ArchRAG: Attributed Community-based Hierarchical Retrieval-Augmented Generation

Shu Wang[1], **Yixiang Fang**[1*], **Yingli Zhou**[1], **Xilin Liu**[2], **Yuchi Ma**[2]
[1]The Chinese University of Hong Kong, Shenzhen
[2]Huawei Cloud Computing Technologies CO., LTD.
{shuwang3, yinglizhou}@link.cuhk.edu.cn, fangyixiang@cuhk.edu.cn, {liuxilin3, mayuchi1}@huawei.com

Abstract

Retrieval-Augmented Generation (RAG) has proven effective in integrating external knowledge into large language models (LLMs) for solving question-answer (QA) tasks. The state-of-the-art RAG approaches often use the graph data as the external data since they capture the rich semantic information and link relationships between entities. However, existing graph-based RAG approaches cannot accurately identify the relevant information from the graph and also consume large numbers of tokens in the online retrieval process. To address these issues, we introduce a novel graph-based RAG approach, called Attributed Community-based Hierarchical RAG (ArchRAG), by augmenting the question using attributed communities, and also introducing a novel LLM-based hierarchical clustering method. To retrieve the most relevant information from the graph for the question, we build a novel hierarchical index structure for the attributed communities and develop an effective online retrieval method. Experimental results demonstrate that ArchRAG outperforms existing methods in both accuracy and token cost.

Code — https://github.com/sam234990/ArchRAG
Extended version — https://arxiv.org/abs/2502.09891

Introduction

Retrieval-Augmented Generation (RAG) has emerged as a core approach for enhancing large language models (LLMs) by enabling access to domain-specific and real-time updated knowledge beyond their pre-training corpus (2024; 2023; 2024; 2024; 2024b; 2024; 2024). By improving the trustworthiness and interpretability of LLMs, RAG has been widely adopted across a broad range of applications (2024a; 2024; 2024; 2023; 2024; 2024a). Current state-of-the-art RAG approaches often use graph-structured data as external knowledge, due to its ability to capture the rich semantics and relationships. Given a question Q, the key idea of graph-based RAG is to retrieve relevant information (e.g., nodes, subgraphs, or textual information) from the graph, incorporate them with Q as the prompt, and feed them into the LLM, as illustrated in Figure 1. Several recent methods (2024; 2024; 2024; 2024a; 2024b; 2024; 2024) address

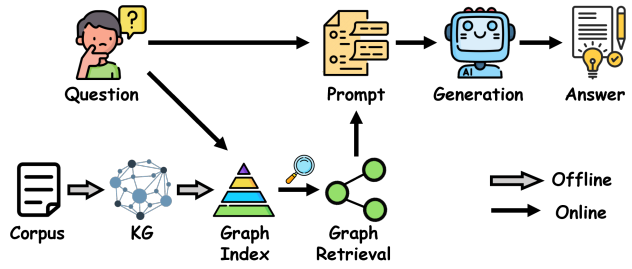


Figure 1: The general workflow of graph-based RAG, which retrieves relevant information (e.g., nodes, subgraphs, or textual information) to facilitate the LLM generation.

two common question answering (QA) tasks: abstract questions, which require reasoning over high-level themes (e.g., "What are the potential impacts of LLMs on education?"), and specific questions, which focus on entity-centric factual details (e.g., "Who won the Turing Award in 2024?").

In the past year, a surge of graph-based RAG methods (2025; 2024b; 2024; 2024; 2024; 2023) has emerged, each proposing different retrieval strategies to extract detailed information for response generation. Among them, GraphRAG (2024), proposed by Microsoft, is the most prominent and the first to leverage community summarization for abstract QA. It builds a knowledge graph (KG) from the external corpus, detects communities using Leiden (2019), and generates a summary for each community using LLMs. For abstract questions that require high-level information, it adopts a Global Search approach, traversing all communities and using LLMs to retrieve the most relevant summaries. In contrast, for specific questions, it employs a Local Search method to retrieve entities, relevant text chunks, and low-level communities, providing the multi-hop detailed information for accurate answers.

Although some methods claim that GraphRAG underperforms and is difficult to apply in practice (2024; 2024a), our re-examination shows that it is primarily constrained by the following limitations (**L**): **L1.** *Low community quality*: GraphRAG uses the Leiden (2019) algorithm to detect communities, but this approach relies solely on graph structure and ignores the rich semantics of nodes and edges. As a result, the detected communities often consist of differ-

*Corresponding author.

ent themes, which leads to the poor quality of community summaries and further decreases its performance. **L2.** *Limited compatibility*: While GraphRAG employs Global and Local Search strategies, each retrieves graph elements at only one granularity, making it inadequate for simultaneously addressing both abstract and specific questions and limiting its applicability in real-world open-ended scenarios. **L3.** *High generation cost*: Although GraphRAG performs well on abstract questions, analyzing all communities with LLMs is both time- and token-consuming. For example, GraphRAG detects 2,984 communities in the MultihopRAG (2024) dataset, and answering just 100 questions incurs a cost of approximately $650 and 106 million tokens[1], which is an impractical overhead.

To tackle the above limitations of GraphRAG, in this paper, we propose a novel graph-based RAG approach, called **A**ttributed **C**ommunity-based **H**ierarchical **RAG** (ArchRAG). ArchRAG leverages attributed communities (ACs) and introduces an efficient hierarchical retrieval strategy to adaptively support both abstract and specific questions. To mitigate **L1**, we detect high-quality ACs by exploiting both links and the attributes of nodes, ensuring that each AC comprises nodes that are not only densely connected but also share similar semantic themes (2009). We further propose a novel LLM-based iterative framework for hierarchical AC detection, which can incorporate any existing community detection methods (2009; 2019; 2007; 2007; 2016). In each iteration, we detect ACs based on both attribute similarity and connectivity, summarize each AC using an LLM, and construct a higher-level graph by treating each AC as a node, connecting pairs with similar summaries. By iterating the above steps multiple times, we obtain a set of ACs that can be organized into a hierarchical tree structure.

