includes a summary of each study’s protocol, and for some studies, results are available in a tabular format. Users can search studies by criteria such as study status, condition or disease, country, and other keywords. Continuously updated, the database now includes over 300,000 research studies conducted across all U.S. states and in more than 200 countries worldwide.
- **Open-source medical KGs**: CMeKG (Clinical Medicine Knowledge Graph)<sup>20</sup>, CPubMed-KG (Large-scale Chinese Open Medical Knowledge Graph)<sup>21</sup> and Disease-KG (Chinese disease Knowledge Graph)<sup>22</sup> are open-source medical knowledge graphs that consolidate a vast amount of medical text data, covering areas such as diseases, medications, symptoms, and diagnostic treatments. The combined knowledge graph includes 1,288,721 entities and 3,569,427 relations.

### 5.6.2 Tools

- **RDKit** [217]: RDKit is open-source toolkit for cheminformatics. RDKit has several key features: it can process chemical structures by reading and writing various file formats, such as SMILES, InChI, and Mol files. It generates multiple types of molecular fingerprints, enabling chemical structure comparison and similarity searches. RDKit also provides algorithms for calculating molecular similarity and supports the representation and processing of chemical reactions, including the identification of reactants and products. Although it does not offer molecular docking capabilities on its own, RDKit can be integrated with other docking tools. Additionally, RDKit supports machine learning algorithms, allowing for pattern recognition in chemical data and the construction of predictive models.
- **CADRO**: The Common Alzheimer’s and Related Dementias Research Ontology (CADRO)<sup>23</sup> is employed to extract a subset of Alzheimer’s disease (AD)-related samples from the medical QA datasets for evaluation. CADRO organizes terms into a three-tiered classification system with eight main categories and multiple subcategories focused on AD and related dementias, containing frequently used terminologies or keywords in the field. From CADRO, users obtain a list of AD-related keywords most relevant to the medical QA datasets.

## 6 Social Graph

The social graph typically consists of entities connected by their social relations and is ubiquitous across real-world applications. A primary example is social networks like Twitter and Facebook, where entities represent individuals linked by social interactions (e.g., friendships, followers/followees, likes, and mentions). These social graphs extend beyond human interactions and are not confined to living entities, such as tortoises co-using the same burrow in animal social networks [340], complementary products co-purchased by the same customer in E-commerce recommender systems [427, 452], or even the LLM-simulated social agents [523, 241]. The wealth of social-relational knowledge in these social graphs is the golden resource for GraphRAG[511, 183, 463, 429, 510, 427, 77, 95, 452, 159, 528, 327, 438, 202, 396] , which is reviewed in this section.

---

<sup>20</sup><https://cmekgpcl.ac.cn/>

<sup>21</sup><https://cpubmed.openi.org.cn/graph/wiki>

<sup>22</sup><https://github.com/nuolade/disease-kb>

<sup>23</sup><https://iadrp.nia.nih.gov/about/cadro>

## 6.1 Application Tasks

- **Entity Property Prediction:** Entity property prediction focuses on predicting properties and classifying categories for social entities in social networks, examples of which include the prediction of partnership compatibility, assessment of morality, detection of account suspensions, identification of toxic behaviors [183] and product property prediction [427].
- **Text Generation:** Text generation aims to produce text that aligns with social contexts and norms. Typically, the interplay between structural and textual information, such as proximity-based network homophily and role-based similarity [5], serves as the foundation for text generation. For instance, Wang et al. [429] retrieves the texts of proximity/role-similar nodes to enhance the text generation of the target node. Kim et al. [202], Xie et al. [463] generate personalized recommendation explanations by retrieving customers'/products' historical reviews. In addition, some other works also leverage semantic similarity to retrieve reference/attributes/opinions and augment the downstream review generation [353, 92, 315].
- **Recommendation:** The recommendation task aims to find the most relevant items to satisfy user demands. Due to the missing and sparse customer/item interactions (e.g., cold-start issues), GraphRAG can be naturally applied to boost customer/item sparse interaction by retrieving additional meta-knowledge. Depending on the concrete recommendation scenario, GraphRAG has been used for graph-based recommendation [438, 95], next-item recommendation [427, 159, 528, 327, 510, 416], and conversational recommendation [116].
- **Question-answering:** Question-answering tasks are encountered not only in knowledge and document graph domains but also in social graphs. For example, a user might ask, "What are the best parks for family gatherings around Los Gatos?" and expect a personalized query answer [511, 198]. Furthermore, such questions might explicitly require graph-structured reasoning. For instance, Wu et al. [452] build a semi-structured knowledge base, and some queries might ask, "Can you list the products made by Nike?" Answering this question requires not only a deep understanding of the query but also familiarity with the structural information of the data.
- **Fake News Detection:** Detecting fake news necessitates considering both semantic content (e.g., the content of the news) and structural interactions (e.g., interactions among news). For instance, Ram et al. [331] evaluates the credibility of a Reddit post based on the credibility of other Reddit posts retrieved by their common interactions with the same common commenters.

## 6.2 Social Graph Construction

The relations that GraphRAG leverages to derive additional information from social graphs can be mainly summarized into three rationales: proximity-based, role-based, and personalization-based rationale. The proximity-based rationale, rooted in the adage "birds of a feather flock together," suggests that nodes close to each other in a social network often share similar properties [291]. For example, close friends tend to possess similar hobbies. The role-based rationale focuses on nodes with similar local subgraph structures sharing similar features or label distributions [93]. For example, managers at the same hierarchical level within a company generally have comparable job titles and responsibilities, and hub airports exhibit similar operational characteristics and strategic importance. Lastly, the personalization-based rationale refers to individuals' uniqueness regarding their characteristics and interactions. For example, in recommender systems, users interact with items in various ways, such as clicking, viewing, adding to a cart, purchasing, and reviewing, each interaction providing valuable knowledge that GraphRAG can leverage to personalize generated content. Based on the above three relational rationales in the social graphs, the social graph construction methods can be summarized as follows:

- **User-User-Interaction** [183]: This type of social graph represents user-to-user interactions commonly seen on social networks such as Twitter, Reddit, and Facebook. Examples include follower-follower relationships on Twitter, friendship relationships on Facebook, and user comments on other users' threads on Reddit. Note that this user-user interaction naturally exists in the real world without human curation or modification compared to documents or knowledge graphs.
- **User-Item-Interaction** [427, 429, 463, 202]: This type of social graph represents user-to-item interactions commonly found on e-commerce platforms like Amazon and eBay. These interactions, including purchasing, adding-to-the-cart, and viewing, can be modeled as a bipartite graph, where each type of interaction reflects a unique user intention.

- **Item-Item-Interaction** [427, 528, 452]: This social graph captures interactions between items, typically identified by shared interactions from the same customer or user. For example, a "co-view" interaction between two products indicates that they are viewed by the same customer, while a "view-add-to-cart" interaction indicates that one product is first viewed and the other is added to the cart next by the same customer. In e-commerce networks, these co-interactions between two items can be broadly categorized into complementary and substitute relationships [554].
- **Metadata-Interaction** [452, 485]: Items and users often possess metadata. For example, products on platforms like Amazon may include brand, manufacturer, and color attributes. This metadata can be represented as additional node types and the corresponding edges indicating relations between products and attributes, such as ownership or association.
- **Agent-Agent Interaction** [51, 523]: With the increasing intelligence of LLMs, recent literature has explored the potential of using LLM-powered agents to simulate social behaviors, such as collaboration, debate, and reflection. These agent-agent interactions could also be used for GraphRAG.

Note that among the five types of interaction mentioned above, the user-user, user-item, and metadata relations are naturally formulated without human curation. In contrast, item-item interactions are generated by manual extraction, and agent-agent interactions require simulations.

### 6.3 Retriever

The relations of the social graphs constructed according to the above methods can be leveraged in GraphRAG to enhance downstream tasks by retrieving additional information. For example, retrieving historical metadata and customer interactions can improve recommendations. Next, we review representative retrieval methods in GraphRAG for social graphs.

- **ID-based Retriever:** Similar to entity linking, the ID-based retriever works by retrieving content specifically generated by a user/item, examples of which include retrieving historical item interactions of a specific customer [511, 183, 510], reviews posted on a specific product [463], and meta-data information about a specific user [452, 346].
- **Filtering-based Retriever:** Based on the ID-based retriever, the Filtering-based Retriever retrieves additional content based on collaborative filtering [416]. For user filtering, Top K users are identified by comparing the similarity of their historical item interactions with those of the target user demanding recommendation. To enhance the context of the target user, it retrieves the most popular items from those Top K users. In contrast, for item filtering, it identifies the Top K items that are most similar to the current item by comparing their user-interaction history. Among them, the most popular ones would be fetched to augment the current item recommendation.
- **Social Relational Retriever:** Like the ID-based Retriever, the Social Relational Retriever focuses on retrieving knowledge from entities sharing certain relations with the target entities on hand. For example, Du et al. [95] hierarchically retrieve neighboring texts several hops from the central node to augment the semantic information of the target user/item. Meanwhile, Wang et al. [429] retrieve texts corresponding to nodes that share high proximity-based and role-based similarity, respectively.
- **Integrated Neural-Symbolic Retriever:** This approach leverages both symbolic and neural retrievers to improve retrieval effectiveness [427, 510, 409]. The symbolic retriever retrieves information by following explicitly defined rules, such as retrieving based on identifiers, structured relationships, or interaction patterns, ensuring that the retrieved data strictly aligns with specific criteria. Meanwhile, the neural retriever complements this by using embedding-based similarity, capturing nuanced patterns and contextual relationships that may not be directly encoded in rules. Integrating them together provides a better trade-off between rule-based precision and neural-based adaptability and generalizability. For instance, Wang et al. [409, 427], Huang et al. [159], Qiu et al. [327] first retrieve K-hop neighboring products from the product knowledge graph (symbolic retriever) for products in the user session and further utilize neural-based adaptive filtering to aggregate items that are most relevant to the current sequence (i.e., neural retriever). Similarly, Zeng et al. [510] address data sparsity and heterogeneity by combining ID-based retrieval that retrieves movies based on the user ID of one party with a text-based retriever that enables movie retrieval between parties.

## 6.4 Organizer

To further enhance the retrieved content, the organizer for the social graphs employs specialized techniques beyond the typical re-ranking and filtering used in other graph domains [511, 152, 77]. For social graphs, the organizer of GraphRAG often uses Keyword Extraction, Profile Summarization, and Hierarchical Graph Aggregation to curate the retrieved content:

- **Keyword Extraction:** Keyword extraction identifies the most relevant and informative keywords from the retrieved content. These extracted keywords guide the downstream generator in prioritizing attention and reduce the risk of overwhelming LLMs with excessive context. For example, Xie et al. [463] use an embedding estimator to pinpoint keywords aligned with a personalized latent query, which are then used to generate explanations.
- **Profile Summarization:** Profile summarization creates rich and detailed user profiles that capture attributes such as age, gender, preferred and disliked genres, favorite directors, country, and language derived from users' past interactions and item metadata [438]. This enriched user data increases the informativeness of the retrieved content while safeguarding privacy by using rewritten profiles. Additionally, after retrieving related movies via user/item filtering, Wang and Lim [416] apply a three-step prompting strategy to extract features tailored to the user and select representative movies by directly instructing LLMs. These extracted features and representative movies are further used as the user profile to guide the final recommendation generation. Guo et al. [137] leverage LLMs to generate more details about the entity profile based on its keywords/short phrases from external data to aid text generation.
- **Hierarchical Graph Aggregation and Summarization:** When retrieving neighborhood textual information, the exponentially expanding receptive field at higher neighborhood layers often causes the aggregated content to exceed manageable limits, potentially diminishing the LLM's effectiveness [236]. This challenge highlights the need for hierarchical graph aggregation and summarization. The primary idea is to first summarize neighborhood information retrieved at each layer before propagating it to the next layer. This approach keeps the volume of text each node receives within consistent bounds. For instance, Du et al. [95] recursively aggregate information from neighboring nodes at higher layers and leverage LLMs to rephrase and compress the content before sharing it with relevant nodes in lower layers. Consequently, this strategy expands the receptive field while optimizing the computational budget for downstream generation tasks.

## 6.5 Generator

Once the relevant content is retrieved and organized appropriately, it is further processed by a downstream generator to produce the final content. Depending on the desired output, existing generators used for social graphs can be categorized into LLM-based text generators and Prediction-based generators. The choice between these two depends on the format of the desired output and the requirements of the social graph applications.

- **LLM-based Text Generation:** This generator is typically used when the downstream application requires text outputs [510, 463, 416, 429], such as item recommendations based on names, recommendation explanations, and review generation. Due to the probabilistic nature of the next token prediction, the generated text may not always precisely align with the desired output. To address this hallucination issue, the ground-truth text is often used for grounding the generated texts, such as matching generated items with the ground-truth items on the platform [152].
- **Prediction-based Generation:** The generator directly predicts outputs and is primarily used for non-textual tasks [95], such as item recommendation and social prediction. For instance, Graph Neural Networks have become popular choices for graph-based recommendation [95, 438] and transformers are used for item recommendation tasks [427]. Furthermore, in social property prediction [183], the generator could simply be a multilayer perception used for classification (e.g., partisanship classification) or regression (e.g., morality regression).

## 6.6 Resources and Tools

This section lists common resources and tools used for GraphRAG on social graphs. The summary and the link of are itemized as follows:

### 6.6.1 Data Resources

- **STARK-Amazon<sup>24</sup>** [452]: STARK-Amazon is a large-scale, semi-structured retrieval benchmark dataset for product search on the Amazon platform, integrating textual and relational knowledge bases. The nodes in the dataset represent products, colors, brands, and categories, while edges capture relationships like `also_bought`, `also_viewed`, `has_brand`, `has_category`, and `has_color`. Rich textual information, including product descriptions, customer reviews, and entity names, provides valuable context for retrieval tasks. Queries are generated by sampling relational templates and grounding them with specific entities, then leveraging LLMs to synthesize relevant textual and relational information. This process results in queries that capture customer interests, interpret specialized descriptions, and deduce relationships involving multiple entities within the query.
- **Amazon-Review<sup>25</sup>** [306, 146, 290, 463, 510]: The Amazon-Review dataset includes reviews with ratings, text, and helpfulness votes, along with product metadata such as descriptions, categories, price, brand, and image features. Additionally, it provides link information through “also viewed” and “also bought” graphs. This dataset has two versions: the initial release logging the review data between 1996 and 2014 [146, 290], and a more recent version that logs ongoing data in 2014 [306]. The latest version also adds new metadata, including detailed product information, bullet points, and an ethical details table. Extensive GraphRAG work [] in social graphs has leveraged this dataset to build RAG frameworks for recommendation research.
- **MovieLens<sup>26</sup>** [416, 438, 510]: The MovieLens datasets describe people’s expressed preferences for movies [143]. These preferences take the form of tuples, each resulting in a person expressing a preference (a 0-5 star rating) for a movie at a particular time. These preferences were entered through the MovieLens website — a recommender system that asks its users to give movie ratings to receive personalized movie recommendations. Side information includes movie title, year, and genre in textual format. [438] further crawl the visual content of movie posters accessible from [here](#). Note that MovieLens-100k is frequently used [438, 510] and provides users with demographic information such as user ID, age, gender, occupation, and zip code.
- **Netflix<sup>27</sup>** [438]: This dataset was constructed to support participants in the Netflix Prize. The movie rating files contain over 100 million ratings from 480 thousand randomly chosen, anonymous Netflix customers over 17 thousand movie titles. The data were collected between October 1998 and December 2005 and reflect the distribution of all ratings received during this period. The date of each rating, the title, and the year of release for each movie ID are also provided. The multi-model side information is collected through web crawling [438], which is further stored [here](#).
- **Yelp<sup>28</sup>** [463]: The Yelp dataset is a rich resource for academic research, particularly in fields like data science, natural language processing (NLP), and machine learning. This comprehensive dataset includes over 6.9 million reviews, 150,346 businesses, and 200,100 photos, covering 11 metropolitan areas. Researchers and students can explore real-world business and user data, with extensive details about businesses such as location, categories, attributes (like hours, parking, and ambiance), and aggregated check-ins. Reviews include full-text content, user ratings, and metadata, while user profiles offer insights into social interactions and behaviors, including friends, compliments, and review history. Additionally, the dataset contains 908,915 tips, providing quick recommendations and insights, and 1.2 million business attributes, offering granular information about service offerings.
- **Weibo<sup>29</sup>** [521]: The Weibo dataset offers a rich snapshot of user interactions, behaviors and social connections within the Sina Weibo platform, capturing both static and dynamic facets of the social network. Starting with 100 randomly selected seed users, the dataset scales to encompass 1.7 million users and around 0.4 billion following relationships, averaging 200 followers per user. Each user is characterized by social profile details, including name, gender, verification status, and follower/followee counts. Tweet content is provided in both original Chinese and indexed formats, supporting robust research in social network analysis, content diffusion, and user interaction dynamics, making it an invaluable resource for RAG studies.

<sup>24</sup>[https://stark.stanford.edu/dataset\\_amazon.html](https://stark.stanford.edu/dataset_amazon.html)

<sup>25</sup><https://jmcauley.ucsd.edu/data/amazon/>

<sup>26</sup><https://grouplens.org/datasets/movielens/>

<sup>27</sup><https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data>

<sup>28</sup><https://www.yelp.com/dataset>

<sup>29</sup><https://www.aminer.cn/influencelocality>

- **Brexit**<sup>30</sup> [565]: This dataset includes a portion of the X (Twitter) network, specifically the remain-leave discourse before the 2016 UK Referendum on exiting the EU. It comprises a network with 7,589 users, 532,459 directed follow relationships, and 19,963 tweets, each associated with a binary stance. The dataset is preprocessed according to [298] to assign each user a scalar value between 0 and 1, referred to as opinion, representing the average stance of the tweets retweeted by the user. The stance of each tweet is either 0 (“Remain”) or 1 (“Leave”).
- **Diginetica**<sup>31</sup> [427]: This dataset comprises user session logs from an e-commerce search engine. The data spans six months and captures user interactions, including clicks, product views, and purchases. Each user session, defined by a one-hour inactivity period, contains anonymized user IDs, hashed queries, product descriptions, metadata (price, hashed product names, image identifiers, and product categories), and log-scaled prices.
- **Yoochoose**<sup>32</sup> [427]: The Yoochoose dataset contains session data from an online European retailer. Each session records the user’s click events, with some sessions also including purchase events. Collected over several months in 2014, the data captures user interactions with the retailer’s website. The product meta-data includes categories.

### 6.6.2 Tools

- **X-Developer Platform**<sup>33</sup>: The X Developer Platform provides powerful tools and resources for developers to integrate X’s real-time, historical, and global data into their own applications. With three main products—X API, X Ads API, and X for Websites—the platform supports a wide range of use cases, from retrieving and analyzing Tweets to managing ad campaigns and embedding X content directly into websites. The X API offers endpoints for managing conversations, exploring trends, and engaging with users, while the X Ads API enables businesses to manage ads with custom targeting and analytics. X for Websites allows seamless embedding of live content to enhance website engagement. Through comprehensive documentation, libraries, and community support, the X Developer Platform enables developers to create innovative solutions using X’s data and engagement tools.
- **Reddit-API**<sup>34</sup>: Reddit is a news aggregation and discussion platform where posts are organized into “subreddits,” user-created boards moderated by the community. The Reddit API provides developers access to the site’s extensive collection of posts and comments. This free API has enabled the development of moderation tools, third-party applications, and training datasets for LLMs such as ChatGPT, Google, and Gemini. By using this API, we can query tremendous user interactions (i.e., comments and posts) and their corresponding textual contents (i.e., subreddit threads), which serve as golden resources for GraphRAG.
- **RecBole**<sup>35</sup> [553]: RecBole is an open-source, unified, and comprehensive library designed for developing and benchmarking recommendation algorithms. Built with Python and PyTorch, RecBole offers researchers a streamlined, efficient framework to experiment with over 100 recommendation algorithms across four main types: General, Sequential, Context-Aware, and Knowledge-Based Recommendations. The platform simplifies data handling by providing pre-processed copies of 43 benchmark datasets, making it easy for users to dive into model testing and development. The provided user-product interactions, as well as text-formatted user/product meta-data, could also be used for GraphRAG.

## 7 Planning and Reasoning Graph

A planning or reasoning graph characterizes the inherent logical flow among different entities, where entities typically represent concrete planning or reasoning substeps, and edges denote their logical relations. For the planning graph [356, 569], a common example is a set of API tools used to achieve certain goals, where nodes represent actions, and edges denote their relational dependencies. For reasoning graphs, a notable example is the recent proposed chain/tree/graph of thoughts

<sup>30</sup><https://github.com/somethingx01/TopicalAttentionBrexit?tab=readme-ov-file>

<sup>31</sup>[https://competitions.codalab.org/competitions/11161#learn\\_the\\_details-data2](https://competitions.codalab.org/competitions/11161#learn_the_details-data2)

<sup>32</sup><https://www.kaggle.com/datasets/chadgostopp/recsys-challenge-2015>

<sup>33</sup><https://developer.x.com/en/docs/x-api/getting-started/about-x-api>

<sup>34</sup><https://support.reddithelp.com/hc/en-us/articles/16160319875092-Reddit-API-Wiki>

<sup>35</sup><https://recbole.io/>

techniques [24, 437, 488] where each node represents a decision-making thinking step connected by the reasoning flow. The dependency constraint and reasoning flow in the planning/reasoning graphs can be naturally represented as relational knowledge, which forms the foundation for GraphRAG in fulfilling planning/reasoning tasks. This section reviews GraphRAG for the planning and reasoning graph [24, 248, 282, 342, 345, 356, 355, 368, 372, 454, 437, 477, 488, 555, 569, 142].

