LEGO-GraphRAG: Modularizing Graph-based Retrieval-Augmented Generation for Design Space Exploration

Yukun Cao
University of Science and Technology of China
ykcho@mail.ustc.edu.cn

Zengyi Gao
University of Science and Technology of China
gzy02@mail.ustc.edu.cn

Zhiyang Li
University of Science and Technology of China
lizhiyang215@gmail.com

Xike Xie
University of Science and Technology of China
xkxie@ustc.edu.cn

S. Kevin Zhou
University of Science and Technology of China
s.kevin.zhou@gmail.com

Jianliang Xu
Hong Kong Baptist University
xujl@comp.hkbu.edu.hk

ABSTRACT

GraphRAG integrates (knowledge) graphs with large language models (LLMs) to improve reasoning accuracy and contextual relevance. Despite its promising applications and strong relevance to multiple research communities, such as databases and natural language processing, GraphRAG currently lacks modular workflow analysis, systematic solution frameworks, and insightful empirical studies. To bridge these gaps, we propose **LEGO-GraphRAG**, a modular framework that enables: **1)** fine-grained decomposition of the GraphRAG workflow, **2)** systematic classification of existing techniques and implemented GraphRAG instances, and **3)** creation of new GraphRAG instances. Our framework facilitates comprehensive empirical studies of GraphRAG on large-scale real-world graphs and diverse query sets, revealing insights into balancing reasoning quality, runtime efficiency, and token or GPU cost, that are essential for building advanced GraphRAG systems.

PVLDB Reference Format:
Yukun Cao, Zengyi Gao, Zhiyang Li, Xike Xie, S. Kevin Zhou, and Jianliang Xu. LEGO-GraphRAG: Modularizing Graph-based Retrieval-Augmented Generation for Design Space Exploration. PVLDB, 18(10): 3269 - 3283, 2025.
doi:10.14778/3748191.3748194

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/gzy02/LEGO-GraphRAG.

1 INTRODUCTION

Recent advancements in large language models (LLMs) have highlighted their strengths in semantic understanding and contextual reasoning, enabled by extensive pre-training on vast corpora. Despite these strengths, LLMs often struggle with domain-specific queries and complex contexts, frequently generating "hallucinations", in which outputs appear credible but lack factual accuracy. Retrieval-augmented generation (RAG) addresses these limitations by integrating external knowledge to enhance factual accuracy and contextual relevance. Early RAG implementations, however, often relied on document retrieval methods that introduced noise and excessive context [26, 55, 60, 122]. This limitation has led to a growing focus on graph-based RAG systems, known as *GraphRAG* [22, 27, 42, 61].

In GraphRAG, conventional document retrieval is replaced by graph-based retrieval, leveraging the structured and relational nature of graphs to extract query-specific reasoning paths that provide more precise and contextually relevant support for LLM reasoning. Typically, the GraphRAG workflow consists of two primary phases:

- **Retrieval:** This phase retrieves knowledge from graphs by identifying reasoning paths relevant to the query.
- **Augmented generation:** The retrieved reasoning paths are used to augment the LLM prompt, enhancing its ability in reasoning and generating accurate and contextually relevant outputs.

Existing GraphRAG studies [22, 45, 50, 65, 71, 72, 74, 98] have demonstrated the potential of integrating graph data management techniques with GraphRAG. However, despite their promise, these studies are still in their early stages and face two significant challenges. First, they lack foundational support in addressing the scalability of graph algorithms, which is crucial for managing large-scale graphs and reducing query latency in real-world applications. Second, there is a knowledge gap regarding the impact of semantic information on graphs on the performance of GraphRAG. Specifically, it is unclear which strategies are most effective for leveraging semantic information from both the query and the graph, and which type of semantic model is best suited for different query scenarios. Resolving these uncertainties is essential to improving the quality, efficiency, and cost of GraphRAG across various query scenarios.