To effectively address **L2**, we organize all ACs and entities into a hierarchical index and retrieve relevant elements from all levels to support both abstract and specific questions. Entities offer fine-grained details, while LLM-generated AC summaries capture relational structures and provide high-level condensed overviews (2024; 2018), making them suitable for both multi-hop reasoning and abstract insight extraction. To support efficient retrieval across levels, we propose C-HNSW (Community-based HNSW), a novel hierarchical index inspired by the HNSW algorithm (2018) for approximate nearest neighbor (ANN) search.

To mitigate the high generation cost caused by traversing all communities (**L3**), we propose a hierarchical search with adaptive filtering to efficiently select the most relevant ACs and entities while maintaining performance. Specifically, we design an efficient hierarchical retrieval algorithm over the proposed C-HNSW index, which supports top-k nearest neighbor search across multiple levels, thereby facilitating access to multi-level relevant information. Furthermore, the adaptive filtering mechanism identifies the most informative results at each level, making the retrieved information complementary.

We have extensively evaluated ArchRAG on real-world datasets, and the results show that it consistently outperforms existing methods in both abstract and specific QA tasks. ArchRAG achieves a 10% higher accuracy than state-of-the-art graph-based RAG methods on specific questions and shows notable gains on abstract QA. Moreover, ArchRAG is very token-efficient, saving up to 250 times the token usage compared to GraphRAG (2024).

In summary, our main contributions are as follows:

- We present a novel graph-based RAG approach by using ACs that are organized hierarchically and detected by an LLM-based hierarchical clustering method.
- To index ACs, we propose a novel hierarchical index structure called C-HNSW and also develop an efficient online retrieval method.
- Extensive experiments show that ArchRAG is both highly effective and efficient, and achieves state-of-the-art performance on both abstract and specific QA tasks.

Related Work

In this section, we review the related works, including Retrieval-Augmentation-Generation (RAG) approaches, and LLMs for graph mining and learning.

• **RAG approaches.** RAG has been proven to excel in many tasks, including open-ended question answering (2024; 2023), programming context (2024b; 2023), SQL rewrite (2025; 2024), and data cleaning (2024; 2022; 2024). The naive RAG technique relies on retrieving query-relevant contexts from external knowledge bases to mitigate the "hallucination" of LLMs. Recently, many RAG approaches (2025; 2024; 2024; 2024; 2024b; 2024; 2024; 2024) have adopted graph structures to organize the information and relationships within documents, leading to improved performance. For more details, please refer to the recent survey of graph-based RAG methods (2024).

• **LLM for graph mining.** Recent advances in LLMs have offered opportunities to leverage LLMs in graph mining. These include using LLMs for KG construction (2024), addressing complex graph mining tasks (2024a; 2024; 2024; 2024b), and employing KG to enhance the LLM reasoning (2023; 2024b; 2023; 2023; 2024; 2023; 2025). For instance, RoG (2023) proposes a planning-retrieval-reasoning framework that retrieves reasoning paths from KGs to guide LLMs conducting faithful reasoning. StructGPT (2023) and ToG (2023) treat LLMs as agents that interact with KGs to find reasoning paths leading to the correct answers.

Our Approach ArchRAG

We begin by presenting the overall workflow and design rationale of ArchRAG, followed by detailed descriptions of each component. As illustrated in Figure 2, our proposed ArchRAG consists of two phases. In the *offline indexing* phase, ArchRAG first constructs a KG from the corpus, then detects ACs by a novel LLM-based hierarchical clustering method, and finally builds the C-HNSW index. During the *online retrieval* phase, ArchRAG first converts the question into a query vector, then retrieves relevant information from

[1]The cost of GPT-4o is $10/M tokens for output and $2.50/M tokens for input (for details, please refer to OpenAI pricing).

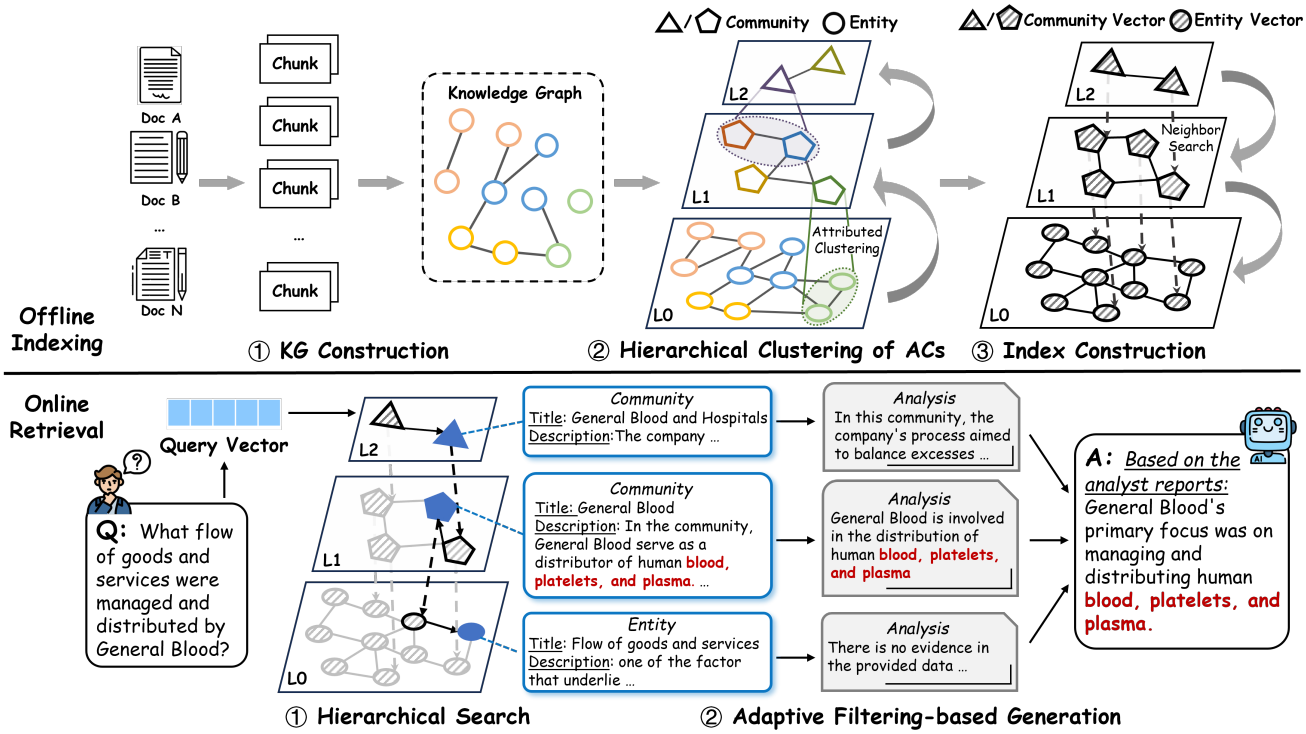


Figure 2: ArchRAG consists of two phases: offline indexing and online retrieval. For the online retrieval phase, we show an example of using ArchRAG to answer a question in the HotpotQA dataset.