## 7.1 Application Tasks

The representative tasks conducted on planning and reasoning graphs are summarized as follows:

- **Sequential Plan Retrieval** [356, 355, 372, 454, 555]: As one of the most frequently encountered tasks, plan retrieval aims to retrieve the plan of steps or tools in the format of subgraphs to complete user queries. For example, given the user query "Please generate an image where a girl is reading a book, and her pose is the same as the boy in "example.jpg," then please describe the new image with your voice.", the retrieved final plan from the global plan graph would be "Post Detection" → "Pose-to-Image" → "Image-to-Text" → "Text-to-Speech."
- **Naturalistic Asynchronous Planning** [248]: In contrast to plan retrieval, which considers only dependency constraints among plans, incorporating time constraints introduces a greater challenge. Naturalistic Asynchronous Planning aims to produce a plan that meets dependency requirements and optimizes task completion efficiency, using time summation, time comparison, and constrained reasoning. For example, a user might request, "To make calzones, here are the steps and times required; please calculate the optimal plan for completion." An efficient plan would execute "roll dough," "add filling," and "fill dough" sequentially, with "preheat oven" in parallel, and then conclude with "bake."
- **Structured Commonsense Reasoning** [342]: Given a belief and an argument, structured common sense reasoning aims to infer the stance and generate/retrieve the corresponding commonsense explanation graph that explains the inferred stance.
- **Defeasible Inference** [282]: Defeasible Inference is a mode of reasoning in which, given a premise, a hypothesis may be weakened or overturned in light of new evidence. A prominent approach is to support defeasible inference through argumentation by constructing inference graphs.
- **Tool Usage** [570, 142]: Instructing LLMs to use external tools for complex real-world problems has gained increasing importance. Recent research has explored advanced planning strategies to enhance LLMs' tool-use intelligence. Notably, two approaches employ A\* search [570] and Monte Carlo Tree Search [142], both utilizing graph-structured reasoning to adaptively retrieve the next tool based on the LLM's internal evaluations and environmental feedback. These methods enable dynamic tool retrieval, refining the model's problem-solving precision and flexibility.
- **Embodied Planning** [368, 477]: Embodied planning tasks in Embodied AI involve guiding agents to perform sequences of actions based on natural language instructions and visual cues in simulated or real-world environments. These tasks, such as organizing or cleaning, challenge agents due to ambiguous instructions, limited task-specific knowledge, sparse feedback, and complex, variable action spaces.

## 7.2 Reasoning and Planning Graph Construction

Most existing methods for constructing reasoning and planning graphs begin by analyzing relational dependencies and subsequently adding edges based on these hard-coded rules. Therefore, rather than limiting our focus to reviewing this only rule-based construction method, we review various dependency categories used for edge addition.

- **Resource Dependency** [355, 372, 454]: This dependency is defined as the shared resources among different actions/decisions. For example, two tools are connected if the output of one tool matches the input of the other, enabling a seamless transition from one process to another. The decision to add edges in existing graph construction methods is made by checking whether one node's input matches another node's output (e.g., plans, tools, or some other abstract processes).
- **Temporal Dependency** [355]: This relation ensures that the sequence of events follows certain orders within the planning and reasoning process. For example, connections in some collected datasets denote the successive order between two APIs for daily life.

- **Inclusive Dependency** [277]: The dependency described indicates that two connected nodes belong to the same category or environment. For example, cobblestones and birdhouses are both part of the category of garden decorations [427]. Hypergraphs can effectively capture such belonging relationships where one entity belongs to multiple environments [111]. Furthermore, these dependencies often form hierarchical structures, where "grandparent" entities encompass "parent" entities, which in turn encompass their "children." As the depth of the hierarchy increases, the space of possible dependencies grows exponentially, presenting a significant computational challenge. To address this, many previous works have proposed encoding such hierarchies in hyperbolic space [277, 257, 527]. To the best of our knowledge, no prior research has explored inclusive dependencies in RAG systems.
- **Causal Dependency** [248]: This dependency indicates the cause-and-effect logic within the graph, where one action/decision causes the trigger of another action/decision. A long-standing example is the causal graph to encode assumptions about the data-generating process.
- **Analogy Dependency** [499, 503]: This dependency underscores analogical reasoning, where relationships take the form "A is to B as C is to D." By recognizing and leveraging such dependency, humans build on existing knowledge to forge new insights across domains. A powerful historical example is the discovery of Coulomb's Law, inspired by the analogy between gravitational forces affecting celestial bodies and electrical forces between charged particles [322].

While resource-dependent and causal relations involve a sequentially structured relation (former step/tool/decision leads to the later step/tool/decision), they are inherently different. For instance, if Tool A generates a report in PDF format and Tool B is designed to extract data from PDF files, Tool A and B share resource dependency because the output of Tool A (the PDF) matches the input of Tool B. However, this connection does not imply a direct cause-and-effect relationship between the two tools, i.e., using Tool A would not necessarily cause us to use Tool B.

### 7.3 Retriever

Retrievers for handling tasks on reasoning and planning graphs are often modeled as graph traversers. The query or task instruction locates the initial seeding nodes to initialize the graph traversal; then a traversal expands the graph's scope either until a preset budget is exhausted or specific criteria are met. A core step throughout is selecting the most relevant neighbors from all potential candidates. Based on the criterion for neighborhood selection, retrievers fall into two main categories: embedding-based methods, which prioritize neighbors based on embedding similarity, and heuristic-based methods, which use local and global reward functions to determine neighbor importance.

- **Embedding-based:** Wu et al. [454] decompose the query, subsequently perform the embedding similarity match between the concatenated subquery with the current retrieved task API and each of the existing APIs, and then select the top one from the existing neighboring APIs. It explores the strategy of training and the one without training.
- **Heuristic-based:** Compared to embedding-based methods, which rely on dedicated training data for mapping, heuristic-based methods define rules to guide the graph retriever effectively[555, 569, 142]. Zhuang et al. [569] model tool planning as a tree search algorithm, incorporating A\* search to adaptively retrieve the most promising tool for subsequent use based on accumulated and anticipated costs. Both cost functions are heuristically designed, drawing on prior literature and practical insights.
- **Thought Propagation Retrieval** [499]: Given an input problem, thought propagation retrieval prompts LLMs to propose a set of analogous problems, and then applies established prompting techniques, like Chain-of-Thought (CoT), to derive solutions. The aggregation module subsequently consolidates solutions from these analogous problems, enhancing the problem-solving process for the original input.

### 7.4 Organizer

The current GraphRAG literature on planning and reasoning graphs generally omits organizer mechanisms, as the retrieval process alone achieves sufficient precision, eliminating the need for reranking. Unlike document or knowledge graphs, which typically apply one-shot embedding-based similarity retrieval to select the top-K relevant content [452], planning and reasoning graphs

use a multi-round embedding similarity process integrated with reasoning steps, enhancing plan fidelity. Moreover, reward-based retrieval involves a sophisticated search that further boosts accuracy. Together, these high-quality strategies reduce the need for fine-grained reranking or filtering.

## 7.5 Generator

Most existing GraphRAG approaches for reasoning and planning tasks either output the retrieved plan directly or integrate it into the LLM for downstream solution generation. For instance, Wu et al. [454] outputs the retrieved graph-structured plan as the final result, while Shen et al. [356] compiles executed results from expert tools to generate the response. Similarly, after constructing a tool invocation graph, Shen et al. [355] directly prompts the LLM to generate the parameters, and Lin et al. [248] leverages the LLM to produce asynchronous plans based on task dependencies, time, and graph constraints. Notably, most of these works focus on fusing textual information, primarily using different graph structure formats (e.g., adjacency matrix, adjacency list, edge list, CSR) presented in textual form.

## 7.6 Resources and Tools

We summarize the useful resources and tools for GraphRAG on planning and reasoning graphs.

## 7.7 Data Resources

- **Hugging Face<sup>36</sup>** [355]: Hugging Face offers a wide array of AI models covering multi-modality tasks in language, vision, audio, video, and more. Each task corresponds to a tool node that handles specific input and output. If tools A and B are connected, the output type of A must match the input type of B. Thus, edges in the Hugging Face plan graph represent the resource-dependent relation. [355] firstly collects the tool repository and builds a tool graph with a collection of tools and their dependencies. Then, to generate each question, they sample a subgraph from the tool graph in the three basic formats: node, chain, and directed acyclic graph (DAG), each of which embodies a specific pattern for tool invocation. After that, the sampled subgraph is sent to LLMs to synthesize user instructions and populate the parameters for the tool subgraphs. At last, LLM-based and rule-based self-critic mechanisms are used to check and filter out the generated instruction to guarantee quality.
- **Multimedia<sup>37</sup>** [355]: Unlike the AI-focused tools of Hugging Face, multimedia tools serve a broader range of user-centric tasks such as file downloading and video editing. The edges remain consistent with the Hugging Face domain and the tool connections denote the resource-dependent relation similar to Hugging Face. The construction of Multimedia is similar to Hugging Face.
- **Daily Life APIs<sup>38</sup>** [355]: Daily life services, including web search and shopping, can also be viewed as tools for specific tasks. The dependencies among these APIs are primarily temporal, meaning that two daily life APIs are connected if one follows the other in sequence. The construction of Daily Life APIs is mostly similar to Hugging Face except that the edges are constructed manually because of the scarcity of publicly available APIs.
- **RestBench<sup>39</sup>** [372, 454]: A dataset consisting of multiple APIs to address complex real-world user instructions in two scenarios: TMDB movie database and Spotify music player. The TMDB movie database offers RESTful APIs encompassing information about movies, TVs, actors, and images. Spotify Music Player provides API endpoints to retrieve content metadata, receive recommendations, create and manage playlists, and control playback. For RestBench, [372] employed NLP experts to brainstorm instructions for different combinations of APIs and correspondingly annotate the gold API solution path for each instruction. Two additional experts are further employed to thoroughly verify the solvability of each instruction and the correctness of the corresponding solution path. To adapt RestBench into a graph-structured dataset, [454] treats each API as a unique task node, modeling their relationships through two key dimensions: (1) categorical association and (2) resource dependencies. For instance, APIs offering movie-related functionalities, such as

---

<sup>36</sup><https://github.com/microsoft/JARVIS>

<sup>37</sup><https://github.com/microsoft/JARVIS>

<sup>38</sup><https://github.com/microsoft/JARVIS>

<sup>39</sup><https://restgpt.github.io/>

retrieving movie details or recommending films, are grouped under the 'movie' category, while APIs focused on person-related tasks, like searching for actors, are classified under the 'person' category. Additionally, if two APIs share a common parameter (e.g., movie-id), a link is established to represent resource dependency. To further enhance semantic distinction, GPT-4 is prompted to assign a unique name and detailed functional description to each API.

- **AsyncHow<sup>40</sup>** [248]: A curated dataset consisting of 1.6K data points for asynchronous planning. Each data point comprises a user instruction specifying tasks with their basic execution constraints, represented by a directed acyclic graph (DAG). In these DAGs, nodes denote actions, and edges represent ordering constraints. Each edge carries a weight indicating the time required to complete the preceding action and transition to the next. Additionally, edges also signify causal links, meaning an action can only proceed if all preceding linked actions are completed. To construct this dataset, we first collected planning tasks from WikiHow [214]. LLMs were then used for preprocessing, time annotation, and step dependency annotation. Specifically, plans containing optional steps, irrelevant tasks, or those lacking quantifiable duration were filtered out. The GPT-3.5 is used to estimate the time duration for each step, and the GPT -4 is used to annotate step dependencies using the DOT language. To generate natural language questions with execution constraints, ten trivially different but plausible templates were used to paraphrase the graph-structured dot language into human-understandable texts. The optimal time duration for a plan was calculated by determining the longest path within the DAG representation of the workflow.
- **EXPLAGRAPH<sup>41</sup>** [342]: Give the initially collected triplets consisting of belief, argument, and stance, the commonsense explanation graphs are constructed through a generic create-verify-refine iterative framework. Firstly, annotators construct a commonsense-augmented explanation graph that explicitly explains the stance. Each graph comprises 3-8 facts, each of which is a triplet with two concepts as entities connected by their relations. The graphical representation allows us to automatically perform in-browser checks for structural constraints, thereby guaranteeing the structural correctness of the graph. To further ensure the semantic correctness of the constructed graph, annotators reason through the graph to infer the stance based solely on the belief and the explanation graph. Finally, for incorrect graphs, another annotator refines them either by adding a new fact, removing an existing one, or replacing an existing fact.
- **GSM8K<sup>42</sup>** [65]: The GSM8K dataset is a collection of 8.5K high-quality, linguistically diverse math word problems designed to assess multi-step reasoning for question answering. These grade school-level problems, solvable by a bright middle school student, require between 2 and 8 steps, primarily using basic arithmetic operations ( $+, -, \times, \div$ ) without needing concepts beyond early Algebra. Solutions are presented in natural language to facilitate general applicability, offering insight into the reasoning processes of large language models. This format highlights how models handle structured, step-by-step reasoning in response to real-world math problems.
- **PrOntoQA<sup>43</sup>** [348]: PrOntoQA is a question-answering dataset that provides examples featuring chains-of-thought to outline the reasoning needed for correct answers. The sentences are syntactically simple and well-suited for semantic parsing, making it valuable for formal analysis of predicted reasoning chains from large language models like GPT-3.

### 7.7.1 Tools

- **ToolBench<sup>44</sup>** [355]: Recent studies on software tool manipulation with LLMs primarily depend on closed model APIs (e.g., OpenAI), as there remains a significant performance gap between these proprietary models and available open-source LLMs. To investigate the underlying causes of this discrepancy and to advance the capabilities of open-source LLMs—particularly in tool manipulation—a benchmark named ToolBench has been developed. ToolBench includes a range of diverse software tools designed for real-world tasks and provides an accessible infrastructure for directly evaluating each model's execution success rate.

---

<sup>40</sup><https://github.com/fangru-lin/graph-llm-asynchow-plan>

<sup>41</sup><https://github.com/swarnaHub/ExplaGraphs>

<sup>42</sup><https://github.com/openai/grade-school-math>

<sup>43</sup><https://github.com/asaparov/prontoqa>

<sup>44</sup><https://github.com/sambanova/toolbench>

## 8 Tabular Graph

Tabular data is another type of structured data that is widely used in real-world applications [343] and is typically stored in relational databases [66]. Tabular data may consist of a single table containing samples and their attributes, or multiple tables that share primary and foreign keys. LLMs have been explored to process and solve tasks involving tabular data, primarily by transforming the tabular data into text through serialization [107, 365]. However, such serialization may lead to some issues: (1) In the case of a single table, the feature columns should exhibit permutation invariance, but serializing the table data may disrupt this invariance; (2) For multiple tables, one table may be connected to another through primary/foreign keys, and leveraging these relationships is crucial for various tasks. Therefore, graph structures can be a suitable representation for tabular data. Additionally, when tables contain too many rows to fit within the context window of LLMs, graphs can facilitate efficient retrieval.

### 8.1 Application Tasks

Tables are widely used in real-world scenarios to store features and relationships between data. Understanding the structure of tabular data is crucial. As a result, there are numerous tasks that can benefit from the use of tabular graphs and below we list a few representative ones.

- **Node-level tasks:** Node-level tasks include node classification and node regression, applied to tasks like cell type prediction [188], fraud detection [335, 364], outlier detection [124], and click-through rate (CTR) prediction [242, 94].
- **Link-level tasks:** Link-level tasks involve link prediction, edge classification, and edge regression. Many tabular data tasks can be modeled as link-level tasks, such as data imputation [557, 497] and recommendation [339].
- **Graph-level tasks:** Graph-level tasks aim to predict the properties of the entire graph, such as table type classification and table similarity prediction [432, 178, 188].
- **Table question answering:** Table QA involves generating answers by understanding and reasoning over tabular data. This task requires comprehension of both the content and their relationships within tables, making graph structures suitable for encoding such information. For example, Zhang [532], Zhang et al. [531] utilize graphs to enhance Table QA.
- **Table retrieval:** Table retrieval focuses on retrieving semantically relevant tables based on natural language queries [411, 61].

### 8.2 Tabular Graph Construction

Graphs are used in tabular data learning to model high-order feature interactions, high-order instance relationships, and relationships between instances across multiple tables. There are typically two types of nodes: instance nodes, which represent each row of a table, and feature nodes, which represent individual features. Generally, the following graphs are constructed:

- **Instance Graph:** An instance graph connects a table's rows (instances), modeling the relationships between instances. It is particularly useful for retrieving relevant instances within tabular data. There are mainly two approaches for constructing instance graphs:
  - *Rule-based methods:* Instances are connected based on predefined rules. For example, heuristics derived from expert knowledge can be used to connect instances that share certain features [267]. Expert knowledge can be leveraged for the graph construction [314]. Another common approach is similarity-based methods, such as connecting instances through K-Nearest Neighbors (KNN) [128, 102] or based on similarity exceeding a threshold [374, 44].
  - *Learnable methods:* In this approach, an instance graph is typically initialized using rules or heuristics. The edge weights are then adjusted during the learning process, allowing the model to dynamically refine the graph structure over time [194, 245, 67, 197].
- **Feature Graph:** A feature graph connects features, with edges representing correlations between pairs of features. Typically, feature relationships are modeled in a learnable manner [480, 559]. Some works construct the feature graph by leveraging feature similarity [136, 242], while others

use heuristics based on expert knowledge to establish connections [223]. Additionally, certain approaches link features if they belong to the same instance [335].

- **Instance-Feature graphs:** The instance-feature graph is a heterogeneous graph, which connects the instance with their corresponding features [135, 497, 411, 451].
- **Cell Graph:** Cell graphs treat each cell in the table as a node. Xue et al. [479] build a cell graph, where each cell node contains both spatial and logical locations attributes while the adjacency matrix represents the neighbor relation or the same-row and same-column relation between two cells.
- **Tabular Hypergraph:** A hypergraph is a generalization of a graph in which an edge, known as a hyperedge, can connect multiple nodes. Since tables are often invariant to arbitrary row and column permutations, a hypergraph is a suitable structure for modeling tabular data [506]. Specifically, Chen et al. [46] model each row and column in a table as a hyperedge for pretraining purposes. Additionally, Du et al. [94] construct a hypergraph from tabular data to capture relationships between instances effectively.

Additionally, Cvetkov-Iliev et al. [70] model table as a knowledge graph, where the feature values or rows represent nodes, and the column names represent the relations. Zhang et al. [531] build a heterogeneous graph based on various predefined relations. Wang et al. [432] construct trees for non-relational tables based on the hierarchical relations.

Previous approaches primarily focus on constructing graphs based on a single table. However, in relational databases, there are often multiple tables. A common method is to first merge these tables into a single table, and then apply the previously mentioned methods to the merged table [195]. However, this approach relies on manually joining the tables as part of a feature engineering step, which requires significant effort and substantial domain expertise. In the following, we will introduce cross-table graphs, which directly build a graph from multiple tables without the need for manual merging.

- **Cross-table Graphs:** Cross-table graphs for tabular data typically connect different instances across multiple tables, where nodes usually represent rows in each table and edges are defined by primary-foreign key relationships [115, 71, 517]. Additionally, Wang et al. [417] propose Row2N/E: if a table has a primary key (PK), each row is treated as a node; however, if there are also two foreign key (FK) columns, each row is instead represented as an edge. These graphs model relationships across multiple tables, enabling the integration of information from different perspectives to facilitate various tasks.

Additionally, Bai et al. [15] build a hypergraph based on multiple tables, where the primary or foreign key as nodes, and the nodes within the same table as hyperedge.

### 8.3 Retriever

Modeling tabular data into graphs is still in its early stage, and most works follow popular retrieval methods, as introduced in Sections 2.4. Additionally, Fey et al. [115] retrieve subgraphs based on timestamps to construct time-consistent computational graphs, while Cvitkovic [71] adopt a deterministic heuristic to select subgraphs.

### 8.4 Generator

Most existing methods that leverage tabular graphs still rely on GNNs or Graph Transformers as generators [223]. Besides, Wang et al. [417] fuse both GNNs and tabular based predictors, such as DeepFM [129], FT-Transformer [125], XGBoost [50] and AutoGluon [101]. Ivanov and Prokhorenko [171] jointly train gradient boosted decision tree (GBDT) and GNN. Recent advancements have seen the application of LLMs to tabular data tasks. This is typically achieved by converting tabular data into sequential formats suitable for LLM input, as explored by Fang et al. [107], Dong and Wang [89], and Sui et al. [378]. However, this transformation can lead to the loss of inherent structural information present in the original tabular format. With advancements in the application of LLMs to graph data, LLMs can be further enhanced to handle various tasks more effectively with the support of tabular graphs.