Our Motivation. Building on the strengths of database research, we aim to address key challenges in GraphRAG and lay the groundwork for future advancements. While efforts like Modular RAG [28] have modularized general RAG, they fall short of addressing the GraphRAG's workflows and needs. We identify three critical gaps:

- **Need for a Unified Framework for Categorizing and Analyzing GraphRAG Solutions.** GraphRAG solutions consist of diverse technologies, including algorithmic approaches (e.g., random walk, PageRank) and neural network-based methods

Proceedings of the VLDB Endowment, Vol. 18, No. 10 ISSN 2150-8097.
doi:10.14778/3748191.3748194

(e.g., end-to-end and LLM models), each playing a distinct role in their functionality. The lack of a unified categorization for systematically analyzing these techniques—despite prior efforts such as Modular RAG [28] and other surveys [24, 35, 86, 126] on LLMs with graphs—hinders effective research summarization and the identification of promising candidates for further study.
- **Need for Modular Retrieval in GraphRAG.** Current implementations of GraphRAG often treat the retrieval phase as a single, monolithic process, making it difficult to isolate, analyze, and optimize individual components. A modular approach is desirable to facilitate the development of more efficient and effective retrieval mechanisms. Moreover, while Modular RAG [28] modularizes general RAG workflows, it neither dissects the core retrieval process of GraphRAG nor clarifies its distinction from general RAG.
- **Absence of a GraphRAG Testbed for New Instance Design and Evaluation.** Optimizing GraphRAG performance requires navigating a complex trade-off between runtime efficiency and reasoning accuracy, influenced by various design factors. A GraphRAG testbed is needed to generate diverse implementation instances, allowing users to evaluate, compare, and apply different approaches while providing clear guidelines and trade-offs for constructing optimal GraphRAG instances tailored to specific scenarios. However, Modular RAG [28] and related surveys [24, 35, 86, 126] focus on conceptual overviews and lack implementation or evaluation support for GraphRAG.

Our Contributions. We presents the first empirical study on GraphRAG and introduces *LEGO-GraphRAG*, a unified and modular framework that provides insights to guide future research through contributions in both framework design and empirical analysis.

For Framework Design:
- We propose LEGO-GraphRAG, a modular framework dividing the retrieval phase into two flexible modules: *subgraph-extraction* and *path-retrieval*, and classifies the techniques into *structure-based* and *semantic-augmented* methods for each module.
- LEGO-GraphRAG's modular framework, with categorized techniques, supports implementing all existing GraphRAG instances while enabling the creation of new ones, promoting both standardization and innovation.
- We identify essential design factors, including reasoning quality, efficiency, and cost, providing a structured trade-off analysis to guide the development of GraphRAG instances.

For Empirical Research:
- We study a comprehensive set of GraphRAG instances, including 7 existing implementations and 16 new instances generated by LEGO-GraphRAG. These instances were extensively evaluated on large-scale real graphs (i.e., Freebase) and 5 commonly used GraphRAG query sets, covering various query scenarios.
- We suggest several modular configurations and a number of improvements to existing implementations for improved reasoning quality and balancing efficiency and cost.

1.1 Related Works

1.1.1 RAG and GraphRAG. Early text-based RAG systems rely on basic retrieval techniques, such as text chunking and cosine similarity for ranking [60], which are prone to retrieving noisy and irrelevant information, leading to lower-quality results [26]. To address this, recent works have introduced both pre- and post-retrieval improvements. Pre-retrieval methods [128] enhance the input query, while post-retrieval techniques [21, 124] refine the ranking and filtering of retrieved results. Toolkits like LlamaIndex and LangChain [10, 66] offer modular control over these stages, improving overall retrieval precision and system interpretability [26]. However, refining the retrieval stage to reliably suppress irrelevant content remains a key challenge [22].

GraphRAG has recently emerged as a promising direction by representing knowledge in graph form, enabling more structured retrieval, multi-hop reasoning with better interpretability [22, 34, 123]. Compared to unstructured text retrieval, graph-based retrieval offers reduced noise, better coverage of entity relations, and lowered token overhead during inference [22, 75, 93, 101].[1] Existing instances (or implementations) in GraphRAG have drawn heavily from knowledge-base question answering (KBQA) techniques, utilizing both information retrieval methods [78, 92] and semantic parsing models [14, 59] to identify relevant subgraphs or reasoning paths. Microsoft GraphRAG [22] was an early effort exploring the advantages of graph-based over text-based retrieval, focusing on graph construction from text and precomputation. GCR [72] combines LLMs and beam search to refine reasoning paths, improving retrieval alignment with queries. RoG [71] and GSR [45] employ Personalized PageRank (PPR) to retrieve query-specific subgraphs, while KELP [65] refines reasoning paths using fine-tuned models like BERT [20]. These studies highlight the diverse graph-based and neural network-based techniques employed in GraphRAG, showcasing the extensive potential of graphs in enhancing the reasoning quality of LLMs.