the C-HNSW index, and finally generates answers through an adaptive filtering-based generation process

Our ArchRAG detects ACs by exploiting both links and attributes, and organizes ACs and entities into a novel hierarchical index, C-HNSW, yielding the following advantages: 1) Each AC group densely connected entities with shared themes and a high-quality summary. 2) The hierarchical structure captures multiple levels of abstraction: lower-level entities and communities encode detailed KG knowledge, while higher-level communities provide global context, enabling ArchRAG to address questions at varying granularity. and 3) The C-HNSW index efficiently retrieves relevant information across levels, supporting fast and accurate responses to both abstract and specific questions.

Offline Indexing

KG construction. ArchRAG builds a KG by prompting the LLM to extract entities and relations from each chunk of the text corpus D. Specifically, all text contexts are segmented into chunks based on specified chunk length, enabling the LLM to extract entities and relations from each chunk using in-context learning (2020), thus forming subgraphs. These subgraphs are then merged, with entities and relations that appear repeatedly across multiple subgraphs being consolidated by the LLM to generate a complete description. Finally, we get a KG, denoted by $G(V, E)$, where V and E are sets of vertices and edges, respectively, and each vertex and edge is associated with textual attributes.

LLM-based hierarchical clustering. We propose an iterative LLM-based hierarchical clustering framework that supports arbitrary graph augmentation (e.g., KNN connections and CODICIL (2013)) and graph clustering algorithms (e.g., weighted Leiden (2019), weighted spectral clustering, and SCAN (2007)). Specifically, we propose to augment the KG by linking entities if their attribute similarities are larger than a threshold, and then associate each pair of linked entities with a weight denoting their attribute similarity value. Next, we generate the ACs using any given graph clustering algorithm. In this way, both node attributes and structural links are jointly considered during community detection.

Algorithm 1 shows the above iterative clustering process. Given a graph augmentation method Aug, clustering algorithm $GCluster$ and stopping condition T, we perform the following steps in each iteration: (1) augmenting the graph (line 3); (2) computing the edge weights (lines 4-5); (3) clustering the augmented graph (line 6); (4) generating a summary for each community using LLM (line 7); and (5) building a new attributed graph where each node denotes an AC and two nodes are linked if their community members are connected (line 9). We repeat the iterations until the stopping condition T (such as insufficient nodes or reaching the specified level limit) is met. Since each iteration corresponds to one layer, all the ACs HC can be organized into a multi-layer hierarchical tree structure, denoted by Δ, where each community in one layer includes multiple communities in the next layer. The extended version also provides more details.

Algorithm 1: LLM-based hierarchical clustering

input : $G(V, E)$, `Aug`, `GCluster`, T
$T \leftarrow$ `False`, $HC \leftarrow \emptyset$;
repeat
  $G'(V, E') \leftarrow$ `Aug`$(G(V, E))$;
  for *each* $e' = (u, v) \in E'$ **do**
    update the weight of e' as $1 - \cos(z_u, z_v)$;
  $C \leftarrow$ `GCluster`$(G'(V, E'))$;
  for *each* $c \in C$ **do** generate summary of c by LLM ;
  $HC \leftarrow HC \cup C$;
  $G(V, E) \leftarrow$ build a new graph using C and E';
  // update T according to $G(V, E)$;
until T=`True`;
return HC;

C-HNSW index. Given a query, to efficiently identify the most relevant information from each layer of the hierarchical tree Δ, a naive method is to build a vector database for the ACs in each layer, which is costly in both time and space. To tackle this issue, we propose to build a single hierarchical index for all the communities. Recall that the ACs in Δ form a tree structure, and the number of nodes decreases as the layer level increases. Since this tree structure is similar to the HNSW (Hierarchical Navigable Small World) index which is the most well-known index for efficient ANN search (2018), we propose to map entities and ACs of Δ into high-dimensional nodes, and then build a unified Community-based HNSW (C-HNSW) index for them.

• **The structure of C-HNSW.** Conceptually, the C-HNSW index is a list of simple graphs with links between them, denoted by $\mathcal{H} = (\mathcal{G}, L_{inter})$ with $\mathcal{G} = \{G_0 = (V_0, E_0), G_1 = (V_1, E_1), \cdots, G_L = (V_L, E_L)\}$, where G_i is a simple graph and each node of the simple graph corresponds to an attributed community or entity. The number L of layers in $\mathcal{H}$ is the same as that of Δ. Specifically, for each attributed community or entity in the i-th layer of Δ, we map it to a high-dimensional node in the i-th layer of $\mathcal{H}$ by using a language model (e.g., nomic-embed-text (2024)).

We next establish two types of links between these high-dimensional nodes, i.e., *intra-layer* and *inter-layer* links:

- **Intra-layer links:** These links exist between nodes in the same layers. Specifically, for each node in each layer, we link it to at least M nearest neighbors within the same layer, where M is a predefined value, and the nearest neighbors are determined according to a given distance metric d. Thus, all the intra-layer links are edges in all the simple graphs: $L_{intra} = \bigcup_{i=0}^{L} E_i$.
- **Inter-layer links:** These links cross two adjacent layers. Specifically, we link each node in each layer to its nearest neighbor in the next layer. As a result, all the inter-layer links can be represented as $L_{inter} = \bigcup_{i=1}^{L} \{(v, \psi(v)) | v \in V_i, \psi(v) \in V_{i-1}\}$, where $\psi(\cdot) : V_i \rightarrow V_{i-1}$ is the injective function that identifies the nearest neighbor of each node in the lower layer.