### 8.5 Resources and Tools

In this section, we list some data sources and tools for GraphRAG on tabular graphs.

### 8.5.1 Data Resources

Relational machine learning has recently gained popularity, leading to the development of several benchmarks and datasets for tabular data. For example:

- **RelBench** [339, 115]<sup>45</sup> is a collection of realistic, large-scale, and diverse benchmark datasets for machine learning on relational databases. It includes various tasks, such as Node-level prediction tasks (e.g., multi-class classification, multi-label classification, regression), Link prediction tasks, Temporal and static prediction tasks. The benchmark features several datasets, including rel-amazon, rel-stack, rel-trial, rel-f1, rel-hm, rel-event, and rel-avito, providing a robust platform for evaluating models across multiple relational data scenarios and tasks.
- **TabGraphs** [23] evaluates various models — including simple baselines, tabular models, and GNNs — on graphs with tabular node features. This benchmark includes several TabGraphs datasets: tolokers-tab, questions-tab, city-reviews, browser-games, hm-categories, hm-prices, city-roads-M, city-roads-L, avazu-devices, web-fraud and web-traffic.
- **Tabular-benchmark** [126]<sup>46</sup> benchmarks 45 tabular datasets from diverse domains, comparing the performance of several tree-based and deep learning models. They evaluate tasks such as numerical classification, numerical regression, categorical classification, and categorical regression.
- Schwartz-Ziv and Armon [363] benchmark 11 tabular datasets, such as MSLR [325], Forest Cover Type, Higgs Boson, and Year Prediction [96]. evaluating several tree-based models, deep learning models, and ensemble models.
- **DBInfer Benchmark** [417]<sup>47</sup> is a set of benchmarks for measuring machine learning solutions over data stored as multiple tables. It includes several large-scale relational database (RDB) datasets—such as AVS, OB, DN, RR, AB, SE, MAG, and SE—with tasks like retention, CTR, purchase, CVR, churn, rating, popularity, venue prediction, citation, charge, and prepay. The benchmark provides implementations of various baselines, including popular tabular models with and without auto-feature-engineering methods, as well as Graph Neural Networks, making it a robust resource for multi-table learning evaluations.
- **RTDL**<sup>48</sup>: RTDL (Research on Tabular Deep Learning) is a collection of papers and packages on deep learning for tabular data. It provides several popular tabular deep learning models.
- **OpenML**<sup>49</sup> is an open platform for sharing datasets, algorithms, and experiments. It provides a large collection of machine learning datasets across various domains, including many tabular datasets.
- **Kaggle**<sup>50</sup> hosts a wide array of datasets for data science competitions, making it a valuable resource for benchmarking tabular machine learning models on diverse real-world data.
- **UC Irvine Machine Learning Repository** [96]<sup>51</sup> is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms.
- **HiTab** [61] is a hierarchical table dataset for question answering and natural language generation.

### 8.5.2 Tools

In this subsection, we list several tools that provide essential functionalities for data preparation, model training, and evaluation. These tools can be effectively leveraged for GraphRAG applications on tabular graphs.

- **PyTorch Tabular** [190]<sup>52</sup>: PyTorch Tabular is a powerful library built on PyTorch designed to simplify the application of deep learning techniques to tabular data. It provides several data

<sup>45</sup><https://relbench.stanford.edu/>

<sup>46</sup><https://github.com/LeoGrin/tabular-benchmark>

<sup>47</sup><https://github.com/aws-labs/multi-table-benchmark>

<sup>48</sup><https://github.com/yandex-research/rtdl>

<sup>49</sup><https://www.openml.org/>

<sup>50</sup><https://www.kaggle.com/>

<sup>51</sup><https://archive.ics.uci.edu/>

<sup>52</sup>[https://github.com/manujosephv/pytorch\\_tabular](https://github.com/manujosephv/pytorch_tabular)

preprocessing functions, including normalization, standardization, encoding of categorical features, and dataloader preparation. Additionally, it includes a variety of tabular machine-learning models and evaluation functions, making it a comprehensive tool for handling tabular data.

- **DeepTables** [179]<sup>53</sup>: DeepTables is an easy-to-use toolkit that harnesses the power of deep learning for tabular data. Built on TensorFlow, it is designed to address classification and regression tasks on tabular data. DeepTables offers a variety of tabular models and supports AutoML.
- **PyTorch Frame** [156]<sup>54</sup>: PyTorch Frame is a deep learning extension for PyTorch, designed for heterogeneous tabular data with different column types, including numerical, categorical, time, text, and images. It provides a wide range of tabular models and supports integration with diverse model architectures, including Large Language Models.

## 9 Other Domains

While substantial GraphRAG research has been investigated in previous domains, GraphRAG remains significantly underexplored in other domains such as infrastructure, biology, and scene. Therefore, we combine them into a single comprehensive section, focusing only on key studies in these fields.

### 9.1 Infrastructure Graph

Infrastructure graphs, defined by Points of Presence (PoPs) interconnected through physical links, play a vital role in serving our daily activities across sectors such as power, water, gas, transportation, and communication. Due to the scarcity of GraphRAG research in infrastructure graphs, our review mainly focuses on graph construction methods and tasks alongside only a brief overview of recent (Graph)RAGs in this domain.

- **Power Networks** [571, 311, 247]: In power networks, nodes represent critical entities such as power plants, substations, transformers, and load points, while edges correspond to transmission and distribution lines facilitating the flow of electrical power across the network. Each node possesses features like power generation capacity, voltage level, load demand, geographical location, reliability metrics, and operational status, capturing the network's operational and spatial aspects. Edges are characterized by features such as transmission capacity, impedance, length, voltage level, and real-time power flow, all essential for analyzing line performance and flow efficiency. This graph-based representation enables a range of essential tasks, including transformer fault diagnosis, power outage prediction, power flow approximation, and power generation optimization.
- **Water Networks** [465]: In water networks, nodes are junctions (pipe connections or users) and sources (reservoirs and tanks), and edges are pipes with water flow. Basic tasks include estimating node water heads and flow in all pipes given the water network layout, pipe characteristics, nodal demands, reservoir levels, and head measurements at limited locations [32].
- **Gas Networks** [407, 486]: nodes represent connection points, gas sources, storage facilities, compressor stations and users, and edges represent natural gas pipelines, carrying user loads, diversion, and input.
- **Transportation Networks** [424, 184]: Road networks, where intersections serve as nodes and road connections as edges, are commonly used to model spatial dependencies for traffic flow and speed forecasting. Similarly, metro and bus networks use stations as nodes with routes as edges, capturing station topology. Region-level problems involve dividing cities into regular or irregular regions, represented as graph nodes, where spatial dependencies reflect land use patterns; regular regions often use grid partitions, while irregular ones use diverse methods like road- or zip-code-based partitions. At the road level, sensor, segment, intersection, and lane graphs capture different road elements, with nodes representing sensors, segments, intersections, or lanes. For station-level networks, nodes represent various transportation hubs (subway, bus, bike, or car-sharing stations) linked by natural connections like subway or road networks, creating an interconnected spatial dependency graph.

<sup>53</sup><https://github.com/DataCanvasIO/DeepTables>

<sup>54</sup><https://github.com/pyg-team/pytorch-frame>

- **Communication Networks** [113, 341]: A computer network system typically comprises two interconnected graph layers: the logical and the physical. The logical layer represents the flow of data, where traffic is routed through multiple intermediate routers before reaching its destination. In this layer, nodes correspond to routers, and edges represent the logical paths that data packets traverse. Conversely, the physical topology layer refers to the underlying infrastructure, with Points of Presence (PoPs) from providers such as Comcast and Verizon serving as nodes and the physical fiber links between them forming the edges. Studies focused on the logical layer often model networks as graphs, where nodes represent devices (e.g., switches, routers) and edges denote links (e.g., fiber-optic cables) or traffic paths [177]. This graph-based representation captures both the forwarding behavior of the network (i.e., traffic interactions) and its connectivity (i.e., topological behavior), offering valuable insights for network management. For instance, optical topology-aware traffic engineering has proven effective in mitigating issues like fiber cuts and flash crowds. For the physical layer, much previous research has focused on inferring the complete as well as aligning the physical and logical layers.

To summarize, the tasks operated on infrastructure networks could be broadly encompassed as utility prediction (e.g., flow and node performance), flow simulation and generation, vulnerability analysis, and network maintenance and operation. Managing and understanding the complex physical relationships within these graphs is essential for improving infrastructure management in key areas like service delivery, function forecasting, and network optimization. In computer networks, for instance, the state of a logical traffic routing path is directly influenced by the underlying physical fiber links. Predicting critical metrics like traffic delay and jitter depends on assessing the status of all involved fiber links [113, 341], as these metrics are highly interdependent. The intricate physical connections in infrastructure graphs create valuable opportunities for employing GraphRAG techniques to enhance infrastructure management and optimization. Although no existing works have explored GraphRAG in infrastructure, few works have leveraged RAG, which we briefly review in the next.

Hussien et al. [168] explore the use of RAG to empower automated vehicles with the ability to anticipate pedestrian and driver behaviors, including pedestrian road-crossing actions and driver lane changes. RAG retrieves explanatory documents from the JAAD and PSI datasets to provide insights into pedestrian behavior. Additionally, Qian et al. [323], Wang et al. [418], Liu et al. [250], Bariah et al. [20] build on recent advances in generative models for vision and language, proposing to harness the capabilities of LLMs within computer networking. A systematic approach has been proposed to develop a foundation model for traffic tasks, such as traffic classification and generation, treating packet hexadecimal bytes as tokens. Furthermore, the potential of generative AI in advancing telecom and networking is reviewed. Specifically, Kotaru [213] introduces an "operator's copilot," a natural language interface that leverages LLMs for efficient data retrieval. This interface helps manage thousands of counters and metrics, minimizing the need for specialists and accelerating issue resolution through data intelligence. These pioneering works in leveraging generative AI for computer networking lay a good foundation for rebuilding (Graph)RAG for infrastructure networks.

## 9.2 Single-cell Graph

Single-cell sequencing allows for the detailed analysis of molecular traits at the level of individual cells. For example, single-cell RNA sequencing (scRNA-seq) [210] quantifies RNA transcript levels, offering valuable information about cell identity, developmental stages, and functional characteristics. Single-cell assay for transposase-accessible chromatin sequencing (scATAC-seq) [29] records the number of reads per accessible chromatin region, resulting in highly dimensional data matrices containing hundreds of thousands of genomic regions. With the explosion of the number of single-cell data, a variety of deep learning approaches [299, 85, 387, 426, 442] have been developed in recent years to tackle single-cell analysis, especially GNN methods [268, 84, 441, 352, 422] have been applied to various downstream tasks. In the task of cell type annotation, sigGCN [422] builds a gene-wise weighted adjacency matrix using data from the STRING database [385] to establish a gene interaction network, with the node features representing corresponding gene expression levels. This graph is fed into a GCN-based autoencoder, which includes a convolutional layer and a max-pooling layer, followed by a flattened layer and a fully connected (FC) layer, to perform cell type annotation. scDeepSort [352] constructs a weighted bipartite graph where both cells and genes serve as nodes, with the edge weights representing the gene expression values for each cell-gene pair. Gene

node features are derived through principal component analysis (PCA), while cell node features are obtained by aggregating the weighted features of the connected gene nodes. The most common way to construct a single-cell graph is to build a KNN graph from single-cell data [414]. To be specific, the data is first normalized to make it comparable across different cells. Next, dimensionality reduction methods, such as principal component analysis (PCA) [281], are employed to reduce the dataset’s high dimensionality while retaining key information. In this lower-dimensional space, distances between cells are calculated, often using Euclidean distance. Each cell’s  $K$  nearest neighbors are then determined, and a graph is formed where cells are represented as nodes, and edges connect each cell to its  $K$  closest neighbors.

**Multi-omics single-cell graphs:** Multi-omics single-cell technologies [22] integrate multiple layers of biological information—such as gene expression (RNA), chromatin accessibility (ATAC), and protein data—at the resolution of individual cells. Multiome [375] refers to the simultaneous measurement of multiple molecular modalities, such as gene expression (RNA) and chromatin accessibility (ATAC), from the same single cell while CITE-seq (Cellular Indexing of Transcriptomes and Epitopes by Sequencing) [295] is a multi-omics technology that allows for the simultaneous measurement of gene expression and protein surface markers at the single-cell level. Taking multiome dataset as an example, scMoGNN [441] builds a heterogeneous graph consisting of cell nodes, gene nodes, and peak nodes. The edge between a cell node and a gene node indicates the expression level of that gene in the cell, while the edge between a cell node and a peak node reflects the expression level of that peak in the cell. Additionally, a threshold can be applied to filter and select the connecting edges. Additionally, pathways and gene activity can be incorporated as prior knowledge to establish connections between gene nodes, and connections between peaks and gene nodes respectively. This graph representation captures both intra-modality (within each modality, like gene expression or chromatin accessibility) and cross-modality (between gene expression and chromatin accessibility) relationships, providing a comprehensive view of regulatory dynamics at the single-cell level

**Spatial transcriptomics single-cell graphs:** Unlike traditional transcriptomics, spatial transcriptomics retains the precise location of spot representing a cell or small group of cells within the tissue. Spatial transcriptomics primarily consists of three key data sources: gene expression, which captures transcript levels in specific spots; spot location, providing the spatial coordinates of each spot within the tissue; and spot images, offering visual context and morphology of the tissue at each spot. In the task of cell type deconvolution, when building the KNN graph, GNNDeconvolver [84] integrates both cell position information and gene expression data to compute the similarity between cells. SpaGCN [154] additionally incorporate spot morphology to calculate similarity between spots. Together, constructing KNN graph with such three data sources provides a richer, more holistic view of the tissue, enabling deeper insights into cellular function, interactions, and spatial organization.

### 9.3 Scene Graph

Scene graphs are a data structure designed to capture spatial and semantic relationships between objects within a scene. They are widely used in computer vision, graphics, and robotics to organize scenes, particularly when multiple objects interact, or the scene contains complex interactions. For example, SceneGraphs [147] is a dataset for visual question answering, containing 100,000 scene graphs that describe the objects, attributes, and relationships within images. It presents tasks that test spatial reasoning and multi-step inference by asking users to answer open-ended questions based on textual descriptions derived from scene graphs.

G-Retriever [147] processes Scene Graphs by first parsing JSON data to index objects and attributes, allowing for efficient retrieval of relevant nodes and edges based on the query. It then constructs a subgraph with the necessary scene information, filtering out unrelated details. Finally, G-Retriever uses this subgraph along with an LLM to generate answers, leveraging spatial and semantic relationships for accurate reasoning and response generation.

P-RAG [477] engages with the environment through agents, building and updating a database of historical trajectories that guide the agent’s actions. Initially, the task’s goal instruction is provided. Before each action, the agent captures an observation image reflecting the current state, which is then converted into a scene graph format to facilitate processing by LLMs. By creating and retrieving scene graph context, P-RAG enhances the LLM’s ability to accurately interpret and navigate complex scenes, making this approach especially useful in planning embodied everyday tasks.

## 9.4 Random Graph

Random graphs are a foundational concept in network theory [304, 174] and are widely used for modeling and studying complex networks in fields such as computer science, physics, biology, and social sciences. They are constructed based on probabilistic rules, leading to various possible structures that reflect the diversity seen in real-world networks. These random graphs can be used to analyze the retrieval, organization, and generation processes within GraphRAG.

There are many models to generate random graphs, such as Erdős-Rényi Model (ER Model) [100], Watts–Strogatz model [436], Barabási–Albert model [7], Random geometric graph [33], scale-free networks [18] and stochastic block model [150]. Additionally, various graph structures, such as Path Graphs, Complete Graphs, Star Graphs, and Barbell Graphs, can be generated to meet specific analytical needs.

Random graph generation models, such as Contextual stochastic block models (CSBM) [80] has been widely leveraged in GNNs analysis [19, 280, 285, 140]. Additionally, random graphs are increasingly applied to examine the behavior of LLMs. For instance, Fatemi et al. [109] leverage Erdős-Rényi graphs, scale-free networks, Barabási–Albert, stochastic block model, star, path and complete graph generators to test LLM performance on various graph reasoning tasks, such as edge existence, node degree, node/edge count, connected nodes and cycle check. GraphLLM [36] generates random graphs in various input formats to assess LLM performance on graph reasoning tasks, including substructure counting, maximum triplet sum, shortest path calculation, and bipartite graph matching. In this approach, a graph transformer encodes the graph, and the embedding-fusion method is used for generation. Bachmann and Nagarajan [13] use star graphs to investigate limitations within the next-token prediction paradigm. Other studies, such as Dai et al. [73], Wang et al. [413], Guo et al. [130], and Luo et al. [275], explore the graph reasoning abilities of LLMs across various tasks.

## 10 Challenges and Future Work

After proposing the holistic framework of GraphRAG and reviewing it in each domain, we begin in this section by outlining the challenges and opportunities associated with each key component of GraphRAG, including graph construction, retriever, organizer, and generator. Then, we discuss the challenges and opportunities for GraphRAG as a holistic system with its evaluation and application.

### 10.1 Graph Construction

- **How to construct graphs?** There are numerous ways to construct graphs, yet different tasks or domains may require different graph structures. For example, deciding on the granularity of nodes and edges, as well as which entities or relationships to extract, is critical. This process may also need to address challenges such as entity disambiguation, entity alignment, and coreference resolution. Understanding when a graph structure is necessary, whether to use single or multiple graphs and how to construct them in the most appropriate way for a specific application is essential but often complex.
- **The format of graphs.** Graphs can be represented in various forms—are these representations equivalent, or do they offer unique advantages? Choosing the most effective representation for a given task can significantly influence performance.
- **Multi-modal Graphs.** Despite building text-based graphs, the retrieved resources, such as images, audio, or video, can be multi-modal. Constructing a cohesive graph from multi-modal data presents a significant challenge, as it requires integrating diverse data types while preserving meaningful relationships.
- **Dynamic graphs.** Many real-world scenarios involve dynamic data that evolve over time, which are essential for downstream tasks. Developing strategies to construct, update, and store graphs dynamically while maintaining both efficiency and effectiveness presents a further challenge worthy of exploration.

## 10.2 Retriever

- **Differentiating Neural and Symbolic Knowledge:** Graph-structured data usually involves two distinct types of knowledge: symbolic-formatted knowledge, such as relations in knowledge graphs, and neural-formatted knowledge, such as entity names. Developing techniques to differentiate these two types of knowledge and designing corresponding retriever strategies for retrieving these two types of knowledge is worthy of further investigation.
- **Harmonization between internal and external knowledge:** Since GraphRAG is often applied when internal knowledge alone is insufficient to address the task, assessing any overlap between internal and external knowledge using a robust calibration method is crucial. In addition, external knowledge may sometimes conflict with internal knowledge. To handle this, it is essential to design an effective knowledge validation and reconciliation mechanism, allowing selective retrieval and updating of external content as needed.
- **Trade-off among Accuracy, Diversity, and Novelty:** Real-world user retrieval often involves complex intentions. Beyond delivering accurate content to ensure high utility—such as achieving high question-answering accuracy—there may also be a demand for diversity and novelty in the retrieved information. Balancing retrieved content’s accuracy, diversity, and novelty remains an open-ended challenge.
- **Reasoning, planning, and thinking along the way:** A real-world retriever may need to dynamically and adaptively update its retrieve process in response to both the initial query and the content retrieved along the way. The question of how to equip the retriever with these adaptive thinking, reasoning, and planning abilities remains a big problem.

## 10.3 Organizer

- **Balancing Completeness and Conciseness.** The retrieved graphs may be large, potentially containing significant information that is not related to the query. The Organizer should balance the need for complete information with the risk of overwhelming the model. This requires techniques for pruning irrelevant nodes and edges while retaining essential context, which is especially challenging in large or noisy graphs. Additionally, some knowledge may already be captured by LLMs, raising the question of how to identify and remove redundant information to further improve efficiency.
- **Optimal Data Structuring:** Determining the most effective way to structure the retrieved content is a challenge. For example, deciding how to convert a structured graph into a format that the generator can leverage, how to arrange the order of retrieved content, and how to preserve the original data structure for structure-sensitive tasks are all important considerations. Different tasks may benefit from different structuring methods.
- **Aligning Different Resources:** Retrieved content may come from various sources and in diverse formats, such as text, graphs, and images. Aligning these components effectively to help the generator poses a significant challenge, especially when integrating multiple data modalities.
- **Data Augmentation:** Incorporating data augmentation within the Organizer involves enriching the retrieved graph content with nodes, edges, or features to improve the robustness of the model and improve downstream tasks. However, this process requires balancing real and augmented data to avoid introducing irrelevant or redundant information.