1.1.2 LLMs with (Knowledge) Graphs. GraphRAG aligns with the broader research to integrate LLMs with structured knowledge, such as KGs. Surveys [56, 84] outline three paradigms: KG-enhanced LLMs, LLM-augmented KGs, and their joint integration. Moreover, [41] reviews knowledge-enhanced pre-trained language models (e.g., LLMs) for language understanding and generation, and [52] examines LLMs as interfaces for data pipelines, highlighting their integration with KGs in AI systems. Our work fits within the KG-enhanced LLMs paradigm and aims to contribute a modular GraphRAG framework to support systematic design and evaluation of reasoning-augmented LLMs with graphs.

2 PRELIMINARIES

2.1 Problem Formalization

GraphRAG is designed to integrate structured knowledge into the reasoning process of LLMs, enhancing their ability to generate content that is both accurate and contextually relevant. In GraphRAG, knowledge is typically represented in the form of Text-Attributed Graphs (TAGs), where both nodes and edges are enriched with textual information, with knowledge graphs serving as a typical example. For clarity and without loss of generality, the graphs referred to in this paper will follow the structure outlined in Definition 1.

[1] A detailed comparison between GraphRAG and text-based RAG is provided in our technical report B.10.

Definition 1 (**Graph in GraphRAG**). *In GraphRAG, a graph is defined as a directed labeled graph $G = (V, E)$, where V represents the set of nodes (entities), and E denotes the set of directed edges that signify relations between those entities. Each node $v \in V$ and each edge $e \in E$ carry semantic information. Specifically, nodes represent entities with associated semantic attributes (e.g., descriptions), while edges represent relations with semantic contexts (e.g., relation types).*

Formally, the graph in GraphRAG is represented as a collection of triples: $G = \{\tau_i = (s_i, r_i, t_i) \mid s_i, t_i \in V, r_i \in E\}$, where each $\tau_i = (s_i, r_i, t_i)$ is a triple representing that the source entity s_i is connected to the target entity t_i by the relation r_i, encapsulating structured, domain-specific knowledge. While a single triple τ_i is a basic unit of knowledge, a sequence of connected triples forms reasoning paths that support more advanced inferences.

In practice, graphs are built from domain knowledge or text (e.g., Microsoft GraphRAG [22] with auxiliary summaries; see Definition 2). As text-to-graph construction is well-studied [11, 22, 32, 33, 64] and not central to our framework, we treat it as optional but include a GraphRAG instance with textual information in our analysis (Section 4.2 e).

Definition 2 (**Graph Construction (*Optional*)**). *In GraphRAG, graph construction is an optional step that builds a text-attributed graph from large-scale text, and may also precompute subgraphs and summaries to enhance GraphRAG. Specifically:*

- ***Construction:** Identify entities and relations from text and represent them as nodes and edges in the graph.*
- ***Precomputation** (Optional)**:** Compute subgraphs and relevant textual summaries from the constructed graph, such as via graph clustering or community detection.*

With text-attributed auxiliary graphs, GraphRAG operates in two main phases: *retrieval* and *augmented generation* phases. During the retrieval phase (Definition 3), the system extracts relevant entities and reasoning paths from the graph, aligning them with the query context. In the augmented generation phase (Definition 4), these retrieved reasoning paths are incorporated to enhance LLM's content generation capabilities.