For example, in Figure 2, the C-HNSW index has three layers (simple graphs), incorporating all the ACs. Within each layer, each node is connected to its two nearest neighbors via intra-layer links, denoted by undirected edges. The inter-layer links are represented by arrows, e.g., the green community at layer L_1 is connected to the green entity (its nearest neighbor at layer L_0).

Intuitively, since the two types of links above are established based on nearest neighbors, C-HNSW allows us to quickly search the relevant information for a query by traversing along with these links. Note that C-HNSW is different from HNSW since it has intra-layer links and each node exists in only one layer.

• **The construction of C-HNSW.** We propose a top-down approach to build a C-HNSW. Specifically, by leveraging the query process of C-HNSW, which will be introduced in online retrieval, nodes are progressively inserted into the index starting from the top layer, connecting intra-layer links within the same layer and updating the inter-layer links. For lack of space, we give the details of the construction algorithm in the extended version.

Online retrieval

In the online retrieval phase, after obtaining the query vector for a given question, ArchRAG generates the final answer by first conducting a hierarchical search on the C-HNSW index and then analyzing and filtering the retrieved information.

Hierarchical search. We propose an efficient and fast retrieval algorithm, hierarchical search, to retrieve nodes from each layer of the C-HNSW structure. Intuitively, retrieving nodes from a given layer in C-HNSW requires starting from the top layer and searching downward through two types of links (i.e., *intra-layer* and *inter-layer* links) to locate the nearest neighbors at the given layer. In contrast, our hierarchical search algorithm accelerates this process by reusing intermediate results, the nearest neighbors found in higher layers, as the starting node for lower layers. This approach avoids redundant computations that would otherwise arise from repeatedly searching from the top layer, thereby enabling efficient multi-layer retrieval.

Algorithm 2 illustrates hierarchical search. Given the C-HNSW $\mathcal{H}$, query point q, and the number k of nearest neighbors to retrieve at each layer, the hierarchical search algorithm can be implemented by the following iterative process:

1. Start from a random node at the highest layer L, which serves as the starting node for layer L (line 1).
2. For each layer i from the top layer L down to layer 0, the algorithm begins at the starting node and performs a greedy traversal (i.e., the `SearchLayer` procedure) to find the set R_i of the k nearest neighbors of q. The set R_i is then merged into the final result set R (lines 4–5).
3. The closest neighbor c of q is then obtained from R_i, and the algorithm proceeds to the next layer by traversing the inter-layer link of c, using it as the starting node for the subsequent search (lines 6–7).

Specifically, the greedy traversal strategy compares the distance between the query point and the visited nodes during the search process. It maintains a candidate expansion

Algorithm 2: Hierarchical search

input : $\mathcal{H} = (\mathcal{G}, L_{inter})$, q, k.
$s \leftarrow$ a random node in the highest layer L;
$R \leftarrow \emptyset$;
for $i \leftarrow L, \cdots, 0$ **do**
  $R_i \leftarrow$ `SearchLayer` $(G_l = (V_l, E_l), q, s, k)$;
  $R \leftarrow R \cup R_i$;
  $c \leftarrow$ get the nearest node from R_i;
  $s \leftarrow$ find the node in layer $i-1$ via the inter-layer links of c;
return R;
Procedure `SearchLayer` $(G_i = (V_i, E_i), q, s, k)$ **:**
  $V \leftarrow \{s\}$, $K \leftarrow \{s\}$, $Q \leftarrow$ initialize a queue containing s;
  while $|K| > 0$ **do**
    $c \leftarrow$ nearest node in Q;
    $f \leftarrow$ furthest node in K;
    if $d(c, q) > d(f, q)$ **then break** ;
    for *each neighbor* $x \in N(c)$ *in* G_i **do**
      if $x \in V$ **then continue**;
      $V \leftarrow V \cup \{x\}$;
      $f \leftarrow$ furthest node in K;
      if $d(x, q) < d(f, q)$ *or* $|K| < k$ **then**
        $Q \leftarrow Q \cup \{x\}$, $K \leftarrow K \cup \{x\}$;
        if $|K| > k$ **then** remove f from K;
  return K;

queue Q and a dynamic nearest neighbor set K containing k elements, along with a stopping condition:

- Expansion Queue Q: For each neighbor x of a visited node, if $d(x, q) < d(f, q)$, where f is the furthest node from R to q, then x is added to the expansion queue.
- Dynamic Nearest Neighbor Set K: Nodes added to C are used to update K, ensuring that it maintains no more than k elements, where k is the number of query results.
- Stopping Condition: The traversal terminates if a node x expanded from Q satisfies $d(n, q) > d(n, f)$, where f is the furthest node in K from the query point q.

After completing the hierarchical search and obtaining the ACs and entities from each layer, we further extract their associated textual information. In particular, at the bottom layer, we also extract the relationships between the retrieved entities, resulting in the textual subgraph representation denoted as R_0. We denote all the retrieved textual information from each layer as R_i, where $i \in 0, 1, \ldots, L$, which will be used in the adaptive filtering-based generation process.

Adaptive filtering-based generation. While some optimized LLMs support longer text inputs, they may still encounter issues such as the "lost in the middle" dilemma (Liu et al. 2024b). Thus, direct utilization of retrieved information comprising multiple text segments for LLM-based answer generation risks compromising output accuracy.

To mitigate this limitation, we propose an adaptive filtering-based method that harnesses the LLM's inherent reasoning capabilities. We first prompt the LLM to extract and generate an analysis report from the retrieved information, identifying the parts that are most relevant to answering the query and assigning relevance scores to these reports.

Dataset	Multihop-RAG	HotpotQA	NarrativeQA
Passages	609	9,221	1,572
Tokens	1,426,396	1,284,956	121,152,448
Nodes	23,353	37,436	650,571
Edges	30,716	30,758	679,426
Questions	2,556	1,000	43,304
Metrics	Acc, Rec	Acc, Rec	Blue, Met, Rou

Table 1: Datasets used in our experiments. Acc, Rec, Blue, Met, and Rou denote Accuracy, Recall, BLEU-1, METEOR, and ROUGE-L F1.