## 10.4 Generator

- **Correct Format for Prompting.** The content retrieved after organization varies significantly in format — such as texts, triplets, or graphs — while current LLMs can only process text inputs. Exploring the most effective format for optimal LLM performance on specific tasks and designing more flexible generators that go beyond text input could be worthwhile avenues for research.
- **Structural Encoding.** When the retrieved content is a subgraph and the downstream generator is an LLM, ensuring that the LLM can interpret the graph’s structural information is essential. Designing effective structural encodings and integrating them into token embeddings pose a key challenge. Although some previous works have incorporated various graph encodings into the

text decoding process, no systematic study has yet demonstrated whether LLMs can distinguish between these encodings and accurately recognize their corresponding geometric structures.

## 10.5 GraphRAG as a System

Rather than individual components, GraphRAG is a system. Designing an efficient and cohesive GraphRAG system presents additional challenges:

- **Integration Across Components:** Ensuring seamless interaction among the Graph Construction, Retriever, Organizer, and Generator components is essential. Each part must operate harmoniously to maintain efficiency and accuracy, which can be challenging to optimize the system globally.
- **Scalability:** As data volumes and query demands increase, each component of the GraphRAG system, such as Graph Construction, Retriever, Organizer, and Generator, must efficiently handle larger, more complex graphs without compromising performance. For example, efficient graph storage, optimized querying (e.g., subgraph sampling and pathfinding), streamlined organization of retrieved components, and responsive generation are all essential. Additionally, training, serving, fine-tuning, and evaluating within GraphRAG demand sophisticated engineering on modern hardware and software stacks, including efficient training, data-efficient fine-tuning, communication-efficient algorithms, implementation of reinforcement learning with human feedback, GPU acceleration and other specialized hardware, model compression for deployment, and online maintenance.
- **Trustworthiness:** Real-world GraphRAG systems often operate in high-stakes domains such as education [394], healthcare [466, 507], and law [446], which imposes multiple desires when deploying GraphRAG systems. Therefore, deeply understanding the broader desires beyond merely optimizing utility is essential to ensure effective deployment. As motivated by trustworthiness and safety research in other fields [39, 166, 420], these key goals beyond utility include reliability, robustness, fairness, and privacy. Although previous works have initiated the exploration of this multi-purpose optimization of RAG [405, 561], they mostly focus on RAG without dedicated investigation of how the relational information captured by GraphRAG could cause additional trustworthy and safety concerns.
  - **Reliability** [215, 328, 376, 495, 235, 305]: Reliability requires the system to deliver consistently low error rates across different scenarios. In RAGs, uncertainty quantification presents two primary challenges: Firstly, during the generation phase, uncertainty stems from the inherent probabilistic generation of LLMs. Standard techniques such as conformal prediction have been applied here with few adaptations [215, 328, 376, 495]. The general idea is to estimate the non-conformal score based on the calibration set and filter out those high-risk answers during the testing phase. Secondly, the retriever and its interaction with LLMs (e.g., treating LLMs as the reasoning agent to perform adaptive retrieval) also possess uncertainty [235, 305]. Unlike RAGs, retrievers and generators in GraphRAG demonstrate the multi-hop nature, which raises new requirements for multi-stage uncertainty quantification. For example, the error rate of one-step generation might accumulate after multi-step generation, and how to calibrate this accumulated error at the global level is the key challenge. The very first work [305] proposed the learn-then-test framework to guarantee the global error rate. However, this solution may not consider the uncertainty of human-LLM interactions. Moreover, the additional learn-then-test component takes additional time and computational load for calibration. Future research aims to develop methods that can quantify and calibrate the uncertainty of (Graph)RAG as a system holistically.
  - **Robustness** [105, 460, 474, 496, 97, 542, 564]: Robustness aims the system to deliver equal-quality response under extreme scenarios such as the presence of noisy and irrelevant content. Existing research on RAGs has highlighted the significant degradation in LLMs' performance when retrieved content includes noise or irrelevant information [496, 105]. Some studies address this challenge by adversarially training LLMs in noisy environments. However, there has been limited exploration of LLMs' robustness from the structural perspective. For example, would the LLM reasoning performance change when the underlying reasoning graph gets perturbed [97, 542, 564]? Addressing these issues requires a deeper focus on the interplay

between structural integrity and the robustness of GraphRAG systems, paving the way for robust performance in tasks requiring complex reasoning and planning.

- **Safety** [573, 478, 435, 79, 460]: Recent studies have highlighted that LLMs are susceptible to various adversarial attacks. Techniques such as prompt manipulation, hint injection, and input perturbation enable attackers to bypass safety mechanisms and exploit vulnerabilities, posing substantial social risks. RAGs that combine the power of LLMs with external databases introduce unique safety challenges. Existing works in RAGs have developed several threats (e.g., adversarial passage injection [573], group-query targeting [478], and jailbreak attacks [435, 79]) and defense strategies (such as isolate-then-aggregate strategy [460]). However, all of them have overlooked the structural vulnerabilities inherent in graph-structured data. For instance, attackers can exploit the network structure of graph-based databases by leveraging principles from network science to design highly impactful attacks. Attackers can maximize their influence on the system’s behavior by strategically targeting nodes with specific structural properties, such as high degree or centrality. This highlights the need for robust defenses that account for both content-based and structural threats in RAG systems.
- **Privacy** [41, 505, 324]: Privacy is a critical concern in RAG systems [513, 561], especially when operating in domains involving sensitive or personal data, such as healthcare, education, or finance [489, 74, 301]. Unlike traditional RAG systems, GraphRAG introduces additional privacy risks due to the relational nature of graphs, which may inadvertently reveal private information through connections or patterns. For instance, even if a sensitive node is protected, it may still be retrieved indirectly via its neighbors. Additionally, in graphs exhibiting homophily—where connected nodes tend to share similar attributes—sensitive information can be inferred from neighboring nodes. The use of GNNs for graph encoding further exacerbates these risks. The message-passing mechanism in GNNs propagates features across nodes and edges, potentially leading to the leakage of sensitive or confidential information during the propagation process [534]. Addressing these challenges requires advanced privacy-preserving techniques that account for the interconnected nature of graphs, rather than treating the data as independent and identically distributed (iid). Privacy protection must be designed across the entire GraphRAG pipeline, from graph construction to generation, ensuring that sensitive information remains secure while preserving the graph’s utility for downstream tasks.
- **Explainability**: Explainability is essential for fostering trust in RAG systems, especially in high-stakes domains like law, healthcare, and finance [550, 31, 500, 99]. Compared to traditional RAG, GraphRAG offers enhanced explainability through the explicit relationships encoded between nodes in the graph [2]. These explicit connections allow the system to generate clear and interpretable explanations tailored to specific tasks, making it more transparent and trustworthy for end users. For example, in the multi-hop question answering, the GraphRAG can be leveraged to generate reasoning paths to the question [272, 406, 381, 185]. These paths provide step-by-step explanations, allowing users to understand how intermediate conclusions were derived and verify the relevance and correctness of the system’s logic. Similarly, in molecular property prediction, specific subgraphs representing functional groups or structural motifs can be utilized to explain predictions, linking molecular features to observed properties or behaviors. However, achieving such explainability requires GraphRAG to retrieve reasonable and contextually relevant subgraphs, which remains a challenge. Ensuring that explanations remain faithful to the underlying model logic while balancing comprehensiveness and simplicity will be critical challenges for future research.

## 10.6 Evaluation of GraphRAG.

Evaluating the performance of a GraphRAG system is challenging due to its complex, multi-component nature. Standard benchmarks may not fully capture the nuances of graph-based construction, retrieval, organization, and generation, so tailored benchmarks are essential to assess each component and their overall impact on the system.

- **Component-Level Optimality:** Each component’s performance directly influences the GraphRAG system as a whole. Evaluating each component, such as Graph Construction, Retriever, Organizer, and Generator, requires specific designs suited to their unique roles. This may involve constructing different datasets and evaluation metrics that align with the intended function of each component.

- **End-to-End Benchmarks:** To assess the system’s overall effectiveness, comprehensive end-to-end benchmarks are essential. These should evaluate the quality of generated outputs, system response time, efficiency, and resource utilization, providing a holistic view of GraphRAG’s performance in real-world applications.
- **Task and Domain-Specific Evaluation:** Different tasks and domains may impose unique requirements on the GraphRAG system, necessitating specialized designs for each component. Methods that perform well in one domain may not generalize effectively to others, highlighting the need for diverse benchmarks. Additionally, Multi-task and multi-domain evaluations help determine the system’s adaptability and effectiveness across varied contexts.
- **Trustworthiness benchmark.** Ensuring the trustworthiness of the GraphRAG system is critical, especially in applications where decisions rely heavily on accurate and unbiased information. A trustworthiness benchmark should evaluate aspects such as robustness against adversarial inputs, data privacy protection, and transparency in how answers are derived, ensuring that outputs are explainable and reliable.

## 10.7 New Applications

While we have introduced applications of GraphRAG in several domains and tasks, there are still many domains that can leverage GraphRAG, such as Code generation [262] and robust cyber defense [330]. Extending its use to other domains is promising but presents unique challenges. Determining how to adapt GraphRAG effectively for new fields requires understanding the specific requirements, data structures, and graph configurations unique to each application area. Developing tailored strategies to construct, retrieve, organize, and generate data for diverse domains is essential for maximizing GraphRAG’s adaptability and effectiveness.

## 11 Conclusion

In this survey, we first introduced the demand and rationale behind GraphRAG, highlighting its ability to enhance retrieval-augmented generation by integrating graph-structured information. We then unified the architectural designs of existing GraphRAG approaches into a holistic framework, comprising five key components: graph construction, retriever, organizer, generator, and data source. For each component, we reviewed representative techniques. Given the diversity of graph structures and their applications across different domains, we also explored GraphRAG designs tailored to specific domains. By reviewing GraphRAG applications across varied domains—from knowledge graphs and document graphs to scientific and social graphs, we illustrated how its flexibility allows it to meet unique demands and address a wide range of tasks. Finally we discuss challenges and opportunities that have the potential to push the boundaries for GraphRAG.

## References

- [1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pages 1–3, 2024.
- [2] Hasan Abu-Rasheed, Mohamad Hussam Abdulsalam, Christian Weber, and Madjid Fathi. Supporting student decisions on learning recommendations: An llm-based chatbot with knowledge graph contextualization for conversational explainability and mentoring. *arXiv preprint arXiv:2401.08517*, 2024.
- [3] Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. *arXiv preprint arXiv:2010.12688*, 2020.
- [4] Charu C Aggarwal, Yao Li, S Yu Philip, and Yuchen Zhao. On edge classification in networks with structure and content. In *2017 IEEE 33rd international conference on data engineering (ICDE)*, pages 187–190. IEEE, 2017.
- [5] Nesreen K Ahmed, Ryan Rossi, John Boaz Lee, Theodore L Willke, Rong Zhou, Xiangnan Kong, and Hoda Eldardiry. Learning role-based graph embeddings. *arXiv preprint arXiv:1802.02896*, 2018.

- [6] Tareq Al-Moslmi, Marc Gallofré Ocaña, Andreas L Opdahl, and Csaba Veres. Named entity extraction for knowledge graphs: A literature overview. *IEEE Access*, 8:32862–32881, 2020.
- [7] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.
- [8] Ralitsa Angelova and Gerhard Weikum. Graph-based text classification: learn from your neighbors. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 485–492, 2006.
- [9] Vlad Argatu, Aaron Haag, and Oliver Lohse. Joint embeddings for graph instruction tuning. *arXiv preprint arXiv:2405.20684*, 2024.
- [10] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering. *arXiv preprint arXiv:1911.10470*, 2019.
- [11] Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46, 2023.
- [12] Hiteshwar Kumar Azad and Akshay Deepak. Query expansion techniques for information retrieval: a survey. *Information Processing & Management*, 56(5):1698–1735, 2019.
- [13] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.
- [14] Jinheon Baek, Alham Fikri Aji, and Amir Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*, 2023.
- [15] Jinze Bai, Jialin Wang, Zhao Li, Donghui Ding, Ji Zhang, and Jun Gao. Atj-net: Auto-table-join network for automatic learning on relational databases. In *Proceedings of the Web Conference 2021*, pages 1540–1551, 2021.
- [16] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.
- [17] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186, 2013.
- [18] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.
- [19] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization. *arXiv preprint arXiv:2102.06966*, 2021.
- [20] Lina Bariah, Qiyang Zhao, Hang Zou, Yu Tian, Faouzi Bader, and Merouane Debbah. Large generative ai models for telecom: The next big thing? *IEEE Communications Magazine*, 2024.
- [21] Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdellazek. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 194–199, 2024.
- [22] Alev Baysoy, Zhiliang Bai, Rahul Satija, and Rong Fan. The technological landscape and applications of single-cell multi-omics. *Nature Reviews Molecular Cell Biology*, 24(10):695–713, 2023.
- [23] Gleb Bazhenov, Oleg Platonov, and Liudmila Prokhorenkova. Tabgraphs: A benchmark and strong baselines for learning on graphs with tabular node features. *arXiv e-prints*, pages arXiv–2409, 2024.

- [24] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczek, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.
- [25] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl\_1):D267–D270, 2004.
- [26] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in neural information processing systems*, 34:2625–2640, 2021.
- [27] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.
- [28] Stephen Bonner, Ian P Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, Andreas Bender, Charles Tapley Hoyt, and William L Hamilton. A review of biomedical datasets relating to drug discovery: a knowledge graph perspective. *Briefings in Bioinformatics*, 23(6):bbac404, 2022.
- [29] Jason D Buenrostro, Paul G Giresi, Lisa C Zaba, Howard Y Chang, and William J Greenleaf. Transposition of native chromatin for multimodal regulatory analysis and personal epigenomics. *Nature methods*, 10(12):1213, 2013.
- [30] Odma Byambasuren, Yunfei Yang, Zhifang Sui, Damai Dai, Baobao Chang, Sujian Li, and Hongying Zan. Preliminary study on the construction of chinese medical knowledge graph. *Journal of Chinese Information Processing*, 33(10):1–9, 2019.
- [31] Erik Cambria, Lorenzo Malandri, Fabio Mercurio, Navid Nobani, and Andrea Seveso. Xai meets llms: A survey of the relation between explainable ai and large language models. *arXiv preprint arXiv:2407.15248*, 2024.
- [32] Antonio Candelieri, Dante Conti, and Francesco Archetti. A graph based analysis of leak localization in urban water networks. *Procedia Engineering*, 70:228–237, 2014.
- [33] Chris Cannings. Random geometric graphs, 2005.
- [34] Riccardo Cappuzzo, Saravanan Thirumuruganathan, and Paolo Papotti. Relational data imputation with graph neural networks. In *EDBT/ICDT 2024, 27th International Conference on Extending Database Technology*, 2024.
- [35] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.
- [36] Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*, 2023.
- [37] Payal Chandak, Kexin Huang, and Marinka Zitnik. Building a knowledge graph to enable precision medicine. *Scientific Data*, 10(1):67, 2023.
- [38] Heng Chang, Jiangnan Ye, Alejo Lopez-Avila, Jinhua Du, and Jia Li. Path-based explanation for knowledge graph completion. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 231–242, 2024.
- [39] April Chen, Ryan A Rossi, Namyong Park, Puja Trivedi, Yu Wang, Tong Yu, Sungchul Kim, Franck Dernoncourt, and Nesreen K Ahmed. Fairness-aware graph neural networks: A survey. *ACM Transactions on Knowledge Discovery from Data*, 18(6):1–23, 2024.

- [40] Bohan Chen and Andrea L Bertozzi. Autokg: Efficient automated knowledge graph generation for language models. In *2023 IEEE International Conference on Big Data (BigData)*, pages 3117–3126. IEEE, 2023.
- [41] Chaochao Chen, Fei Zheng, Jamie Cui, Yuwei Cao, Guanfeng Liu, Jia Wu, and Jun Zhou. Survey and open problems in privacy-preserving knowledge graph: merging, query, representation, completion, and applications. *International Journal of Machine Learning and Cybernetics*, pages 1–20, 2024.
- [42] Hung-Ting Chen, Michael JQ Zhang, and Eunsol Choi. Rich knowledge sources bring complex knowledge conflicts: Recalibrating models to reflect conflicting evidence. *arXiv preprint arXiv:2210.13701*, 2022.
- [43] Hung-Ting Chen, Fangyuan Xu, Shane A Arora, and Eunsol Choi. Understanding retrieval augmentation for long-form question answering. *arXiv preprint arXiv:2310.12150*, 2023.
- [44] Katrina Chen, Xiuqin Liang, Zheng Ma, and Zhibin Zhang. Gedi: A graph-based end-to-end data imputation framework. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 723–730. IEEE, 2023.
- [45] Moye Chen, Wei Li, Jiachen Liu, Xinyan Xiao, Hua Wu, and Haifeng Wang. Sgsum: transforming multi-document summarization into sub-graph selection. *arXiv preprint arXiv:2110.12645*, 2021.
- [46] Pei Chen, Soumajyoti Sarkar, Leonard Lausen, Balasubramaniam Srinivasan, Sheng Zha, Ruihong Huang, and George Karypis. Hytrel: Hypergraph-enhanced tabular data representation learning. *Advances in Neural Information Processing Systems*, 36, 2024.
- [47] Penghe Chen, Yu Lu, Vincent W Zheng, Xiyang Chen, and Boda Yang. Knowedu: A system to construct knowledge graph for education. *Ieee Access*, 6:31553–31563, 2018.
- [48] Qianglong Chen, Feng Ji, Haiqing Chen, and Yin Zhang. Improving commonsense question answering by graph-based iterative retrieval over multiple knowledge sources. *arXiv preprint arXiv:2011.02705*, 2020.
- [49] Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Lлага: Large language and graph assistant. *arXiv preprint arXiv:2402.08170*, 2024.
- [50] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794, 2016.
- [51] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*, 2(4):6, 2023.
- [52] Xiaohui Chen, Yinkai Wang, Jiaxing He, Yuanqi Du, Soha Hassoun, Xiaolin Xu, and Li-Ping Liu. Graph generative pre-trained transformer. *arXiv preprint arXiv:2501.01073*, 2025.
- [53] Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert systems with applications*, 141:112948, 2020.
- [54] Xinran Chen, Xuanang Chen, Ben He, Tengfei Wen, and Le Sun. Analyze, generate and refine: Query expansion with llms for zero-shot open-domain qa. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 11908–11922, 2024.
- [55] Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668*, 2023.
- [56] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61, 2024.

- [57] Dawei Cheng, Yujia Ye, Sheng Xiang, Zhenwei Ma, Ying Zhang, and Changjun Jiang. Anti-money laundering by group-aware deep graph learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12444–12457, 2023.
- [58] Kewei Cheng, Nesreen K Ahmed, Theodore Willke, and Yizhou Sun. Structure guided prompt: Instructing large language model in multi-step reasoning by exploring graph structure of the text. *arXiv preprint arXiv:2402.13415*, 2024.
- [59] Keyuan Cheng, Gang Lin, Haoyang Fei, Lu Yu, Muhammad Asif Ali, Lijie Hu, Di Wang, et al. Multi-hop question answering under temporal knowledge editing. *arXiv preprint arXiv:2404.00492*, 2024.
- [60] Yi Cheng and Xiuli Ma. scgac: a graph attentional architecture for clustering single-cell rna-seq data. *Bioinformatics*, 38(8):2187–2193, 2022.
- [61] Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. Hitab: A hierarchical table dataset for question answering and natural language generation. *arXiv preprint arXiv:2108.06712*, 2021.
- [62] Narendra Choudhary and Chandan K Reddy. Complex logical reasoning over knowledge graphs using large language models. *arXiv preprint arXiv:2305.01157*, 2023.
- [63] Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. Connecting the dots: Document-level neural relation extraction with edge-oriented graphs. *arXiv preprint arXiv:1909.00228*, 2019.
- [64] Madalina Ciortan and Matthieu Defrance. Gnn-based embedding for clustering scRNA-seq data. *Bioinformatics*, 38(4):1037–1044, 2022.
- [65] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [66] Edgar F Codd. Relational database: A practical foundation for productivity. In *ACM Turing award lectures*, page 1981. 2007.
- [67] Luca Cosmo, Anees Kazi, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael Bronstein. Latent-graph learning for disease prediction. In *Medical Image Computing and Computer Assisted Intervention—MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II* 23, pages 643–653. Springer, 2020.
- [68] Hejie Cui, Zijie Lu, Pan Li, and Carl Yang. On positional and structural node features for graph neural networks on non-attributed graphs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3898–3902, 2022.
- [69] Wanqing Cui, Keping Bi, Jiafeng Guo, and Xueqi Cheng. More: Multi-modal retrieval augmented generative commonsense reasoning. *arXiv preprint arXiv:2402.13625*, 2024.
- [70] Alexis Cvetkov-Iliev, Alexandre Allauzen, and Gaël Varoquaux. Relational data embeddings for feature enrichment with background information. *Machine Learning*, 112(2):687–720, 2023.
- [71] Milan Cvitkovic. Supervised learning on relational databases with graph neural networks. *arXiv preprint arXiv:2002.02046*, 2020.
- [72] Xinbang Dai, Yuncheng Hua, Tongtong Wu, Yang Sheng, and Guilin Qi. Counter-intuitive: Large language models can better understand knowledge graphs than we thought. *arXiv preprint arXiv:2402.11541*, 2024.
- [73] Xinnan Dai, Qihao Wen, Yifei Shen, Hongzhi Wen, Dongsheng Li, Jiliang Tang, and Caihua Shan. Revisiting the graph reasoning ability of large language models: Case studies in translation, connectivity and shortest path. *arXiv preprint arXiv:2408.09529*, 2024.
- [74] Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. Security and privacy challenges of large language models: A survey. *arXiv preprint arXiv:2402.00888*, 2024.