Definition 3 (**Retrieval Phase**). *This phase starts by processing the query q to extract relevant entities and relations that correspond to nodes and edges in the graph G: $Extract(q, G) \rightarrow \epsilon_q = (\{v_i^{(q)}\}, \{e_i^{(q)}\})$, where ϵ_q is the set of entities and relations extracted.*[2]

Using the extracted entities $\{v_i^{(q)}\}$ and relations $\{e_i^{(q)}\}$ in ϵ_q, a variety of methods (e.g., search algorithms, neural network-based retrieval/ranking models) are applied to retrieve reasoning paths $\mathcal{P}_q$ that connect ϵ_q to potential target answers, potentially spanning multiple hops. The retrieval process is thus formalized by: $Retrieve(\epsilon_q, G) \rightarrow \mathcal{P}_q = \{P_i^{(q)}\}$ where each reasoning path $P_i^{(q)} \in \mathcal{P}_q$ of length k is represented as a sequence of triples $\langle \tau_1, \tau_2, \ldots, \tau_k \rangle$:

$$P_i^{(q)} = \langle \tau_1, \tau_2, \ldots, \tau_k \rangle = \langle (s_1, r_1, t_1), (t_1, r_2, t_2), \ldots, (t_{k-1}, r_k, t_k) \rangle$$

[2]Extracting entities/relations from the semantic context of q has been well-studied [38, 50, 92] and is beyond the scope of this paper. Following prior works on reasoning tasks in graphs [4, 16, 79, 92, 119], queries are typically categorized by the minimum number of reasoning hops required to reach the answer from the query entities or relations (e.g., one-hop queries or multi-hop queries). In addition, queries can also be categorized by the number of answers and the number of query entities involved.

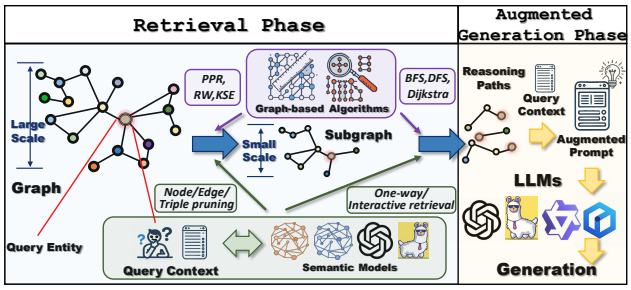

Figure 1: The GraphRAG Workflow

Definition 4 (**Augmented Generation Phase**). *In this phase, the retrieved reasoning paths $\mathcal{P}_q$ are merged with the original query q, forming an augmented prompt q': $Augment(q, \mathcal{P}_q) \rightarrow q'$. This augmented prompt is then input to the LLMs, enhancing their ability to generate more accurate and contextually relevant content for the final answer: $Generate(q', LLM) \rightarrow$ final answer.*

Figure 1 illustrates the GraphRAG workflow, where retrieval forms the foundation by supplying key reasoning paths for LLM augmentation and generation. As these stages hinge on retrieval quality and LLM strength, this work focuses on analyzing the modular design and solution space of the retrieval phase.

3 LEGO-GRAPHRAG FRAMEWORK

3.1 Overview

Table 1 outlines the design framework of LEGO-GraphRAG, structured along two key dimensions: **modules** and **method types**. Specifically, the core retrieval phase is divided into two modules: the *subgraph-extraction* module and the *path-retrieval* module. Each module incorporates two solution types: *structure-based* methods and *semantic-augmented* methods. Accordingly, GraphRAG instances within the LEGO-GraphRAG framework are classified into five distinct groups based on the combinations of modules and solution types used. Table 2 provides a detailed breakdown of these groups, encompassing both existing methods and new instances developed in our empirical study. In general, the LEGO-GraphRAG framework is materialized by two key considerations.

- **Natural Segmentation of the GraphRAG Process.** The reasoning path retrieval process is naturally segmented into two phases: a) extracting a query-relevant subgraph to scale down the search space; and b) meticulously retrieving reasoning paths from the extracted subgraph.
- **Facilitation of GraphRAG Research Analysis.** The framework enables a comprehensive analysis of recent GraphRAG advancements, which predominantly focus on enhancing one or both modules, as summarized in Table 2.

3.1.1 Subgraph-Extraction vs. Path-Retrieval Modules.

The subgraph-extraction (SE) module aims to enhance the effectiveness and efficiency of reasoning path retrieval by reducing the search space from the full graph to a smaller set of query-relevant subgraphs. The path-retrieve (PR) module then operates on these subgraphs to retrieve reasoning paths using specific methods. Based on the retrieval phase of GraphRAG formalized in Definition 3, the two modules are formalized as follows.