Then, all analysis reports are integrated and sorted, ensuring that the most relevant content is used to summarize the final response to the query, with any content exceeding the text limit being truncated. This process can be represented as:

$$A_i = LLM(P_{filter} || R_i) \quad (1)$$

$$Output = LLM(P_{merge} || Sort(\{A_0, A_1, \cdots, A_n\})) \quad (2)$$

where P_{filter} and P_{merge} represent the prompts for extracting relevant information and summarizing, respectively, A_i, $i \in 0 \cdots n$ denotes the filtered analysis report. The sort function orders the content based on the relevance scores from the analysis report.

Experiments

Setup

Datasets. We evaluate ArchRAG on both specific and abstract QA tasks. For specific QA, we use Multihop-RAG (2024), HotpotQA (2018), and NarrativeQA (2018), all of which are extensively utilized within the QA and Graph-based RAG research communities (2022; 2024; 2022; 2024; 2024; 2023). For abstract QA, we follow the GraphRAG (2024) method and reuse the Multihop-RAG corpus, prompting LLM to generate questions that convey a high-level understanding of dataset contents. The statistics of these datasets are reported in Table 1.

Baselines. Our experiments consider three configurations:

- **Inference-only:** Using an LLM to answer questions without retrieval, i.e., Zero-Shot and CoT (2022).
- **Retrieval-only:** Retrieval models extract relevant chunks from all documents and use them as prompts for LLMs. We select strong and widely used retrieval models: BM25 (1994) and Vanilla RAG.
- **Graph-based RAG:** These methods leverage graph data during retrieval. We select RAPTOR (2024), HippoRAG (2024), GraphRAG (2024), and LightRAG (2024). Particularly, GraphRAG has two versions, i.e., GGraphRAG and LGraphRAG, which use global and local search methods, respectively. Similarly, LightRAG integrates local search, global search, and hybrid search, denoted by LLightRAG, HLightRAG, and HyLightRAG, respectively.

In GGraphRAG, all communities below the selected level are first retrieved, and then the LLM is used to filter out irrelevant communities. This process can be viewed as utilizing the LLM as a *retriever* to find relevant communities

	VR	LR	C1	C2	AR
VR	50	46	18	18	1
LR	54	50	21	29	16
C1	82	79	50	86	18
C2	82	71	14	50	16
AR	99	84	82	84	50

(a) Comprehensiveness

	VR	LR	C1	C2	AR
VR	50	64	3	12	8
LR	36	50	52	63	33
C1	97	48	50	52	46
C2	88	37	48	50	42
AR	92	67	54	58	50

(b) Diversity

	VR	LR	C1	C2	AR
VR	50	39	15	21	8
LR	61	50	59	30	35
C1	85	41	50	60	42
C2	79	70	40	50	38
AR	92	65	58	62	50

(c) Empowerment

	VR	LR	C1	C2	AR
VR	50	46	14	18	4
LR	54	50	48	31	22
C1	86	52	50	70	31
C2	82	69	30	50	30
AR	96	78	69	70	50

(d) Overall

Figure 3: Head-to-head win rates for abstract QA, comparing each row method against each column (higher is better). VR, LR, and AR denote Vanilla RAG, HyLightRAG, and ArchRAG, respectively.

within the corpus. According to the selected level of communities (2024), GGraphRAG can be further categorized into C1 and C2, representing high-level and intermediate-level communities, respectively, with C2 as the default.

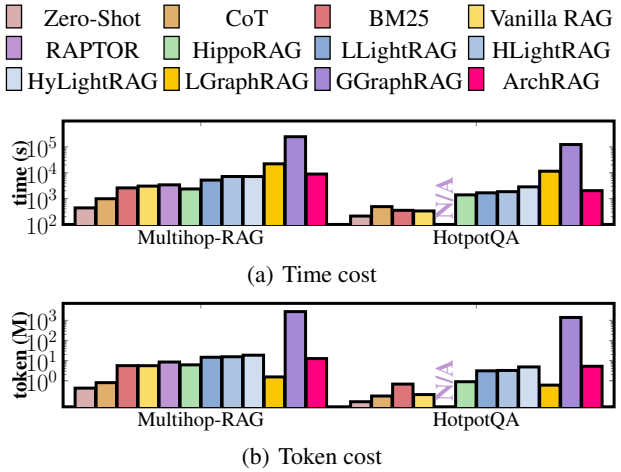
Metrics & Implementation. For the specific QA tasks, we use Accuracy and Recall to evaluate performance on the first two datasets based on whether gold answers are included in the generations instead of strictly requiring exact matching, following (2024; 2022; 2023). We also use the official metrics of BLEU, METEOR, and ROUGE-L F1 in the NarrativeQA dataset. For the abstract QA task, we follow prior work (2024) and adopt a head-to-head comparison approach using an LLM evaluator (GPT-4o). Overall, we utilize four evaluation dimensions: Comprehensiveness, Diversity, Empowerment, and Overall. For implementation, we mainly use Llama 3.1-8B (2024) as the default LLM and use nomic-embed-text (2024) as the text embedding model. We use KNN for graph augmentation and the weighted Leiden algorithm for community detection. For retrieval item k, we search the same number of items at each layer, with $k = 5$ as the default. All methods are required to complete index construction and query execution within 3 days, respectively.

Overall results

We compare our method with baseline methods in solving both abstract and specific QA tasks.

• **Results of abstract QA tasks.** We compare ArchRAG against baselines across four dimensions on the Multihop-RAG dataset. For the LightRAG, we only compare the HyLightRAG method, as it represents the best version (2024). As shown in Figure 3, GGraphRAG outperforms other baseline methods, while our method achieves comparable performance on the diversity and empowerment dimensions and significantly surpasses it on the comprehensive dimension. Overall, by leveraging ACs, ArchRAG demonstrates superior performance in addressing abstract QA tasks.