- [75] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*, 2018.
- [76] Nicola De Cao, Wilker Aziz, and Ivan Titov. Question answering by reasoning across documents with graph convolutional networks. *arXiv preprint arXiv:1808.09920*, 2018.
- [77] Yashar Deldjoo, Zhankui He, Julian McAuley, Anton Korikov, Scott Sanner, Arnau Ramisa, René Vidal, Maheswaran Sathiamoorthy, Atoosa Kasirzadeh, and Silvia Milano. A review of modern recommender systems using generative models (gen-recsys). *arXiv preprint arXiv:2404.00579*, 2024.
- [78] Julien Delile, Srayanta Mukherjee, Anton Van Pamel, and Leonid Zhukov. Graph-based retriever captures the long tail of biomedical knowledge. *arXiv preprint arXiv:2402.12352*, 2024.
- [79] Gelei Deng, Yi Liu, Kailong Wang, Yuekang Li, Tianwei Zhang, and Yang Liu. Pandora: Jailbreak gpts by retrieval augmented generation poisoning, 2024. URL <https://arxiv.org/abs/2402.08416>.
- [80] Yash Deshpande, Subhabrata Sen, Andrea Montanari, and Elchanan Mossel. Contextual stochastic block models. *Advances in Neural Information Processing Systems*, 31, 2018.
- [81] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [82] Bhuvan Dhingra, Qiao Jin, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Neural models for reasoning over multiple mentions using coreference. *arXiv preprint arXiv:1804.05922*, 2018.
- [83] Laura Dietz, Hannah Bast, Shubham Chatterjee, Jeffrey Dalton, Jian-Yun Nie, and Rodrigo Nogueira. Neuro-symbolic representations for information retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3436–3439, 2023.
- [84] Jiayuan Ding, Lingxiao Li, Qiaolin Lu, Julian Venegas, Yixin Wang, Lidan Wu, Wei Jin, Hongzhi Wen, Renming Liu, Wenzhuo Tang, et al. Spatialctd: A large-scale tumor microenvironment spatial transcriptomic dataset to evaluate cell type deconvolution for immuno-oncology. *Journal of Computational Biology*, 2024.
- [85] Jiayuan Ding, Renming Liu, Hongzhi Wen, Wenzhuo Tang, Zhaoheng Li, Julian Venegas, Runze Su, Dylan Molho, Wei Jin, Yixin Wang, et al. Dance: A deep learning library and benchmark platform for single-cell analysis. *Genome Biology*, 25(1):72, 2024.
- [86] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77, 2022.
- [87] Yujuan Ding, Wenqi Fan, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meets llms: Towards retrieval-augmented large language models. *arXiv preprint arXiv:2405.06211*, 2024.
- [88] Shubhansh Dohare, Harish Karnick, and Vivek Gupta. Text summarization using abstract meaning representation. *arXiv preprint arXiv:1706.01678*, 2017.
- [89] Haoyu Dong and Zhiruo Wang. Large language models for tabular data: Progresses and future directions. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2997–3000, 2024.
- [90] Jialin Dong, Bahare Fatemi, Bryan Perozzi, Lin F Yang, and Anton Tsitsulin. Don’t forget to connect! improving rag with graph-based reranking. *arXiv preprint arXiv:2405.18414*, 2024.
- [91] Kangning Dong and Shihua Zhang. Deciphering spatial domains from spatially resolved transcriptomics with an adaptive graph attention auto-encoder. *Nature communications*, 13(1):1739, 2022.

- [92] Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 623–632, 2017.
- [93] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1320–1329, 2018.
- [94] Kounianhua Du, Weinan Zhang, Ruiwen Zhou, Yangkun Wang, Xilong Zhao, Jiarui Jin, Quan Gan, Zheng Zhang, and David P Wipf. Learning enhanced representation for tabular data via neighborhood propagation. *Advances in Neural Information Processing Systems*, 35: 16373–16384, 2022.
- [95] Yingpeng Du, Ziyan Wang, Zhu Sun, Haoyan Chua, Hongzhi Liu, Zhonghai Wu, Yining Ma, Jie Zhang, and Youchen Sun. Large language model with graph convolution for recommendation. *arXiv preprint arXiv:2402.08859*, 2024.
- [96] Dheeru Dua, Casey Graff, et al. Uci machine learning repository. 2017.
- [97] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36, 2024.
- [98] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.
- [99] Jonas Elsborg and Marco Salvatore. Using llms and explainable ml to analyze biomarkers at single-cell level for improved understanding of diseases. *Biomolecules*, 13(10):1516, 2023.
- [100] P ERDdS and A R&wi. On random graphs i. *Publ. math. debrecen*, 6(290-297):18, 1959.
- [101] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.
- [102] Federico Errica. On class distributions induced by nearest neighbor graphs for node classification of tabular data. *Advances in Neural Information Processing Systems*, 36, 2024.
- [103] Peter Ertl. An algorithm to identify functional groups in organic molecules. *Journal of cheminformatics*, 9:1–7, 2017.
- [104] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 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*, pages 6491–6501, 2024.
- [105] Feiteng Fang, Yuelin Bai, Shiwen Ni, Min Yang, Xiaojun Chen, and Ruifeng Xu. Enhancing noise robustness of retrieval-augmented language models with adaptive adversarial training. *arXiv preprint arXiv:2405.20978*, 2024.
- [106] Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. Reano: Optimising retrieval-augmented reader models through knowledge graph generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2094–2112, 2024.
- [107] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Jane Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, Christos Faloutsos, et al. Large language models (llms) on tabular data: Prediction, generation, and understanding-a survey. 2024.
- [108] Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. Hierarchical graph network for multi-hop question answering. *arXiv preprint arXiv:1911.03631*, 2019.

- [109] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*, 2023.
- [110] Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, 2020.
- [111] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.
- [112] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628, 2010.
- [113] Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Krzysztof Rusek, Shihan Xiao, Xiang Shi, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Routenet-fermi: Network modeling with graph neural networks. *IEEE/ACM transactions on networking*, 31(6):3080–3095, 2023.
- [114] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.
- [115] Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Relational deep learning: Graph representation learning on relational databases. *arXiv preprint arXiv:2312.04615*, 2023.
- [116] Luke Friedman, Sameer Ahuja, David Allen, Zhenning Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, et al. Leveraging large language models in conversational recommender systems. *arXiv preprint arXiv:2305.07961*, 2023.
- [117] Yanglan Gan, Xingyu Huang, Guobing Zou, Shuigeng Zhou, and Jihong Guan. Deep structural clustering for single-cell rna-seq data jointly through autoencoder and graph neural network. *Briefings in Bioinformatics*, 23(2):bbac018, 2022.
- [118] Hanning Gao, Lingfei Wu, Po Hu, Zhihua Wei, Fangli Xu, and Bo Long. 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 (Volume 1: Long Papers)*, pages 82–92, 2022.
- [119] Yifu Gao, Linbo Qiao, Zhigang Kan, Zhihua Wen, Yongquan He, and Dongsheng Li. Two-stage generative question answering on temporal knowledge graph using large language models. *arXiv preprint arXiv:2402.16568*, 2024.
- [120] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.
- [121] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.
- [122] Yuyao Ge, Shenghua Liu, Lingrui Mei, Lizhe Chen, and Xueqi Cheng. Can graph descriptive order affect solving graph problems with llms? *arXiv preprint arXiv:2402.07140v4*, 2024.
- [123] Goran Glavaš and Jan Šnajder. Event-centered information retrieval using kernels on event graphs. In *Proceedings of TextGraphs-8 graph-based methods for natural language processing*, pages 1–5, 2013.

- [124] Adam Goodge, Bryan Hooi, See-Kiong Ng, and Wee Siong Ng. Lunar: Unifying local outlier detection methods via graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6737–6745, 2022.
- [125] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34: 18932–18943, 2021.
- [126] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35:507–520, 2022.
- [127] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.
- [128] Yifan Gu, Xuebing Yang, Lei Tian, Hongyu Yang, Jicheng Lv, Chao Yang, Jinwei Wang, Jianing Xi, Guilan Kong, and Wensheng Zhang. Structure-aware siamese graph neural networks for encounter-level patient similarity learning. *Journal of Biomedical Informatics*, 127:104027, 2022.
- [129] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.
- [130] Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*, 2023.
- [131] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568, 2020.
- [132] Taicheng Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh Chawla, Olaf Wiest, Xiangliang Zhang, et al. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in Neural Information Processing Systems*, 36:59662–59688, 2023.
- [133] Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. Knowledgenavigator: Leveraging large language models for enhanced reasoning over knowledge graph. *Complex & Intelligent Systems*, 10(5):7063–7076, 2024.
- [134] Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. Knowledgenavigator: Leveraging large language models for enhanced reasoning over knowledge graph. *Complex & Intelligent Systems*, pages 1–14, 2024.
- [135] Wei Guo, Rong Su, Renhao Tan, Hufeng Guo, Yingxue Zhang, Zhirong Liu, Ruiming Tang, and Xiuqiang He. Dual graph enhanced embedding neural network for ctr prediction. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 496–504, 2021.
- [136] Xiawei Guo, Yuhan Quan, Huan Zhao, Quanming Yao, Yong Li, and Weiwei Tu. Tabgnn: Multiplex graph neural network for tabular data prediction. *arXiv preprint arXiv:2108.09127*, 2021.
- [137] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*, 2024.
- [138] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
- [139] William L Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

- [140] Haoyu Han, Juanhui Li, Wei Huang, Xianfeng Tang, Hanqing Lu, Chen Luo, Hui Liu, and Jiliang Tang. Node-wise filtering in graph neural networks: A mixture of experts approach. *arXiv preprint arXiv:2406.03464*, 2024.
- [141] Xu Han, Tianyu Gao, Yankai Lin, Hao Peng, Yaoliang Yang, Chaojun Xiao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. More data, more relations, more context and more openness: A review and outlook for relation extraction. *arXiv preprint arXiv:2004.03186*, 2020.
- [142] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.
- [143] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.
- [144] Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 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*, pages 553–561, 2021.
- [145] Pengfei He, Zitao Li, Yue Xing, Yaling Li, Jiliang Tang, and Bolin Ding. Make llms better zero-shot reasoners: Structure-orientated autonomous reasoning. *arXiv preprint arXiv:2410.19000*, 2024.
- [146] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.
- [147] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*, 2024.
- [148] Zhongmou He, Jing Zhu, Shengyi Qian, Joyce Chai, and Danai Koutra. Linkgpt: Teaching large language models to predict missing links. *arXiv preprint arXiv:2406.04640*, 2024.
- [149] Nima Hemmati and Gholamreza Ghassem-Sani. Multi-hop question answering using sparse graphs. *Engineering Applications of Artificial Intelligence*, 126:107128, 2023.
- [150] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.
- [151] Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1):411–420, 2017.
- [152] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, pages 364–381. Springer, 2024.
- [153] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.
- [154] Jian Hu, Xiangjie Li, Kyle Coleman, Amelia Schroeder, Nan Ma, David J Irwin, Edward B Lee, Russell T Shinohara, and Mingyao Li. Spagcn: Integrating gene expression, spatial location and histology to identify spatial domains and spatially variable genes by graph convolutional network. *Nature methods*, 18(11):1342–1351, 2021.
- [155] Linmei Hu, Tianchi Yang, Luhao Zhang, Wanjun Zhong, Duyu Tang, Chuan Shi, Nan Duan, and Ming Zhou. Compare to the knowledge: Graph neural fake news detection with external knowledge. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 754–763, 2021.

- [156] Weihua Hu, Yiwen Yuan, Zecheng Zhang, Akihiro Nitta, Kaidi Cao, Vid Kocijan, Jure Leskovec, and Matthias Fey. Pytorch frame: A modular framework for multi-modal tabular learning. *arXiv preprint arXiv:2404.00776*, 2024.
- [157] Ziniu Hu, Yizhou Sun, and Kai-Wei Chang. Relation-guided pre-training for open-domain question answering. *arXiv preprint arXiv:2109.10346*, 2021.
- [158] Ziniu Hu, Yichong Xu, Wenhao Yu, Shuohang Wang, Ziyi Yang, Chenguang Zhu, Kai-Wei Chang, and Yizhou Sun. 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*, pages 9562–9581, 2022.
- [159] Chao Huang, Jiahui Chen, Lianghao Xia, Yong Xu, Peng Dai, Yanqing Chen, Liefeng Bo, Jiashu Zhao, and Jimmy Xiangji Huang. Graph-enhanced multi-task learning of multi-level transition dynamics for session-based recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4123–4130, 2021.
- [160] Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. Few-shot named entity recognition: An empirical baseline study. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 10408–10423, 2021.
- [161] Kexin Huang. scggn: scrna-seq dropout imputation via induced hierarchical cell similarity graph. *arXiv preprint arXiv:2008.03322*, 2020.
- [162] Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R Voss. Zero-shot transfer learning for event extraction. *arXiv preprint arXiv:1707.01066*, 2017.
- [163] Xingyue Huang, Miguel Romero, Ismail Ceylan, and Pablo Barceló. A theory of link prediction via relational weisfeiler-leman on knowledge graphs. *Advances in Neural Information Processing Systems*, 36, 2024.
- [164] Yongfeng Huang, Yanyang Li, Yichong Xu, Lin Zhang, Ruyi Gan, Jiaxing Zhang, and Liwei Wang. 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 (Volume 1: Long Papers)*, pages 13417–13432, 2023.
- [165] Yongjie Huang and Meng Yang. Breadth first reasoning graph for multi-hop question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5810–5821, 2021.
- [166] Yue Huang, Lichao Sun, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, et al. Position: Trustilm: Trustworthiness in large language models. In *International Conference on Machine Learning*, pages 20166–20270. PMLR, 2024.
- [167] Zhilin Huang, Ling Yang, Xiangxin Zhou, Chujun Qin, Yijie Yu, Xiawu Zheng, Zikun Zhou, Wentao Zhang, Yu Wang, and Wenming Yang. Interaction-based retrieval-augmented diffusion models for protein-specific 3d molecule generation. In *International Conference on Machine Learning*, 2024.
- [168] Mohamed Manzour Hussien, Angie Nataly Melo, Augusto Luis Ballardini, Carlota Salinas Maldonado, Rubén Izquierdo, and Miguel Ángel Sotelo. Rag-based explainable prediction of road users behaviors for automated driving using knowledge graphs and large language models. *arXiv preprint arXiv:2405.00449*, 2024.
- [169] Filip Ilievski, Pedro Szekely, and Bin Zhang. Cskg: The commonsense knowledge graph. In *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*, pages 680–696. Springer, 2021.
- [170] John J Irwin and Brian K Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182, 2005.

- [171] Sergei Ivanov and Liudmila Prokhorenkova. Boost then convolve: Gradient boosting meets graph neural networks. *arXiv preprint arXiv:2101.08543*, 2021.
- [172] Maor Ivgi, Uri Shaham, and Jonathan Berant. Efficient long-text understanding with short-text models. *Transactions of the Association for Computational Linguistics*, 11:284–299, 2023.
- [173] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*, 2023.
- [174] Svante Janson, Tomasz Luczak, and Andrzej Rucinski. *Random graphs*. John Wiley & Sons, 2011.
- [175] Minbyul Jeong, Jiwoong Sohn, Mujeen Sung, and Jaewoo Kang. Improving medical reasoning through retrieval and self-reflection with retrieval-augmented large language models. *arXiv preprint arXiv:2401.15269*, 2024.
- [176] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*, 2024.
- [177] Xingguo Ji and Qingmin Meng. Traffic classification based on graph convolutional network. In *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, pages 596–601. IEEE, 2020.
- [178] Ran Jia, Haoming Guo, Xiaoyuan Jin, Chao Yan, Lun Du, Xiaojun Ma, Tamara Stankovic, Marko Lozajic, Goran Zoranovic, Igor Ilic, et al. Getpt: Graph-enhanced general table pre-training with alternate attention network. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 941–950, 2023.
- [179] Haifeng Wu Jian Yang, Xuefeng Li. DeepTables: A Deep Learning Python Package for Tabular Data. <https://github.com/DataCanvasIO/DeepTables>, 2022. Version 0.2.x.
- [180] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- [181] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 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*, pages 9237–9251, 2023.
- [182] Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. *arXiv preprint arXiv:2402.11163*, 2024.
- [183] Julie Jiang and Emilio Ferrara. Social-lilm: Modeling user behavior at scale using language models and social network data. *arXiv preprint arXiv:2401.00893*, 2023.
- [184] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert systems with applications*, 207:117921, 2022.
- [185] Xink Jiang, Ruizhe Zhang, Yongxin Xu, Rihong Qiu, Yue Fang, Zhiyuan Wang, Jinyi Tang, Hongxin Ding, Xu Chu, Junfeng Zhao, et al. Hykge: A hypothesis knowledge graph enhanced framework for accurate and reliable medical llms responses. *arXiv preprint arXiv:2312.15883*, 2024.
- [186] Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Suhang Wang, Yu Meng, and Jiawei Han. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. *arXiv preprint arXiv:2404.07103*, 2024.
- [187] Qiao Jin, Won Kim, Qingyu Chen, Donald C Comeau, Lana Yeganova, W John Wilbur, and Zhiyong Lu. Medcpt: Contrastive pre-trained transformers with large-scale pubmed search logs for zero-shot biomedical information retrieval. *Bioinformatics*, 39(11):btad651, 2023.

- [188] Rihui Jin, Jianan Wang, Wei Tan, Yongrui Chen, Guilin Qi, and Wang Hao. Tabprompt: Graph-based pre-training and prompting for few-shot table understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7373–7383, 2023.
- [189] Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, Xiaojian Jiang, Jieixin Xu, Qiuxia Li, and Jun Zhao. Tug-of-war between knowledge: Exploring and resolving knowledge conflicts in retrieval-augmented language models. *arXiv preprint arXiv:2402.14409*, 2024.
- [190] Manu Joseph. Pytorch tabular: A framework for deep learning with tabular data, 2021.
- [191] Mingxuan Ju, Wenhao Yu, Tong Zhao, Chuxu Zhang, and Yanfang Ye. Grape: Knowledge graph enhanced passage reader for open-domain question answering. *arXiv preprint arXiv:2210.02933*, 2022.
- [192] Wei Ju, Yifan Wang, Yifang Qin, Zhengyang Mao, Zhiping Xiao, Junyu Luo, Junwei Yang, Yiyang Gu, Dongjie Wang, Qingqing Long, et al. Towards graph contrastive learning: A survey and beyond. *arXiv preprint arXiv:2405.11868*, 2024.
- [193] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.
- [194] Seokho Kang. K-nearest neighbor learning with graph neural networks. *Mathematics*, 9(8):830, 2021.
- [195] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE, 2015.
- [196] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.
- [197] Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M Bronstein. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1606–1617, 2022.
- [198] Sara Kemper, Justin Cui, Kai Dicarlantonio, Kathy Lin, Danjie Tang, Anton Korikov, and Scott Sanner. Retrieval-augmented conversational recommendation with prompt-based semi-structured natural language state tracking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2786–2790, 2024.
- [199] Imed Keraghel, Stanislas Morbieu, and Mohamed Nadif. A survey on recent advances in named entity recognition. *arXiv preprint arXiv:2401.10825*, 2024.
- [200] Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9410–9421, 2023.
- [201] Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. Factkg: Fact verification via reasoning on knowledge graphs. *arXiv preprint arXiv:2305.06590*, 2023.
- [202] Jihyeok Kim, Seungtaek Choi, Reinald Kim Amplayo, and Seung-won Hwang. Retrieval-augmented controllable review generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2284–2295, 2020.
- [203] Kiseung Kim and Jay-Yoon Lee. Re-rag: Improving open-domain qa performance and interpretability with relevance estimator in retrieval-augmented generation. *arXiv preprint arXiv:2406.05794*, 2024.
- [204] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. Pubchem 2019 update: improved access to chemical data. *Nucleic acids research*, 47(D1):D1102–D1109, 2019.