• **Results of specific QA tasks.** Table 2 reports the performance of each method on three datasets. Note that GGraphRAG fails to complete querying on the NarrativeQA dataset within the 3-day time limit. RAPTOR is unable to build the index on datasets like HotpotQA, which contains a large number of text chunks. Its Gaussian Mixture Model (GMM) clustering algorithm requires prohibitive computational time and suffers from non-termination issues during clustering. Clearly, ArchRAG demonstrates a substantial performance advantage over other baseline methods on these datasets. The experimental results suggest that not all communities are suitable for specific QA tasks, as the GGraphRAG performs poorly. Furthermore, GraphRAG does not consider node attributes during clustering, which causes the community's summary to become dispersed, making it difficult for the LLM to extract relevant information from a large number of communities. Thus, we gain an interesting insight: *LLM may not be a good retriever, but is a good analyzer.* We further analyze the reasons behind the underperformance of each graph-based RAG method and support our claims with empirical evidence in the extended version.



(a) Time cost

(b) Token cost

Figure 4: Comparison of query efficiency.

• **Efficiency of ArchRAG.** We compare the time cost and token usage of ArchRAG with those of other baseline methods. As shown in Figure 4, ArchRAG demonstrates significant time and cost efficiency for online queries. For example, token usage on the HotpotQA dataset is cut by $\mathbf{250}\times$ with ArchRAG compared to GraphRAG-Global, from 1,394M tokens down to 5.1M tokens.

To further evaluate ArchRAG, we test the **efficiency of hierarchical search**, **indexing performance**, and **effectiveness on an additional dataset**, RAG-QA Arena (2024). Results show that ArchRAG achieves up to $5.4\times$ faster retrieval than basic HNSW, maintains efficient indexing, and achieves state-of-the-art performance on the RAG-QA Arena dataset. Additional details are provided in the extended version.

Baseline Type	Method	Multihop-RAG		HotpotQA		NarrativeQA		
		(Accuracy)	(Recall)	(Accuracy)	(Recall)	(BLEU-1)	(METEOR)	(ROUGE-L F1)
Inference-only	Zero-shot	47.7	23.6	28.0	31.8	<u>8.0</u>	7.9	8.6
	CoT	54.5	28.7	32.5	39.7	5.0	8.1	6.4
Retrieval-only	BM25	37.6	19.4	49.7	53.6	2.0	4.9	2.8
	Vanilla RAG	58.6	31.4	50.6	56.1	2.0	4.9	2.8
Graph-based RAG	RAPTOR	<u>59.1</u>	<u>34.1</u>	N/A	N/A	5.5	<u>12.5</u>	<u>9.1</u>
	HippoRAG	38.9	19.1	<u>51.3</u>	<u>56.8</u>	2.2	5.0	2.8
	LLightRAG	44.1	25.1	34.1	41.8	4.5	8.7	6.6
	HLightRAG	48.5	28.7	25.6	33.3	4.4	8.1	6.1
	HyLightRAG	50.3	30.3	35.6	43.3	5.0	9.4	7.0
	LGraphRAG	40.1	23.8	29.7	35.5	3.9	3.3	3.5
	GGraphRAG	45.9	28.4	33.5	42.6	OOT	OOT	OOT
Our proposed	ArchRAG	**68.8**	**37.2**	**65.4**	**69.2**	**11.5**	**15.6**	**17.6**

Table 2: Performance comparison of different methods across various datasets for solving specific QA tasks. The best and second-best results are marked in bold and underlined. OOT: Not finished within 3 days.

Method variants	Multihop-RAG		HotpotQA	
	(Acc)	(Rec)	(Acc)	(Rec)
ArchRAG	68.8	37.2	65.4	69.2
- Spec.	67.1	36.7	60.5	63.7
- Spec. (No Aug)	62.8	34.2	64.8	63.2
- Leiden	63.2	34.1	61.7	64.8
- Single-Layer	63.8	36.4	60.1	63.6
- Entity-Only	61.2	34.7	59.9	63.1
- Direct Prompt	59.9	29.6	40.7	45.4

Table 3: Comparing the performance of different variants of ArchRAG on the specific QA tasks. Acc and Rec denote Accuracy and Recall, respectively.

Method	Multihop-RAG		HotpotQA	
	(Acc)	(Rec)	(Acc)	(Rec)
GGraphRAG	45.9	28.4	33.5	42.6
GraphRAG+AC	49.3	31.4	50.6 (51.0% ↑)	52.8
ArchRAG	68.8	37.2	65.4	69.2

Table 4: Results of GraphRAG variants using our ACs.

Detailed Analysis

To better understand the effectiveness of our proposed ArchRAG, we perform the following ablation study and experiment with a GGraphRAG variant.

• **Ablation study.** To evaluate the contributions of different components, we design several ArchRAG variants and conduct ablation experiments. These include two modifications to the LLM-based hierarchical clustering framework, three targeting core design elements in ArchRAG—attributes, hierarchy, and communities—and one direct prompting variant, as detailed below:

- Spec.: Spectral clustering instead of weighted Leiden.
- Spec. (No Aug): Spectral clustering without graph augmentation.
- Leiden: Replaces our clustering framework with Leiden.
- Single-Layer: Replaces our hierarchical index and search with a single-layer community.
- Entity-Only: Generate the response using entities only.
- Direct Prompt: Direct prompts the LLM to generate the response without the adaptive filtering-based generation.

As shown in Table 3, the performance of ArchRAG on specific QA tasks decreases when each feature is removed, with the removal of the community component resulting in the most significant drop. Additionally, the direct variant demonstrates that the adaptive filtering-based generation process can effectively extract relevant information from retrieved elements.

• **Impact of Attributed Communities in RAG.** We propose a new variant, *GraphRAG+AC*, which replaces the original Leiden-based communities in GraphRAG with our ACs, while preserving the original Global Search pipeline. As shown in Table 4, this variant results in a significant performance improvement compared to the original approach. Specifically, on the HotpotQA dataset, GraphRAG+AC improves accuracy by 51% compared to GGraphRAG.

We further test ArchRAG under **different LLM backbones**, various **top-k retrieval** settings, evaluate the **community quality** of its LLM-based hierarchical clustering, and also provide additional **case studies** of our ArchRAG. Detailed experiments are provided in the extended version.

Conclusion

In this paper, we propose ArchRAG, a novel graph-based RAG approach, by augmenting the question using attributed communities from the knowledge graph built on the external corpus, and building a novel index for efficient retrieval of relevant information. Our experiments show that ArchRAG is highly effective and efficient. In the future, we will explore fast parallel graph-based RAG methods to process large-scale external corpus.

Acknowledgments

This work was supported in part by the 1+1+1 CUHK-CUHK(SZ)-GDSTC Joint Collaboration Fund under Grant 2025A0505000045, Guangdong Provincial Key Laboratory of Mathematical Foundations for Artificial Intelligence (2023B1212010 001), Shenzhen Research Institute of Big Data under Grant SIF20240002, and Huawei Collaboration Fund under Grant TC20240920019.

References

Angelidis, S.; and Lapata, M. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. *arXiv preprint arXiv:1808.08858*.

Asai, A.; Wu, Z.; Wang, Y.; Sil, A.; and Hajishirzi, H. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Brown, T. B. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Cao, Y.; Han, S.; Gao, Z.; Ding, Z.; Xie, X.; and Zhou, S. K. 2024. Graphinsight: Unlocking insights in large language models for graph structure understanding. *arXiv preprint arXiv:2409.03258*.

Chen, N.; Li, Y.; Tang, J.; and Li, J. 2024a. Graphwiz: An instruction-following language model for graph computational problems. In *KDD*.

Chen, S.; He, Y.; Cui, W.; Fan, J.; Ge, S.; Zhang, H.; Zhang, D.; and Chaudhuri, S. 2024b. Auto-Formula: Recommend Formulas in Spreadsheets using Contrastive Learning for Table Representations. *Proc. ACM Manag. Data*, 2(3): 1–27.

Chen, S.; Tang, N.; Fan, J.; Yan, X.; Chai, C.; Li, G.; and Du, X. 2023. Haipipe: Combining human-generated and machine-generated pipelines for data preparation. *Proc. ACM Manag. Data*, 1(1): 1–26.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Edge, D.; Trinh, H.; Cheng, N.; Bradley, J.; Chao, A.; Mody, A.; Truitt, S.; and Larson, J. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Fan, W.; Ding, Y.; Ning, L.; Wang, S.; Li, H.; Yin, D.; Chua, T.-S.; and Li, Q. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of KDD '24*, 6491–6501.

Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; and Wang, H. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Ghimire, A.; Prather, J.; and Edwards, J. 2024. Generative AI in Education: A Study of Educators' Awareness, Sentiments, and Influencing Factors. *arXiv preprint arXiv:2403.15586*.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proc. KDD '16*, 855–864.

Guo, Z.; Xia, L.; Yu, Y.; Ao, T.; and Huang, C. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. *arXiv e-prints*, arXiv–2410.

Gutiérrez, B. J.; Shu, Y.; Gu, Y.; Yasunaga, M.; and Su, Y. 2024. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. *arXiv preprint arXiv:2405.14831*.

Han, R.; Zhang, Y.; Qi, P.; Xu, Y.; Wang, J.; Liu, L.; Wang, W. Y.; Min, B.; and Castelli, V. 2024. RAG-QA Arena: Evaluating Domain Robustness for Long-form Retrieval Augmented Question Answering. In *Proc. EMNLP 2024*, 4354–4374.

He, X.; Tian, Y.; Sun, Y.; Chawla, N. V.; Laurent, T.; LeCun, Y.; Bresson, X.; and Hooi, B. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Hu, Y.; and Lu, Y. 2024. Rag and rau: A survey on retrieval-augmented language model in natural language processing. *arXiv preprint arXiv:2404.19543*.

Hu, Z.; Xu, Y.; Yu, W.; Wang, S.; Yang, Z.; Zhu, C.; Chang, K.-W.; and Sun, Y. 2022. Empowering language models with knowledge graph reasoning for question answering. *arXiv preprint arXiv:2211.08380*.

Huang, Y.; and Huang, J. 2024. A Survey on Retrieval-Augmented Text Generation for Large Language Models. *arXiv preprint arXiv:2404.10981*.

Huang, Y.; Zhang, S.; and Xiao, X. 2025. KET-RAG: A Cost-Efficient Multi-Granular Indexing Framework for Graph-RAG. *arXiv preprint arXiv:2502.09304*.

Jeong, S.; Baek, J.; Cho, S.; Hwang, S. J.; and Park, J. C. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

Jiang, J.; Zhou, K.; Dong, Z.; Ye, K.; Zhao, W. X.; and Wen, J.-R. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Kočiskỳ, T.; Schwarz, J.; Blunsom, P.; Dyer, C.; Hermann, K. M.; Melis, G.; and Grefenstette, E. 2018. The narrativeqa reading comprehension challenge. *TACL*, 6: 317–328.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Li, D.; Yang, S.; Tan, Z.; Baik, J. Y.; Yun, S.; Lee, J.; Chacko, A.; Hou, B.; Duong-Tran, D.; Ding, Y.; et al. 2024. DALK: Dynamic Co-Augmentation of LLMs and KG to answer Alzheimer's Disease Questions with Scientific Literature. *arXiv preprint arXiv:2405.04819*.

Li, Y.; Wang, S.; Ding, H.; and Chen, H. 2023. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, 374–382.

Li, Z.; Yuan, H.; Wang, H.; Cong, G.; and Bing, L. 2025. LLM-R2: A Large Language Model Enhanced Rule-based Rewrite System for Boosting Query Efficiency. *Proceedings of the VLDB Endowment*, 1(18): 53–65.

Liu, L.; Yang, X.; Lei, J.; Liu, X.; Shen, Y.; Zhang, Z.; Wei, P.; Gu, J.; Chu, Z.; Qin, Z.; et al. 2024a. A Survey on Medical Large Language Models: Technology, Application, Trustworthiness, and Future Directions. *arXiv preprint arXiv:2406.03712*.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12: 157–173.

Luo, L.; Li, Y.-F.; Haffari, G.; and Pan, S. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Ma, S.; Xu, C.; Jiang, X.; Li, M.; Qu, H.; Yang, C.; Mao, J.; and Guo, J. 2024. Think-on-Graph 2.0: Deep and Faithful Large Language Model Reasoning with Knowledge-guided Retrieval Augmented Generation. *arXiv preprint arXiv:2407.10805*.