- [205] Yejin Kim, Eojin Kang, Juae Kim, and H Howie Huang. Causal reasoning in large language models: A knowledge graph approach. In *Causality and Large Models@ NeurIPS 2024*.
- [206] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [207] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.
- [208] Sungho Ko, Hyunjin Cho, Hyungjoo Chae, Jinyoung Yeo, and Dongha Lee. Evidence-focused fact summarization for knowledge-augmented zero-shot question answering. *arXiv preprint arXiv:2403.02966*, 2024.
- [209] Furkan Kocayusufoglu, Arlei Silva, and Ambuj K Singh. Flowgen: A generative model for flow graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 813–823, 2022.
- [210] Aleksandra A Kolodziejczyk, Jong Kyoung Kim, Valentine Svensson, John C Marioni, and Sarah A Teichmann. The technology and biology of single-cell rna sequencing. *Molecular cell*, 58(4):610–620, 2015.
- [211] Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. Text generation from knowledge graphs with graph transformers. *arXiv preprint arXiv:1904.02342*, 2019.
- [212] Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. Gofa: A generative one-for-all model for joint graph language modeling. *arXiv preprint arXiv:2407.09709*, 2024.
- [213] Manikanta Kotaru. Adapting foundation models for operator data analytics. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, pages 172–179, 2023.
- [214] Mahnaz Koupaei and William Yang Wang. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*, 2018.
- [215] Bhawesh Kumar, Charlie Lu, Gauri Gupta, Anil Palepu, David Bellamy, Ramesh Raskar, and Andrew Beam. Conformal prediction with large language models for multi-choice question answering, 2023. URL <https://arxiv.org/abs/2305.18404>.
- [216] Deepika Kumawat and Vinesh Jain. Pos tagging approaches: A comparison. *International Journal of Computer Applications*, 118(6), 2015.
- [217] Greg Landrum et al. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*, 8(31.10):5281, 2013.
- [218] Justine Lebrun, Sylvie Philipp-Foliguet, and Philippe-Henri Gosselin. Image retrieval with graph kernel on regions. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.
- [219] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford’s multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the fifteenth conference on computational natural language learning: Shared task*, pages 28–34, 2011.
- [220] Zhicheng Lee, Zhidian Huang, Zijun Yao, Jinxin Liu, Amy Xin, Lei Hou, and Juanzi Li. Diakop: Dialogue-based knowledge-oriented programming for neural-symbolic knowledge base question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 5234–5238, 2024.
- [221] Yibin Lei, Yu Cao, Tianyi Zhou, Tao Shen, and Andrew Yates. Corpus-steered query expansion with large language models. *arXiv preprint arXiv:2402.18031*, 2024.
- [222] Jure Leskovec, Natasa Milic-Frayling, and Marko Grobelnik. Extracting summary sentences based on the document semantic graph. 2005.

- [223] Cheng-Te Li, Yu-Che Tsai, Chih-Yao Chen, and Jay Chiehen Liao. Graph neural networks for tabular data learning: A survey with taxonomy and directions. *arXiv preprint arXiv:2401.02143*, 2024.
- [224] Dawei Li, Shu Yang, Zhen Tan, Jae Young Baik, Sunkwon Yun, Joseph Lee, Aaron Chacko, Bojian Hou, Duy Duong-Tran, Ying Ding, et al. Dalk: Dynamic co-augmentation of llms and kg to answer alzheimer’s disease questions with scientific literature. *arXiv preprint arXiv:2405.04819*, 2024.
- [225] Dawei Li, Shu Yang, Zhen Tan, Jae Young Baik, Sunkwon Yun, Joseph Lee, Aaron Chacko, Bojian Hou, Duy Duong-Tran, Ying Ding, et al. Dalk: Dynamic co-augmentation of llms and kg to answer alzheimer’s disease questions with scientific literature. *arXiv preprint arXiv:2405.04819*, 2024.
- [226] Guanghua Li, Wensheng Lu, Wei Zhang, Defu Lian, Kezhong Lu, Rui Mao, Kai Shu, and Hao Liao. Re-search for the truth: Multi-round retrieval-augmented large language models are strong fake news detectors. *arXiv preprint arXiv:2403.09747*, 2024.
- [227] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*, 2022.
- [228] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE transactions on knowledge and data engineering*, 34(1):50–70, 2020.
- [229] Liang Li, Ruiying Geng, Bowen Li, Can Ma, Yinliang Yue, Binhua Li, and Yongbin Li. Graph-to-text generation with dynamic structure pruning. *arXiv preprint arXiv:2209.07258*, 2022.
- [230] Linfeng Li, Peng Wang, Jun Yan, Yao Wang, Simin Li, Jinpeng Jiang, Zhe Sun, Buzhou Tang, Tsung-Hui Chang, Shenghui Wang, et al. Real-world data medical knowledge graph: construction and applications. *Artificial intelligence in medicine*, 103:101817, 2020.
- [231] Miao Li, Jianzhong Qi, and Jey Han Lau. Compressed heterogeneous graph for abstractive multi-document summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13085–13093, 2023.
- [232] Mufei Li, Siqi Miao, and Pan Li. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *arXiv preprint arXiv:2410.20724*, 2024.
- [233] Ronghan Li, Lifang Wang, Zejun Jiang, Zhongtian Hu, Meng Zhao, and Xinyu Lu. Mutually improved dense retriever and gnn-based reader for arbitrary-hop open-domain question answering. *Neural Computing and Applications*, 34(14):11831–11851, 2022.
- [234] Shiyang Li, Yifan Gao, Haoming Jiang, Qingyu Yin, Zheng Li, Xifeng Yan, Chao Zhang, and Bing Yin. Graph reasoning for question answering with triplet retrieval. *arXiv preprint arXiv:2305.18742*, 2023.
- [235] Shuo Li et al. Traq: Trustworthy retrieval augmented question answering via conformal prediction. *arXiv preprint arXiv:2307.04642*, 2023. URL <https://arxiv.org/abs/2307.04642>.
- [236] Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhui Chen. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*, 2024.
- [237] Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. Leveraging graph to improve abstractive multi-document summarization. *arXiv preprint arXiv:2005.10043*, 2020.
- [238] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *arXiv preprint arXiv:2305.13269*, 2023.

- [239] Yongqi Li, Wenjie Li, and Liqiang Nie. A graph-guided multi-round retrieval method for conversational open-domain question answering. *arXiv preprint arXiv:2104.08443*, 2021.
- [240] Yongqi Li, Wenjie Li, and Liqiang Nie. Dynamic graph reasoning for conversational open-domain question answering. *ACM Transactions on Information Systems (TOIS)*, 40(4):1–24, 2022.
- [241] Yuan Li, Yixuan Zhang, and Lichao Sun. Metaagents: Simulating interactions of human behaviors for llm-based task-oriented coordination via collaborative generative agents. *arXiv preprint arXiv:2310.06500*, 2023.
- [242] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 539–548, 2019.
- [243] Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. *arXiv preprint arXiv:2410.08815*, 2024.
- [244] Zhuoyang Li, Liran Deng, Hui Liu, Qiaoqiao Liu, and Junzhao Du. Unioqa: A unified framework for knowledge graph question answering with large language models. *arXiv preprint arXiv:2406.02110*, 2024.
- [245] Jay Chiehen Liao and Cheng-Te Li. Tabgsl: Graph structure learning for tabular data prediction. *arXiv preprint arXiv:2305.15843*, 2023.
- [246] Ruotong Liao, Xu Jia, Yangzhe Li, Yunpu Ma, and Volker Tresp. Gentkg: Generative forecasting on temporal knowledge graph with large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4303–4317, 2024.
- [247] Wenlong Liao, Birgitte Bak-Jensen, Jayakrishnan Radhakrishna Pillai, Yuelong Wang, and Yusen Wang. A review of graph neural networks and their applications in power systems. *Journal of Modern Power Systems and Clean Energy*, 10(2):345–360, 2021.
- [248] Fangru Lin, Emanuele La Malfa, Valentin Hofmann, Elle Michelle Yang, Anthony Cohn, and Janet B Pierrehumbert. Graph-enhanced large language models in asynchronous plan reasoning. *arXiv preprint arXiv:2402.02805*, 2024.
- [249] Bang Liu, Ting Zhang, Di Niu, Jinghong Lin, Kunfeng Lai, and Yu Xu. Matching long text documents via graph convolutional networks. *arXiv preprint arXiv:1802.07459*, pages 2793–2799, 2018.
- [250] Chang Liu, Xiaohui Xie, Xinggong Zhang, and Yong Cui. Large language models for networking: Workflow, advances and challenges. *arXiv preprint arXiv:2404.12901*, 2024.
- [251] Guangyi Liu, Yongqi Zhang, Yong Li, and Quanming Yao. Explore then determine: A gnn-llm synergy framework for reasoning over knowledge graph. *arXiv preprint arXiv:2406.01145*, 2024.
- [252] Haochen Liu, Song Wang, Yaochen Zhu, Yushun Dong, and Jundong Li. Knowledge graph-enhanced large language models via path selection. *arXiv preprint arXiv:2406.13862*, 2024.
- [253] Jerry Liu. LlamaIndex, 11 2022. URL [https://github.com/jerryjliu/llama\\_index](https://github.com/jerryjliu/llama_index).
- [254] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. Towards graph foundation models: A survey and beyond. *arXiv preprint arXiv:2310.11829*, 2023.
- [255] Jinzhe Liu, Xiangsheng Huang, Zhuo Chen, and Yin Fang. Drak: Unlocking molecular insights with domain-specific retrieval-augmented knowledge in llms. *Authorea Preprints*, 2024.
- [256] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

- [257] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *Advances in neural information processing systems*, 32, 2019.
- [258] Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. Once: Boosting content-based recommendation with both open-and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 452–461, 2024.
- [259] Tengfei Liu, Yongli Hu, Boyue Wang, Yanfeng Sun, Junbin Gao, and Baocai Yin. Hierarchical graph convolutional networks for structured long document classification. *IEEE transactions on neural networks and learning systems*, 34(10):8071–8085, 2022.
- [260] Wei Liu, Xiyan Fu, and Michael Strube. Modeling structural similarities between documents for coherence assessment with graph convolutional networks. *arXiv preprint arXiv:2306.06472*, 2023.
- [261] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908, 2020.
- [262] Xiangyan Liu, Bo Lan, Zhiyuan Hu, Yang Liu, Zhicheng Zhang, Wenmeng Zhou, Fei Wang, and Michael Shieh. Codexgraph: Bridging large language models and code repositories via code graph databases. *arXiv preprint arXiv:2408.03910*, 2024.
- [263] Yinhai Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [264] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. Graph self-supervised learning: A survey. *IEEE transactions on knowledge and data engineering*, 35(6):5879–5900, 2022.
- [265] Yuyan Liu, Sirui Ding, Sheng Zhou, Wenqi Fan, and Qiaoyu Tan. Moleculargpt: Open large language model (llm) for few-shot molecular property prediction. *arXiv preprint arXiv:2406.12950*, 2024.
- [266] Xinwei Long, Jiali Zeng, Fandong Meng, Zhiyuan Ma, Kaiyan Zhang, Bowen Zhou, and Jie Zhou. Generative multi-modal knowledge retrieval with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18733–18741, 2024.
- [267] Haohui Lu and Shahadat Uddin. A weighted patient network-based framework for predicting chronic diseases using graph neural networks. *Scientific reports*, 11(1):22607, 2021.
- [268] Qiaolin Lu, Jiayuan Ding, Lingxiao Li, Yi Chang, Jiliang Tang, and Xiaojie Qiu. Graph contrastive learning of subcellular-resolution spatial transcriptomics improves cell type annotation and reveals critical molecular pathways. *bioRxiv*, pages 2024–03, 2024.
- [269] Zhiyong Lu. Pubmed and beyond: a survey of web tools for searching biomedical literature. *Database*, 2011:baq036, 2011.
- [270] Haoran Luo, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, et al. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. *arXiv preprint arXiv:2310.08975*, 2023.
- [271] LINHAO LUO, Yuan-Fang Li, Reza Haf, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.
- [272] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*, 2023.
- [273] Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *arXiv preprint arXiv:2410.13080*, 2024.

- [274] Shitong Luo, Chence Shi, Minkai Xu, and Jian Tang. Predicting molecular conformation via dynamic graph score matching. *Advances in Neural Information Processing Systems*, 34: 19784–19795, 2021.
- [275] Zihan Luo, Xiran Song, Hong Huang, Jianxun Lian, Chenhao Zhang, Jinqi Jiang, Xing Xie, and Hai Jin. Graphinstruct: Empowering large language models with graph understanding and reasoning capability. *arXiv preprint arXiv:2403.04483*, 2024.
- [276] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Christopher Leung, Jiajie Tang, and Jiebo Luo. Llm-rec: Personalized recommendation via prompting large language models. *arXiv preprint arXiv:2307.15780*, 2023.
- [277] Qiyo Ma, Menglin Yang, Mingxuan Ju, Tong Zhao, Neil Shah, and Rex Ying. Harec: Hyperbolic graph-llm alignment for exploration and exploitation in recommender systems. *arXiv preprint arXiv:2411.13865*, 2024.
- [278] Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, and Jian Guo. Think-on-graph 2.0: Deep and interpretable large language model reasoning with knowledge graph-guided retrieval. *arXiv preprint arXiv:2407.10805*, 2024.
- [279] Yao Ma and Jiliang Tang. *Deep learning on graphs*. Cambridge University Press, 2021.
- [280] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134*, 2021.
- [281] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.
- [282] Aman Madaan, Dheeraj Rajagopal, Niket Tandon, Yiming Yang, and Eduard Hovy. Could you give me a hint? generating inference graphs for defeasible reasoning. *arXiv preprint arXiv:2105.05418*, 2021.
- [283] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, 2023.
- [284] Himanshu Maheshwari, Sambaran Bandyopadhyay, Aparna Garimella, and Anandhavelu Natarajan. Presentations are not always linear! gnn meets llm for document-to-presentation transformation with attribution. *arXiv preprint arXiv:2405.13095*, 2024.
- [285] Haitao Mao, Zhikai Chen, Wei Jin, Haoyu Han, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? *Advances in neural information processing systems*, 36, 2024.
- [286] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Michael Galkin, and Jiliang Tang. Graph foundation models. *arXiv preprint arXiv:2402.02216*, 2024.
- [287] Xuting Mao, Hao Sun, Xiaoqian Zhu, and Jianping Li. Financial fraud detection using the related-party transaction knowledge graph. *Procedia Computer Science*, 199:733–740, 2022.
- [288] Vaibhav Mavi, Anubhav Jangra, Jatowt Adam, et al. Multi-hop question answering. *Foundations and Trends® in Information Retrieval*, 17(5):457–586, 2024.
- [289] Costas Mavromatis and George Karypis. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*, 2024.
- [290] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.
- [291] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

- [292] Xin Mei, Xiaoyan Cai, Libin Yang, and Nanxin Wang. Graph transformer networks based text representation. *Neurocomputing*, 463:91–100, 2021.
- [293] Yu Meng, Yunyi Zhang, Jiaxin Huang, Xuan Wang, Yu Zhang, Heng Ji, and Jiawei Han. Distantly-supervised named entity recognition with noise-robust learning and language model augmented self-training. *arXiv preprint arXiv:2109.05003*, 2021.
- [294] R Mihalcea. *Graph-based natural language processing and information retrieval*. Cambridge University Press, 2011.
- [295] Eleni P Mimitou, Anthony Cheng, Antonino Montalbano, Stephanie Hao, Marlon Stoeckius, Mateusz Legut, Timothy Roush, Alberto Herrera, Efthymia Papalex, Zhengqing Ouyang, et al. Multiplexed detection of proteins, transcriptomes, clonotypes and crispr perturbations in single cells. *Nature methods*, 16(5):409–412, 2019.
- [296] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022.
- [297] Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*, 2019.
- [298] Marco Minici, Federico Cinus, Corrado Monti, Francesco Bonchi, and Giuseppe Manco. Cascade-based echo chamber detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1511–1520, 2022.
- [299] Dylan Molho, Jiayuan Ding, Wenzhuo Tang, Zhaoheng Li, Hongzhi Wen, Yixin Wang, Julian Venegas, Wei Jin, Renming Liu, Runze Su, et al. Deep learning in single-cell analysis. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–62, 2024.
- [300] Sai Munikoti, Anurag Acharya, Sridevi Wagle, and Sameera Horawalavithana. Atlantic: Structure-aware retrieval-augmented language model for interdisciplinary science. *arXiv preprint arXiv:2311.12289*, 2023.
- [301] Lino Murali, G Gopakumar, Daleesha M Viswanathan, and Prema Nedungadi. Towards electronic health record-based medical knowledge graph construction, completion, and applications: A literature study. *Journal of biomedical informatics*, 143:104403, 2023.
- [302] Guoshun Nan, Zhijiang Guo, Ivan Sekulić, and Wei Lu. Reasoning with latent structure refinement for document-level relation extraction. *arXiv preprint arXiv:2005.06312*, 2020.
- [303] Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. Named entity recognition and relation extraction: State-of-the-art. *ACM Computing Surveys (CSUR)*, 54(1):1–39, 2021.
- [304] Mark Newman. *Networks*. Oxford university press, 2018.
- [305] Bo Ni, Yu Wang, Lu Cheng, Erik Blasch, and Tyler Derr. Towards trustworthy knowledge graph reasoning: An uncertainty aware perspective. *arXiv preprint arXiv:2410.08985*, 2024.
- [306] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. 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)*, pages 188–197, 2019.
- [307] Yuxiang Nie, Heyan Huang, Wei Wei, and Xian-Ling Mao. Capturing global structural information in long document question answering with compressive graph selector network. *arXiv preprint arXiv:2210.05499*, 2022.
- [308] Mengjia Niu, Hao Li, Jie Shi, Hamed Haddadi, and Fan Mo. Mitigating hallucinations in large language models via self-refinement-enhanced knowledge retrieval. *arXiv preprint arXiv:2405.06545*, 2024.

- [309] Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. *arXiv preprint arXiv:2012.14610*, 2020.
- [310] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [311] Damian Owerko, Fernando Gama, and Alejandro Ribeiro. Optimal power flow using graph neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5930–5934. IEEE, 2020.
- [312] Vardaan Pahuja, Boshi Wang, Hugo Latapie, Jayanth Srinivasa, and Yu Su. A retrieve-and-read framework for knowledge graph link prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1992–2002, 2023.
- [313] Liangming Pan, Yuxi Xie, Yansong Feng, Tat-Seng Chua, and Min-Yen Kan. Semantic graphs for generating deep questions. *arXiv preprint arXiv:2004.12704*, 2020.
- [314] Sarah Parisot, Sofia Ira Ktena, Enzo Ferrante, Matthew Lee, Ricardo Guerrero, Ben Glocker, and Daniel Rueckert. Disease prediction using graph convolutional networks: application to autism spectrum disorder and alzheimer’s disease. *Medical image analysis*, 48:117–130, 2018.
- [315] Dae Hoon Park, Hyun Duk Kim, ChengXiang Zhai, and Lifan Guo. Retrieval of relevant opinion sentences for new products. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 393–402, 2015.
- [316] Jinyoung Park, Ameen Patel, Omar Zia Khan, Hyunwoo J Kim, and Joo-Kyung Kim. Graph-guided reasoning for multi-hop question answering in large language models. *arXiv preprint arXiv:2311.09762*, 2023.
- [317] Sachin Pawar, Girish K Palshikar, and Pushpak Bhattacharyya. Relation extraction: A survey. *arXiv preprint arXiv:1712.05191*, 2017.
- [318] Alexander R Pelletier, Joseph Ramirez, Irsyad Adam, Simha Sankar, Yu Yan, Ding Wang, Dylan Steinecke, Wei Wang, and Peipei Ping. Explainable biomedical hypothesis generation via retrieval augmented generation enabled large language models. *arXiv preprint arXiv:2407.12888*, 2024.
- [319] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*, 2024.
- [320] Zhuoyi Peng and Yi Yang. Connecting the dots: Inferring patent phrase similarity with retrieved phrase graphs. *arXiv preprint arXiv:2403.16265*, 2024.
- [321] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.
- [322] Joseph Priestley. “*The History and Present State of Electricity: With Original Experiments*. C. Bathurst and T. Lowndes, in Fleet-Street, J. Rivington and J. Johnson, in . . . , 1775.
- [323] Chen Qian, Xiaochang Li, Qineng Wang, Gang Zhou, and Huajie Shao. Netbench: A large-scale and comprehensive network traffic benchmark dataset for foundation models. *arXiv preprint arXiv:2403.10319*, 2024.
- [324] Jianwei Qian, Xiang-Yang Li, Chunhong Zhang, Linlin Chen, Taeho Jung, and Junze Han. Social network de-anonymization and privacy inference with knowledge graph model. *IEEE Transactions on Dependable and Secure Computing*, 16(4):679–692, 2017.