Malkov, Y. A.; and Yashunin, D. A. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4): 824–836.

Mallen, A.; Asai, A.; Zhong, V.; Das, R.; Khashabi, D.; and Hajishirzi, H. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Mavromatis, C.; and Karypis, G. 2024. GNN-RAG: Graph Neural Retrieval for Large Language Model Reasoning. *arXiv preprint arXiv:2405.20139*.

Naeem, Z. A.; Ahmad, M. S.; Eltabakh, M.; Ouzzani, M.; and Tang, N. 2024. RetClean: Retrieval-Based Data Cleaning Using LLMs and Data Lakes. *Proceedings of the VLDB Endowment*, 17(12): 4421–4424.

Narayan, A.; Chami, I.; Orr, L.; and Ré, C. 2022. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment*, 16(4): 738–746.

Nie, Y.; Kong, Y.; Dong, X.; Mulvey, J. M.; Poor, H. V.; Wen, Q.; and Zohren, S. 2024. A Survey of Large Language Models for Financial Applications: Progress, Prospects and Challenges. *arXiv preprint arXiv:2406.11903*.

Nussbaum, Z.; Morris, J. X.; Duderstadt, B.; and Mulyar, A. 2024. Nomic Embed: Training a Reproducible Long Context Text Embedder. arXiv:2402.01613.

Peng, B.; Zhu, Y.; Liu, Y.; Bo, X.; Shi, H.; Hong, C.; Zhang, Y.; and Tang, S. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Qian, Y.; He, Y.; Zhu, R.; Huang, J.; Ma, Z.; Wang, H.; Wang, Y.; Sun, X.; Lian, D.; Ding, B.; et al. 2024. UniDM: A Unified Framework for Data Manipulation with Large Language Models. *Proceedings of Machine Learning and Systems*, 6: 465–482.

Robertson, S. E.; and Walker, S. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of SIGIR '94*, 232–241. Springer.

Ruan, Y.; Fuhry, D.; and Parthasarathy, S. 2013. Efficient community detection in large networks using content and links. In *Proceedings of the 22nd international conference on World Wide Web*, 1089–1098.

Sarthi, P.; Abdullah, S.; Tuli, A.; Khanna, S.; Goldie, A.; and Manning, C. D. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *arXiv preprint arXiv:2401.18059*.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.

Siriwardhana, S.; Weerasekera, R.; Wen, E.; Kaluarachchi, T.; Rana, R.; and Nanayakkara, S. 2023. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11: 1–17.

Sun, J.; Xu, C.; Tang, L.; Wang, S.; Lin, C.; Gong, Y.; Shum, H.-Y.; and Guo, J. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Sun, Z.; Zhou, X.; and Li, G. 2024. R-Bot: An LLM-based Query Rewrite System. *arXiv preprint arXiv:2412.01661*.

Tang, J.; Zhang, Q.; Li, Y.; and Li, J. 2024. Grapharena: Benchmarking large language models on graph computational problems. *arXiv preprint arXiv:2407.00379*.

Tang, Y.; and Yang, Y. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

Traag, V. A.; Waltman, L.; and Van Eck, N. J. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1): 1–12.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and computing*, 17: 395–416.

Wang, K.; Duan, F.; Wang, S.; Li, P.; Xian, Y.; Yin, C.; Rong, W.; and Xiong, Z. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259*.

Wang, S.; Xu, T.; Li, H.; Zhang, C.; Liang, J.; Tang, J.; Yu, P. S.; and Wen, Q. 2024a. Large language models for education: A survey and outlook. *arXiv preprint arXiv:2403.18105*.

Wang, Y.; Lipka, N.; Rossi, R. A.; Siu, A.; Zhang, R.; and Derr, T. 2024b. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19206–19214.

Wu, J.; Zhu, J.; Qi, Y.; Chen, J.; Xu, M.; Menolascina, F.; and Grau, V. 2024a. Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation. *arXiv preprint arXiv:2408.04187*.

Wu, S.; Xiong, Y.; Cui, Y.; Wu, H.; Chen, C.; Yuan, Y.; Huang, L.; Liu, X.; Kuo, T.-W.; Guan, N.; et al. 2024b. Retrieval-augmented generation for natural language processing: A survey. *arXiv preprint arXiv:2407.13193*.

Xu, S.; Pang, L.; Yu, M.; Meng, F.; Shen, H.; Cheng, X.; and Zhou, J. 2024. Unsupervised Information Refinement Training of Large Language Models for Retrieval-Augmented Generation. *arXiv preprint arXiv:2402.18150*.

Xu, X.; Yuruk, N.; Feng, Z.; and Schweiger, T. A. 2007. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 824–833.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yu, H.; Gan, A.; Zhang, K.; Tong, S.; Liu, Q.; and Liu, Z. 2024. Evaluation of Retrieval-Augmented Generation: A Survey. *arXiv preprint arXiv:2405.07437*.

Zhang, N.; Choubey, P. K.; Fabbri, A.; Bernadett-Shapiro, G.; Zhang, R.; Mitra, P.; Xiong, C.; and Wu, C.-S. 2024a. SiReRAG: Indexing Similar and Related Information for Multihop Reasoning. *arXiv preprint arXiv:2412.06206*.

Zhang, Q.; Hong, X.; Tang, J.; Chen, N.; Li, Y.; Li, W.; Tang, J.; and Li, J. 2024b. Gcoder: Improving large language model for generalized graph problem solving. *arXiv preprint arXiv:2410.19084*.

Zhao, P.; Zhang, H.; Yu, Q.; Wang, Z.; Geng, Y.; Fu, F.; Yang, L.; Zhang, W.; and Cui, B. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Zheng, Y.; Gan, W.; Chen, Z.; Qi, Z.; Liang, Q.; and Yu, P. S. 2024. Large language models for medicine: a survey. *International Journal of Machine Learning and Cybernetics*, 1–26.

Zhou, Y.; Cheng, H.; and Yu, J. X. 2009. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2(1): 718–729.

Zhu, Y.; Wang, X.; Chen, J.; Qiao, S.; Ou, Y.; Yao, Y.; Deng, S.; Chen, H.; and Zhang, N. 2024. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27(5): 58.