- [325] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.
- [326] Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 6140–6150, 2019.
- [327] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Transactions on Information Systems (TOIS)*, 38(3):1–23, 2020.
- [328] Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. Conformal language modeling, 2024. URL <https://arxiv.org/abs/2306.10193>.
- [329] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- [330] Moqsadur Rahman, Krish O Piryani, Aaron M Sanchez, Sai Munikoti, Luis De La Torre, Maxwell S Levin, Monika Akbar, Mahmud Hossain, Monowar Hasan, and Mahantesh Halappanavar. Retrieval augmented generation for robust cyber defense. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (United States), 2024.
- [331] Ashwin Ram, Yigit Ege Bayiz, Arash Amini, Mustafa Munir, and Radu Marculescu. Credirag: Network-augmented credibility-based retrieval for misinformation detection in reddit. *arXiv preprint arXiv:2410.12061*, 2024.
- [332] Gowtham Ramesh, Makesh Sreedhar, and Junjie Hu. Single sequence prediction over reasoning graphs for multi-hop qa. *arXiv preprint arXiv:2307.00335*, 2023.
- [333] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.
- [334] Jiahua Rao, Xiang Zhou, Yutong Lu, Huiying Zhao, and Yuedong Yang. Imputing single-cell rna-seq data by combining graph convolution and autoencoder neural networks. *Iscience*, 24(5), 2021.
- [335] Susie Xi Rao, Shuai Zhang, Zhichao Han, Zitao Zhang, Wei Min, Zhiyao Chen, Yinan Shan, Yang Zhao, and Ce Zhang. xfraud: explainable fraud transaction detection. *arXiv preprint arXiv:2011.12193*, 2020.
- [336] Leonardo FR Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426*, 2020.
- [337] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, 2004.
- [338] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.
- [339] Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E Lenssen, Yiwen Yuan, Zecheng Zhang, et al. Relbench: A benchmark for deep learning on relational databases. *arXiv preprint arXiv:2407.20060*, 2024.
- [340] Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.
- [341] Krzysztof Rusek, José Suárez-Varela, Paul Almasan, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Routenet: Leveraging graph neural networks for network modeling and optimization in sdn. *IEEE Journal on Selected Areas in Communications*, 38(10):2260–2270, 2020.

- [342] Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. Explagraphs: An explanation graph generation task for structured commonsense reasoning. *arXiv preprint arXiv:2104.07644*, 2021.
- [343] Maria Sahakyan, Zeyar Aung, and Talal Rahwan. Explainable artificial intelligence for tabular data: A survey. *IEEE access*, 9:135392–135422, 2021.
- [344] Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. Inter-sentence relation extraction with document-level graph convolutional neural network. *arXiv preprint arXiv:1906.04684*, 2019.
- [345] Md Sadman Sakib and Yu Sun. Consolidating trees of robotic plans generated using large language models to improve reliability. *arXiv preprint arXiv:2401.07868*, 2024.
- [346] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*, 2023.
- [347] Diego Sanmartin. Kg-rag: Bridging the gap between knowledge and creativity. *arXiv preprint arXiv:2405.12035*, 2024.
- [348] Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.
- [349] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.
- [350] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer, 2018.
- [351] Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-based graph document modeling. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 543–552, 2014.
- [352] Xin Shao, Haihong Yang, Xiang Zhuang, Jie Liao, Penghui Yang, Junyun Cheng, Xiaoyan Lu, Huajun Chen, and Xiaohui Fan. scdeepsort: a pre-trained cell-type annotation method for single-cell transcriptomics using deep learning with a weighted graph neural network. *Nucleic acids research*, 49(21):e122–e122, 2021.
- [353] Vasu Sharma, Harsh Vardhan Sharma, Ankita Bishnu, and Labhesh Patel. Cyclegen: Cyclic consistency based product review generator from attributes. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 426–430, 2018.
- [354] Ahsan Shehzad, Feng Xia, Shagufta Abid, Ciuyuan Peng, Shuo Yu, Dongyu Zhang, and Karin Verspoor. Graph transformers: A survey. *arXiv preprint arXiv:2407.09777*, 2024.
- [355] Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueling Zhuang. Taskbench: Benchmarking large language models for task automation. *arXiv preprint arXiv:2311.18760*, 2023.
- [356] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueling Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36, 2024.
- [357] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.
- [358] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärlí, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.
- [359] Peng Shi and Jimmy Lin. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*, 2019.

- [360] Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. Retrieval-enhanced knowledge editing for multi-hop question answering in language models. *arXiv preprint arXiv:2403.19631*, 2024.
- [361] Harry Shomer, Yao Ma, Haitao Mao, Juanhui Li, Bo Wu, and Jiliang Tang. LPFormer: An adaptive graph transformer for link prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2686–2698, 2024.
- [362] Robik Shrestha, Yang Zou, Qiuyu Chen, Zhiheng Li, Yusheng Xie, and Siqi Deng. Fairrag: Fair human generation via fair retrieval augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11996–12005, 2024.
- [363] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.
- [364] Karandeep Singh, Yu-Che Tsai, Cheng-Te Li, Meeyoung Cha, and Shou-De Lin. Graphfc: Customs fraud detection with label scarcity. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4829–4835, 2023.
- [365] Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms. *arXiv preprint arXiv:2310.10358*, 2023.
- [366] Karthik Soman, Peter W Rose, John H Morris, Rabia E Akbas, Brett Smith, Braian Peetoom, Catalina Villouta-Reyes, Gabriel Cerono, Yongmei Shi, Angela Rizk-Jackson, et al. Biomedical knowledge graph-optimized prompt generation for large language models. *Bioinformatics*, 40(9):btae560, 2024.
- [367] Sheetal S Sonawane and Parag A Kulkarni. Graph based representation and analysis of text document: A survey of techniques. *International Journal of Computer Applications*, 96(19), 2014.
- [368] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.
- [369] Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*, 2018.
- [370] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. A graph-to-sequence model for amr-to-text generation. *arXiv preprint arXiv:1805.02473*, 2018.
- [371] Qianqian Song and Jing Su. Dstg: deconvoluting spatial transcriptomics data through graph-based artificial intelligence. *Briefings in bioinformatics*, 22(5):bbaa414, 2021.
- [372] Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, et al. Restgpt: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*, 2023.
- [373] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.
- [374] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks*, 129:249–260, 2020.
- [375] Tim Stuart and Rahul Satija. Integrative single-cell analysis. *Nature reviews genetics*, 20(5): 257–272, 2019.
- [376] Jiayuan Su, Jing Luo, Hongwei Wang, and Lu Cheng. Api is enough: Conformal prediction for large language models without logit-access, 2024. URL <https://arxiv.org/abs/2403.01216>.

- [377] Ying Su, Jipeng Zhang, Yangqiu Song, and Tong Zhang. Pipenet: Question answering with semantic pruning over knowledge graphs. *arXiv preprint arXiv:2401.17536*, 2024.
- [378] Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654, 2024.
- [379] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 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*, pages 4231–4242, 2018.
- [380] Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 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)*, pages 2380–2390, 2019.
- [381] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.
- [382] Yueqing Sun, Qi Shi, Le Qi, and Yu Zhang. Jointlk: Joint reasoning with language models and knowledge graphs for commonsense question answering. *arXiv preprint arXiv:2112.02732*, 2021.
- [383] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.
- [384] John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285, 1988.
- [385] Damian Szkłarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, Lars J Jensen, and Christian von Mering. STRING v11: Protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Research*, 47(D1):D607–D613, 2019. ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/gky1131.
- [386] Hengliang Tang, Yuan Mi, Fei Xue, and Yang Cao. An integration model based on graph convolutional network for text classification. *IEEE Access*, 8:148865–148876, 2020.
- [387] Wenzhuo Tang, Hongzhi Wen, Renming Liu, Jiayuan Ding, Wei Jin, Yuying Xie, Hui Liu, and Jiliang Tang. Single-cell multimodal prediction via transformers. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2422–2431, 2023.
- [388] Wenzhuo Tang, Haitao Mao, Danial Dervovic, Ivan Brugere, Saumitra Mishra, Yuying Xie, and Jiliang Tang. Cross-domain graph data scaling: A showcase with diffusion models. *arXiv preprint arXiv:2406.01899*, 2024.
- [389] Dhaval Taunk, Lakshya Khanna, Siri Venkata Pavan Kumar Kandru, Vasudeva Varma, Charu Sharma, and Makarand Tapaswi. Grapeqa: Graph augmentation and pruning to enhance question-answering. In *Companion Proceedings of the ACM Web Conference 2023*, pages 1138–1144, 2023.
- [390] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

- [391] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- [392] Mokanarangan Thayaparan, Marco Valentino, Viktor Schlegel, and André Freitas. Identifying supporting facts for multi-hop question answering with document graph networks. *arXiv preprint arXiv:1910.00290*, 2019.
- [393] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.
- [394] Maung Thway, Jose Recatala-Gomez, Fun Siong Lim, Kedar Hippalgaonkar, and Leonard W. T. Ng. Battling botpoop using genai for higher education: A study of a retrieval augmented generation chatbots impact on learning. *arXiv preprint arXiv:2312.10997*, 2023.
- [395] Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19080–19088, 2024.
- [396] Yu Tian, Yuhao Yang, Xudong Ren, Pengfei Wang, Fangzhao Wu, Qian Wang, and Chenliang Li. Joint knowledge pruning and recurrent graph convolution for news recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 51–60, 2021.
- [397] SM Tommoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 2024.
- [398] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwala Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [399] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022.
- [400] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16:1–28, 2015.
- [401] Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9073–9080, 2020.
- [402] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [403] S Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.
- [404] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.
- [405] Sridevi Wagle, Sai Munikoti, Anurag Acharya, Sara Smith, and Sameera Horawalavithana. Empirical evaluation of uncertainty quantification in retrieval-augmented language models for science. *arXiv preprint arXiv:2311.09358*, 2023.
- [406] Chaojie Wang, Yishi Xu, Zhong Peng, Chenxi Zhang, Bo Chen, Xinrun Wang, Lei Feng, and Bo An. keqing: knowledge-based question answering is a nature chain-of-thought mentor of llm. *arXiv preprint arXiv:2401.00426*, 2023.

- [407] Chen Wang, Dengji Zhou, Xiaoguo Wang, Song Liu, Tiemin Shao, Chongyuan Shui, and Jun Yan. Multiscale graph based spatio-temporal graph convolutional network for energy consumption prediction of natural gas transmission process. *Energy*, 307:132489, 2024.
- [408] Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. Heterogeneous graph neural networks for extractive document summarization. *arXiv preprint arXiv:2004.12393*, 2020.
- [409] Difeng Wang, Wei Hu, Ermei Cao, and Weijian Sun. Global-to-local neural networks for document-level relation extraction. *arXiv preprint arXiv:2009.10359*, 2020.
- [410] Dongsheng Wang, Zhiqiang Ma, Armineh Nourbakhsh, Kang Gu, and Sameena Shah. Docgraphlm: Documental graph language model for information extraction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1944–1948, 2023.
- [411] Fei Wang, Kexuan Sun, Muhan Chen, Jay Pujara, and Pedro Szekely. Retrieving complex tables with multi-granular graph representation learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1472–1482, 2021.
- [412] Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö Arik. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *arXiv preprint arXiv:2410.07176*, 2024.
- [413] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36, 2024.
- [414] Juexin Wang, Anjun Ma, Yuzhou Chang, Jianting Gong, Yuexu Jiang, Ren Qi, Cankun Wang, Hongjun Fu, Qin Ma, and Dong Xu. scgnn is a novel graph neural network framework for single-cell rna-seq analyses. *Nature communications*, 12(1):1–11, 2021. **Code Link:** <https://github.com/juexinwang/scGNN>.
- [415] Kunze Wang, Soyeon Caren Han, Siqu Long, and Josiah Poon. Me-gcn: Multi-dimensional edge-embedded graph convolutional networks for semi-supervised text classification. *arXiv preprint arXiv:2204.04618*, 2022.
- [416] Lei Wang and Ee-Peng Lim. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153*, 2023.
- [417] Minjie Wang, Quan Gan, David Wipf, Zhenkun Cai, Ning Li, Jianheng Tang, Yanlin Zhang, Zizhao Zhang, Zunyao Mao, Yakun Song, et al. 4dbinfer: A 4d benchmarking toolbox for graph-centric predictive modeling on relational dbs. *arXiv preprint arXiv:2404.18209*, 2024.
- [418] Qineng Wang, Chen Qian, Xiaochang Li, Ziyu Yao, and Huajie Shao. Lens: A foundation model for network traffic. *arXiv preprint arXiv:2402.03646*, 2024.
- [419] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*, 2023.
- [420] Song Wang, Yushun Dong, Binchi Zhang, Zihan Chen, Xingbo Fu, Yinhan He, Cong Shen, Chuxu Zhang, Nitesh V Chawla, and Jundong Li. Safety in graph machine learning: Threats and safeguards. *arXiv preprint arXiv:2405.11034*, 2024.
- [421] Tianming Wang, Xiaojun Wan, and Hanqi Jin. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33, 2020.
- [422] Tianyu Wang, Jun Bai, and Sheida Nabavi. Single-cell classification using graph convolutional networks. *BMC Bioinformatics*, 22(1):364, 2021. ISSN 1471-2105. doi: 10.1186/s12859-021-04278-2. URL <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-021-04278-2>.

- [423] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.
- [424] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of the web conference 2020*, pages 1082–1092, 2020.
- [425] Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases. *arXiv preprint arXiv:2308.11761*, 2023.
- [426] Yixin Wang, Jiayuan Ding, Lidan Wu, Aster Wardhani, Patrick Danaher, Qiaolin Lu, Hongzhi Wen, Wenzhuo Tang, Yi Chang, Yu Leo Lei, et al. Mem-gan: A pseudo membrane generator for single-cell imaging in fluorescent microscopy. *bioRxiv*, pages 2023–11, 2023.
- [427] Yu Wang, Amin Javari, Janani Balaji, Walid Shalaby, Tyler Derr, and Xiquan Cui. Knowledge graph-based session recommendation with session-adaptive propagation. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 264–273, 2024.
- [428] Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214, 2024.
- [429] Yu Wang, Nedim Lipka, Ruiyi Zhang, Alexa Siu, Yuying Zhao, Bo Ni, Xin Wang, Ryan Rossi, and Tyler Derr. Augmenting textual generation via topology aware retrieval. *arXiv preprint arXiv:2405.17602*, 2024.
- [430] Yujie Wang, Hu Zhang, Jiye Liang, and Ru Li. Dynamic heterogeneous-graph reasoning with language models and knowledge representation learning for commonsense question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14048–14063, 2023.
- [431] Yuqi Wang, Boran Jiang, Yi Luo, Dawei He, Peng Cheng, and Liangcai Gao. Reasoning on efficient knowledge paths: Knowledge graph guides large language model for domain question answering. *arXiv preprint arXiv:2404.10384*, 2024.
- [432] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790, 2021.
- [433] Zichao Wang, Weili Nie, Zhuoran Qiao, Chaowei Xiao, Richard Baraniuk, and Anima Anandkumar. Retrieval-based controllable molecule generation. In *The Eleventh International Conference on Learning Representations*.
- [434] Zichao Wang, Weili Nie, Zhuoran Qiao, Chaowei Xiao, Richard Baraniuk, and Anima Anandkumar. Retrieval-based controllable molecule generation. *arXiv preprint arXiv:2208.11126*, 2022.
- [435] Ziqiu Wang, Jun Liu, Shengkai Zhang, and Yang Yang. Poisoned langchain: Jailbreak llms by langchain, 2024. URL <https://arxiv.org/abs/2406.18122>.
- [436] Duncan J Watts and Steven H Strogatz. Collective dynamics of ‘small-world’ networks. *nature*, 393(6684):440–442, 1998.
- [437] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [438] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 806–815, 2024.

- [439] Yanbin Wei, Qiushi Huang, Yu Zhang, and James Kwok. Kicgpt: Large language model with knowledge in context for knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8667–8683, 2023.
- [440] Hongzhi Wen, Wei Jin, Jiayuan Ding, Christopher Xu, Yuying Xie, and Jiliang Tang. Bi-channel masked graph autoencoders for spatially resolved single-cell transcriptomics data imputation. In *NeurIPS 2022 AI for Science: Progress and Promises*.
- [441] Hongzhi Wen, Jiayuan Ding, Wei Jin, Yiqi Wang, Yuying Xie, and Jiliang Tang. Graph neural networks for multimodal single-cell data integration. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 4153–4163, 2022.
- [442] Hongzhi Wen, Wenzhuo Tang, Wei Jin, Jiayuan Ding, Renming Liu, Feng Shi, Yuying Xie, and Jiliang Tang. Single cells are spatial tokens: Transformers for spatial transcriptomic data imputation. *arXiv preprint arXiv:2302.03038*, 2023.
- [443] Yilin Wen, Zifeng Wang, and Jimeng Sun. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729*, 2023.
- [444] Tim Weninger, Yonatan Bisk, and Jiawei Han. Document-topic hierarchies from document graphs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 635–644, 2012.
- [445] Daniel S Wigh, Jonathan M Goodman, and Alexei A Lapkin. A review of molecular representation in the age of machine learning. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 12(5):e1603, 2022.
- [446] Nirmalie Wiratunga, Ramitha Abeyratne, Lasal Jayawardena, Kyle Martin, Stewart Massie, Ikechukwu Nkisi-Orji, Ruvan Weerasinghe, Anne Liret, and Bruno Fleisch. Cbr-rag: Case-based reasoning for retrieval augmented generation in llms for legal question answering. In *International Conference on Case-Based Reasoning*, pages 445–460. Springer, 2024.
- [447] Eugene Wong and Karel Youssefi. Decomposition—a strategy for query processing. *ACM Transactions on Database Systems (TODS)*, 1(3):223–241, 1976.
- [448] Bo Wu, Bo Lang, and Yang Liu. Gksh: Graph based image retrieval using supervised kernel hashing. In *Proceedings of the International Conference on Internet Multimedia Computing and Service*, pages 88–93, 2016.
- [449] Junde Wu, Jiayuan Zhu, and Yunli Qi. Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation. *arXiv preprint arXiv:2408.04187*, 2024.
- [450] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328, 2023.
- [451] Qitian Wu, Chenxiao Yang, and Junchi Yan. Towards open-world feature extrapolation: An inductive graph learning approach. *Advances in Neural Information Processing Systems*, 34: 19435–19447, 2021.
- [452] Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vasilis N Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. *arXiv preprint arXiv:2404.13207*, 2024.
- [453] Xiaobao Wu, Liangming Pan, William Yang Wang, and Anh Tuan Luu. Updating language models with unstructured facts: Towards practical knowledge editing. *arXiv preprint arXiv:2402.18909*, 2024.
- [454] Xixi Wu, Yifei Shen, Caihua Shan, Kaitao Song, Siwei Wang, Bohang Zhang, Jiarui Feng, Hong Cheng, Wei Chen, Yun Xiong, et al. Can graph learning improve task planning? *arXiv preprint arXiv:2405.19119*, 2024.
- [455] Yike Wu, Nan Hu, Sheng Bi, Guilin Qi, Jie Ren, Anhuan Xie, and Wei Song. Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering. *arXiv preprint arXiv:2309.11206*, 2023.

- [456] Yike Wu, Nan Hu, Guilin Qi, Sheng Bi, Jie Ren, Anhuan Xie, and Wei Song. Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering. *arXiv preprint arXiv:2309.11206*, 2023.
- [457] Yike Wu, Yi Huang, Nan Hu, Yuncheng Hua, Guilin Qi, Jiaoyan Chen, and Jeff Z Pan. Cotkr: Chain-of-thought enhanced knowledge rewriting for complex knowledge graph question answering. *arXiv preprint arXiv:2409.19753*, 2024.
- [458] Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. Towards open-world recommendation with knowledge augmentation from large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 12–22, 2024.
- [459] Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A Rossi, Haoliang Wang, and Julian McAuley. Knowledge-aware query expansion with large language models for textual and relational retrieval. *arXiv preprint arXiv:2410.13765*, 2024.
- [460] Chong Xiang, Tong Wu, Zexuan Zhong, David Wagner, Danqi Chen, and Prateek Mittal. Certifiably robust rag against retrieval corruption. *arXiv preprint arXiv:2405.15556*, 2024.
- [461] Shunxin Xiao, Shiping Wang, Yuanfei Dai, and Wenzhong Guo. Graph neural networks in node classification: survey and evaluation. *Machine Vision and Applications*, 33(1):4, 2022.
- [462] Qianqian Xie, Jimin Huang, Tulika Saha, and Sophia Ananiadou. Gretel: Graph contrastive topic enhanced language model for long document extractive summarization. *arXiv preprint arXiv:2208.09982*, 2022.
- [463] Zhouhang Xie, Sameer Singh, Julian McAuley, and Bodhisattwa Prasad Majumder. Factual and informative review generation for explainable recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13816–13824, 2023.
- [464] Amy Xin, Yunjia Qi, Zijun Yao, Fangwei Zhu, Kaisheng Zeng, Xu Bin, Lei Hou, and Juanzi Li. Llmael: Large language models are good context augmenters for entity linking. *arXiv preprint arXiv:2407.04020*, 2024.
- [465] Lu Xing and Lina Sela. Graph neural networks for state estimation in water distribution systems: Application of supervised and semisupervised learning. *Journal of Water Resources Planning and Management*, 148(5):04022018, 2022.
- [466] Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. Benchmarking retrieval-augmented generation for medicine. *arXiv preprint arXiv:2312.10997*, 2023.
- [467] Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. Benchmarking retrieval-augmented generation for medicine. *arXiv preprint arXiv:2402.13178*, 2024.
- [468] Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, et al. Answering complex open-domain questions with multi-hop dense retrieval. *arXiv preprint arXiv:2009.12756*, 2020.
- [469] Chenyang Xu, Lei Cai, and Jingyang Gao. An efficient scRNA-seq dropout imputation method using graph attention network. *BMC bioinformatics*, 22:1–18, 2021.
- [470] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*, 2018.
- [471] Mingzhou Xu, Liangyou Li, Derek Wong, Qun Liu, Lidia S Chao, et al. Document graph for neural machine translation. *arXiv preprint arXiv:2012.03477*, 2020.
- [472] Ran Xu, Wenqi Shi, Yue Yu, Yuchen Zhuang, Bowen Jin, May D Wang, Joyce C Ho, and Carl Yang. Ram-ehr: Retrieval augmentation meets clinical predictions on electronic health records. *arXiv preprint arXiv:2403.00815*, 2024.

- [473] Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. Knowledge conflicts for llms: A survey. *arXiv preprint arXiv:2403.08319*, 2024.
- [474] Shicheng Xu, Liang Pang, Huawei Shen, and Xueqi Cheng. Unveil the duality of retrieval-augmented generation: Theoretical analysis and practical solution. *arXiv preprint arXiv:2406.00944*, 2024.
- [475] Weiwen Xu, Yang Deng, Huihui Zhang, Deng Cai, and Wai Lam. Exploiting reasoning chains for multi-hop science question answering. *arXiv preprint arXiv:2109.02905*, 2021.
- [476] Weiwen Xu, Huihui Zhang, Deng Cai, and Wai Lam. Dynamic semantic graph construction and reasoning for explainable multi-hop science question answering. *arXiv preprint arXiv:2105.11776*, 2021.
- [477] Weiye Xu, Min Wang, Wengang Zhou, and Houqiang Li. P-rag: Progressive retrieval augmented generation for planning on embodied everyday task. In *ACM Multimedia 2024*.
- [478] Jiaqi Xue, Mengxin Zheng, Yebowen Hu, Fei Liu, Xun Chen, and Qian Lou. Badrag: Identifying vulnerabilities in retrieval augmented generation of large language models. *arXiv preprint arXiv:2406.00083*, 2024.
- [479] Wenyuan Xue, Baosheng Yu, Wen Wang, Dacheng Tao, and Qingyong Li. Tgrnet: A table graph reconstruction network for table structure recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1295–1304, 2021.
- [480] Jiahuan Yan, Jintai Chen, Yixuan Wu, Danny Z Chen, and Jian Wu. T2g-former: organizing tabular features into relation graphs promotes heterogeneous feature interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10720–10728, 2023.
- [481] Diji Yang, Jinmeng Rao, Kezhen Chen, Xiaoyuan Guo, Yawen Zhang, Jie Yang, and Yi Zhang. Im-rag: Multi-round retrieval-augmented generation through learning inner monologues. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 730–740, 2024.
- [482] Kang Yang, Kamal Al-Sabahi, Yanmin Xiang, and Zuping Zhang. An integrated graph model for document summarization. *Information*, 9(9):232, 2018.
- [483] Rui Yang, Haoran Liu, Qingcheng Zeng, Yu He Ke, Wanxin Li, Lechao Cheng, Qingyu Chen, James Caverlee, Yutaka Matsuo, and Irene Li. Kg-rank: Enhancing large language models for medical qa with knowledge graphs and ranking techniques. *arXiv preprint arXiv:2403.05881*, 2024.
- [484] Rui Yang, Haoran Liu, Qingcheng Zeng, Yu He Ke, Wanxin Li, Lechao Cheng, Qingyu Chen, James Caverlee, Yutaka Matsuo, and Irene Li. Kg-rank: Enhancing large language models for medical qa with knowledge graphs and ranking techniques. *arXiv preprint arXiv:2403.05881*, 2024.
- [485] Shenghao Yang, Weizhi Ma, Peijie Sun, Min Zhang, Qingyao Ai, Yiqun Liu, and Mingchen Cai. Common sense enhanced knowledge-based recommendation with large language model. *arXiv preprint arXiv:2403.18325*, 2024.
- [486] Zhaoming Yang, Zhe Liu, Jing Zhou, Chaofan Song, Qi Xiang, Qian He, Jingjing Hu, Michael H Faber, Enrico Zio, Zhenlin Li, et al. A graph neural network (gnn) method for assigning gas calorific values to natural gas pipeline networks. *Energy*, 278:127875, 2023.
- [487] Zukang Yang and Zixuan Zhu. Curiousllm: Elevating multi-document qa with reasoning-infused knowledge graph prompting. *arXiv preprint arXiv:2404.09077*, 2024.
- [488] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.
- [489] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211, 2024.

- [490] Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Weichuan Liu, Lei Hou, and Juanzi Li. Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation. *arXiv preprint arXiv:2406.19215*, 2024.
- [491] Michihiro Yasunaga, Rui Zhang, Kshitij Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. Graph-based neural multi-document summarization. *arXiv preprint arXiv:1706.06681*, 2017.
- [492] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 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*, pages 535–546, 2021.
- [493] Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323, 2022.
- [494] Michihiro Yasunaga, Jure Leskovec, and Percy Liang. Linkbert: Pretraining language models with document links. *arXiv preprint arXiv:2203.15827*, 2022.
- [495] Fanghua Ye, Mingming Yang, Jianhui Pang, Longyue Wang, Derek F. Wong, Emine Yilmaz, Shuming Shi, and Zhaopeng Tu. Benchmarking llms via uncertainty quantification, 2024. URL <https://arxiv.org/abs/2401.12794>.
- [496] Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*, 2023.
- [497] Jiaxuan You, Xiaobai Ma, Yi Ding, Mykel J Kochenderfer, and Jure Leskovec. Handling missing data with graph representation learning. *Advances in Neural Information Processing Systems*, 33:19075–19087, 2020.
- [498] Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuhang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. *arXiv preprint arXiv:2110.04330*, 2021.
- [499] Junchi Yu, Ran He, and Rex Ying. Thought propagation: An analogical approach to complex reasoning with large language models. *arXiv preprint arXiv:2310.03965*, 2023.
- [500] Xinli Yu, Zheng Chen, Yuan Ling, Shujing Dong, Zongyi Liu, and Yanbin Lu. Temporal data meets llm-explainable financial time series forecasting. *arXiv preprint arXiv:2306.11025*, 2023.
- [501] Xueli Yu, Weizhi Xu, Zeyu Cui, Shu Wu, and Liang Wang. Graph-based hierarchical relevance matching signals for ad-hoc retrieval. In *Proceedings of the Web Conference 2021*, pages 778–787, 2021.
- [502] Zhuohan Yu, Yifu Lu, Yunhe Wang, Fan Tang, Ka-Chun Wong, and Xiangtao Li. Zinb-based graph embedding autoencoder for single-cell rna-seq interpretations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 4671–4679, 2022.
- [503] Siyu Yuan, Jiangjie Chen, Changzhi Sun, Jiaqing Liang, Yanghua Xiao, and Deqing Yang. Analogykb: Unlocking analogical reasoning of language models with a million-scale knowledge base. *arXiv preprint arXiv:2305.05994*, 2023.
- [504] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.
- [505] Ahtsham Zafar, Venkatesh Balavadhani Parthasarathy, Chan Le Van, Saad Shahid, Arsalan Shahid, et al. Building trust in conversational ai: A comprehensive review and solution architecture for explainable, privacy-aware systems using llms and knowledge graph. *arXiv preprint arXiv:2308.13534*, 2023.
- [506] Lukáš Zahradník, Jan Neumann, and Gustav Šír. A deep learning blueprint for relational databases. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

- [507] Cyril Zakka, Rohan Shad, Akash Chaurasia, Alex R Dalal, Jennifer L Kim, Michael Moor, Robyn Fong, Curran Phillips, Kevin Alexander, Euan Ashley, et al. Almanac—retrieval-augmented language models for clinical medicine. *NEJM AI*, 1(2):A1oa2300068, 2024.
- [508] Xuan Zang, Xianbing Zhao, and Buzhou Tang. Hierarchical molecular graph self-supervised learning for property prediction. *Communications Chemistry*, 6(1):34, 2023.
- [509] Deborah A Zarin, Tony Tse, Rebecca J Williams, Robert M Califf, and Nicholas C Ide. The clinicaltrials.gov results database—update and key issues. *New England Journal of Medicine*, 364(9):852–860, 2011.
- [510] Huimin Zeng, Zhenrui Yue, Qian Jiang, and Dong Wang. Federated recommendation via hybrid retrieval augmented generation. *arXiv preprint arXiv:2403.04256*, 2024.
- [511] Jingying Zeng, Richard Huang, Waleed Malik, Langxuan Yin, Bojan Babic, Danny Shacham, Xiao Yan, Jaewon Yang, and Qi He. Large language models for social networks: Applications, challenges, and solutions. *arXiv preprint arXiv:2401.02575*, 2024.
- [512] Shenglai Zeng, Jiankun Zhang, Pengfei He, Jie Ren, Tianqi Zheng, Hanqing Lu, Han Xu, Hui Liu, Yue Xing, and Jiliang Tang. Mitigating the privacy issues in retrieval-augmented generation (rag) via pure synthetic data. *arXiv preprint arXiv:2406.14773*, 2024.
- [513] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv:2402.16893*, 2024.
- [514] Shenglai Zeng, Jiankun Zhang, Bingheng Li, Yuping Lin, Tianqi Zheng, Dante Everaert, Hanqing Lu, Hui Liu, Yue Xing, Monica Xiao Cheng, et al. Towards knowledge checking in retrieval-augmented generation: A representation perspective. *arXiv preprint arXiv:2411.14572*, 2024.
- [515] Boyu Zhang, Hongyang Yang, Tianyu Zhou, Muhammad Ali Babar, and Xiao-Yang Liu. Enhancing financial sentiment analysis via retrieval augmented large language models. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 349–356, 2023.
- [516] Chen Zhang, Qiuchi Li, and Dawei Song. Aspect-based sentiment classification with aspect-specific graph convolutional networks. *arXiv preprint arXiv:1909.03477*, 2019.
- [517] Han Zhang, Quan Gan, David Wipf, and Weinan Zhang. Gfs: Graph-based feature synthesis for prediction over relational database. *Proceedings of the VLDB Endowment. ISSN*, 2150: 8097, 2023.
- [518] Haopeng Zhang and Jiawei Zhang. Text graph transformer for document classification. In *Conference on empirical methods in natural language processing (EMNLP)*, 2020.
- [519] Haopeng Zhang, Xiao Liu, and Jiawei Zhang. Contrastive hierarchical discourse graph for scientific document summarization. In *The 4th workshop on Computational Approaches to Discourse at the 61st Annual Meeting of the Association for Computational Linguistics (ACL CODI'23), July 9-14, 2023, Toronto, Canada.*, 2023.
- [520] Haozhen Zhang, Tao Feng, and Jiaxuan You. Graph of records: Boosting retrieval augmented generation for long-context summarization with graphs. *arXiv preprint arXiv:2410.11001*, 2024.
- [521] Jing Zhang, Biao Liu, Jie Tang, Ting Chen, and Juanzi Li. Social influence locality for modeling retweeting behaviors. In *Twenty-third international joint conference on artificial intelligence*. Citeseer, 2013.
- [522] Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784, 2022.

- [523] Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. Exploring collaboration mechanisms for llm agents: A social psychology view. *arXiv preprint arXiv:2310.02124*, 2023.
- [524] Lihui Zhang and Ruifan Li. Ke-gcl: Knowledge enhanced graph contrastive learning for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 76–87, 2022.
- [525] Ming Zhang, Jiyu Lu, Jiahao Yang, Jun Zhou, Meilin Wan, and Xuejun Zhang. From coarse to fine: Enhancing multi-document summarization with multi-granularity relationship-based extractor. *Information Processing & Management*, 61(3):103696, 2024.
- [526] Ningyu Zhang, Xiang Chen, Xin Xie, Shumin Deng, Chuanqi Tan, Mosha Chen, Fei Huang, Luo Si, and Huajun Chen. Document-level relation extraction as semantic segmentation. *arXiv preprint arXiv:2106.03618*, 2021.
- [527] Shichang Zhang, Da Zheng, Jian Zhang, Qi Zhu, Soji Adeshina, Christos Faloutsos, George Karypis, Yizhou Sun, et al. Hierarchical compression of text-rich graphs via large language models. *arXiv preprint arXiv:2406.11884*, 2024.
- [528] Wei Zhang, Zeyuan Chen, Hongyuan Zha, and Jianyong Wang. Learning from substitutable and complementary relations for graph-based sequential product recommendation. *ACM Transactions on Information Systems (TOIS)*, 40(2):1–28, 2021.
- [529] Wen Zhang, Yanbin Lu, Bella Dubrov, Zhi Xu, Shang Shang, and Emilio Maldonado. Deep hierarchical product classification based on pre-trained multilingual knowledge. 2021.
- [530] Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. Greaselm: Graph reasoning enhanced language models. In *International Conference on Learning Representations*.
- [531] Xingyao Zhang, Linjun Shou, Jian Pei, Ming Gong, Lijie Wen, and Daxin Jiang. A graph representation of semi-structured data for web question answering. *arXiv preprint arXiv:2010.06801*, 2020.
- [532] Xuanyu Zhang. Cfgnn: Cross flow graph neural networks for question answering on complex tables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9596–9603, 2020.
- [533] Xulang Zhang, Rui Mao, and Erik Cambria. A survey on syntactic processing techniques. *Artificial Intelligence Review*, 56(6):5645–5728, 2023.
- [534] Yi Zhang, Yuying Zhao, Zhaoqing Li, Xueqi Cheng, Yu Wang, Olivera Kotevska, S Yu Philip, and Tyler Derr. A survey on privacy in graph neural networks: Attacks, preservation, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2024.
- [535] Yichi Zhang, Zhuo Chen, Wen Zhang, and Huajun Chen. Making large language models perform better in knowledge graph completion. *arXiv preprint arXiv:2310.06671*, 2023.
- [536] Yu Zhang, Hao Cheng, Zhihong Shen, Xiaodong Liu, Ye-Yi Wang, and Jianfeng Gao. Pre-training multi-task contrastive learning models for scientific literature understanding. *arXiv preprint arXiv:2305.14232*, 2023.
- [537] Yue Zhang, Zhihao Zhang, Wenbin Lai, Chong Zhang, Tao Gui, Qi Zhang, and Xuan-Jing Huang. to-tree: Parsing pdf text blocks into a tree. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10704–10714, 2024.
- [538] Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. Every document owns its structure: Inductive text classification via graph neural networks. *arXiv preprint arXiv:2004.13826*, 2020.
- [539] Yuhao Zhang, Peng Qi, and Christopher D Manning. Graph convolution over pruned dependency trees improves relation extraction. *arXiv preprint arXiv:1809.10185*, 2018.

- [540] Yunyi Zhang, Ruozhen Yang, Xueqiang Xu, Jinfeng Xiao, Jiaming Shen, and Jiawei Han. Tele-class: Taxonomy enrichment and llm-enhanced hierarchical text classification with minimal supervision. *arXiv preprint arXiv:2403.00165*, 2024.
- [541] Yuyu Zhang, Ping Nie, Arun Ramamurthy, and Le Song. Answering any-hop open-domain questions with iterative document reranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 481–490, 2021.
- [542] Zhehao Zhang, Jiaao Chen, and Difyi Yang. Darg: Dynamic evaluation of large language models via adaptive reasoning graph. *arXiv preprint arXiv:2406.17271*, 2024.
- [543] Zhenyu Zhang, Bowen Yu, Xiaobo Shu, Xue Mengge, Tingwen Liu, and Li Guo. From what to why: Improving relation extraction with rationale graph. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 86–95, 2021.
- [544] Zhiqiang Zhang, Linan Wang, Xiaoqin Xie, and Haiwei Pan. A graph based document retrieval method. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, pages 426–432. IEEE, 2018.
- [545] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270, 2020.
- [546] Zixuan Zhang and Heng Ji. Abstract meaning representation guided graph encoding and decoding for joint information extraction. In *Proc. The 2021 Conference of the North American Chapter of the Association for Computational Linguistics-Human Language Technologies (NAACL-HLT2021)*, 2021.
- [547] Zixuan Zhang, Heba Elfardy, Markus Dreyer, Kevin Small, Heng Ji, and Mohit Bansal. Enhancing multi-document summarization with cross-document graph-based information extraction. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1696–1707, 2023.
- [548] Chenlong Zhao, Xiwen Zhou, Xiaopeng Xie, and Yong Zhang. Hierarchical attention graph for scientific document summarization in global and local level. *arXiv preprint arXiv:2405.10202*, 2024.
- [549] Haiteng Zhao, Shengchao Liu, Ma Chang, Hannan Xu, Jie Fu, Zhihong Deng, Lingpeng Kong, and Qi Liu. Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning. *Advances in Neural Information Processing Systems*, 36:5850–5887, 2023.
- [550] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38, 2024.
- [551] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024.
- [552] Qifang Zhao, Weidong Ren, Tianyu Li, Xiaoxiao Xu, and Hong Liu. Graphgpt: Graph learning with generative pre-trained transformers. *arXiv preprint arXiv:2401.00529*, 2023.
- [553] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *proceedings of the 30th acm international conference on information & knowledge management*, pages 4653–4664, 2021.
- [554] Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, et al. Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering*, 2024.
- [555] Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36, 2024.

- [556] Chen Zheng and Parisa Kordjamshidi. Srlgrn: Semantic role labeling graph reasoning network. *arXiv preprint arXiv:2010.03604*, 2020.
- [557] Jiajun Zhong, Ning Gui, and Weiwei Ye. Data imputation with iterative graph reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11399–11407, 2023.
- [558] Huiwei Zhou, Yibin Xu, Weihong Yao, Zhe Liu, Chengkun Lang, and Haibin Jiang. Global context-enhanced graph convolutional networks for document-level relation extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5259–5270, 2020.
- [559] Kaixiong Zhou, Zirui Liu, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu. Table2graph: Transforming tabular data to unified weighted graph. In *IJCAI*, pages 2420–2426, 2022.
- [560] Qihang Zhou, Jiming Chen, Haoyu Liu, Shibo He, and Wenchao Meng. Detecting multivariate time series anomalies with zero known label. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4963–4971, 2023.
- [561] Yujia Zhou, Yan Liu, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Zheng Liu, Chaozhuo Li, Zhicheng Dou, Tsung-Yi Ho, and Philip S Yu. Trustworthiness in retrieval-augmented generation systems: A survey. *arXiv preprint arXiv:2409.10102*, 2024.
- [562] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*, 2021.
- [563] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020.
- [564] Kaijie Zhu, Jindong Wang, Qinlin Zhao, Ruochen Xu, and Xing Xie. Dyval 2: Dynamic evaluation of large language models by meta probing agents. *arXiv preprint arXiv:2402.14865*, 2024.
- [565] Lixing Zhu, Yulan He, and Deyu Zhou. Neural opinion dynamics model for the prediction of user-level stance dynamics. *Information Processing & Management*, 57(2):102031, 2020.
- [566] Yinghao Zhu, Changyu Ren, Zixiang Wang, Xiaochen Zheng, Shiyun Xie, Junlan Feng, Xi Zhu, Zhoujun Li, Liantao Ma, and Chengwei Pan. Emerge: Integrating rag for improved multimodal ehr predictive modeling. *arXiv preprint arXiv:2406.00036*, 2024.
- [567] Yinghao Zhu, Changyu Ren, Shiyun Xie, Shukai Liu, Hangyuan Ji, Zixiang Wang, Tao Sun, Long He, Zhoujun Li, Xi Zhu, et al. Realm: Rag-driven enhancement of multimodal electronic health records analysis via large language models. *arXiv preprint arXiv:2402.07016*, 2024.
- [568] Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Ningyu Zhang, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, and Huajun Chen. Knowagent: Knowledge-augmented planning for llm-based agents. *arXiv preprint arXiv:2403.03101*, 2024.
- [569] Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A Rossi, Somdeb Sarkhel, and Chao Zhang. Toolchain\*: Efficient action space navigation in large language models with a\* search. In *The Twelfth International Conference on Learning Representations*.
- [570] Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A Rossi, Somdeb Sarkhel, and Chao Zhang. Toolchain\*: Efficient action space navigation in large language models with a\* search. *arXiv preprint arXiv:2310.13227*, 2023.
- [571] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. Mat-power: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems*, 26(1):12–19, 2010.
- [572] Tao Zou, Le Yu, Yifei Huang, Leilei Sun, and Bowen Du. Pretraining language models with text-attributed heterogeneous graphs. *arXiv preprint arXiv:2310.12580*, 2023.

- [573] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. Poisonedrag: Knowledge corruption attacks to retrieval-augmented generation of large language models, 2024. URL <https://arxiv.org/abs/2402.07867>.