Definition 5 (**Subgraph-Extraction (SE)**). *Given a query q, a graph $G = (V, E)$, and a set of entities and relations ϵ_q derived from specified query q, the SE module aims to extract a query-specific subgraph $g_q \subseteq G$, defined as: $SE(G, q, \epsilon_q) \rightarrow g_q$.*

Definition 6 (**Path-Retrieval (PR)**). *Given an extracted subgraph $g_q \subseteq G$ and the entity and relation set ϵ_q derived from query q, the process of this module obtains a reasoning path set $\mathcal{P}_q = \{P_i^{(q)}\}$ with the following process: $PR(G \text{ or } g_q, q, \epsilon_q) \rightarrow \mathcal{P}_q = \{P_i^{(q)}\}$.*

Of the two modules, the *path-retrieval* module is essential for any GraphRAG instance, while the *subgraph-extraction* module is optional. The optionality of the *subgraph-extraction* module depends largely on the graph size, primarily due to computational efficiency. For smaller graphs, where the node and edge space remain manageable, direct retrieval of reasoning paths is feasible, making the *subgraph-extraction* module optional. However, for large-scale graphs, the exponential growth in the number of nodes, edges, and paths significantly complicates the retrieval [23, 53, 54, 67]. The *subgraph-extraction* module addresses this by extracting a query-relevant subgraph, reducing the retrieval space and improving computational efficiency and relevance. After passing through these modules, the refined reasoning paths enhance the reasoning abilities of LLMs during the generation phase, as detailed in Definition 4.

3.1.2 **Structure-based vs. Semantic-augmented Methods.** For each module, systematically classifying and summarizing potential solutions is important, as it enables the identification of potential strategies and provides a deeper understanding of existing implementations. At their core, the workflows of the *subgraph-extraction* and *path-retrieval* modules share potential similarities, as both involve retrieving a small-scale but query-relevant subset from the graph. Accordingly, the design solutions for these modules can be interwoven and further categorized into two complementary categories: *structure-based methods*, focusing on the graph's topological properties, and *semantic-augmented methods*, utilizing the semantic information of both the graph and the query, as detailed in Table 1.

Structure-based methods (SBE and SBR) use graph algorithms [25, 57, 69, 99, 113, 118] to iteratively explore nodes (entities) and edges (relations), constructing subgraphs or reasoning paths. Semantic-augmented methods (SAE, OSAR, and ISAR) exploit the semantic relevance between the query and the nodes, edges, and triples of graphs, using two types of semantic models: end-to-end models (EEMs) [88, 89, 104] and large language models (LLMs) [8, 98, 127]. The two types of models are applied differently. SAE/OSAR methods refine candidate subgraphs/paths for better semantic alignment. ISAR methods incorporate semantic evaluation directly into the path-retrieval process, using EEMs/LLMs to interactively identify relevant reasoning paths. In the sequel, we investigate the design solutions of the two modules, subgraph-extraction (Section 3.2) and path-retrieval (Section 3.3), with the aforementioned two types of methods, structure-based and semantic-augmented methods.

3.2 Design Solutions of Subgraph-Extraction

The design solution of the subgraph-extraction module is comprised of *structure-based extraction* (**SBE** in Section 3.2.1) and *semantic-based extraction* (**SAE** in Section 3.2.2).

3.2.1 **Structure-Based Extraction (SBE).** These solutions exploit the structural information of a graph $G = (V, E)$, starting with the query-relevant entities $\{v_i^{(q)}\} \in \epsilon_q$,[3] to identify corresponding nodes (entities) and edges (relations) that are pertinent to the query q. This process constructs smaller subgraphs centered around each $v_i^{(q)}$, which are then aggregated to produce the query-related subgraph g_q. Formally, the procedure is defined as:

$$SE(G, q, \epsilon_q) \rightarrow g_q = \bigcup\nolimits_{v_i^{(q)} \in \epsilon_q} g_{(q, v_i^{(q)})}$$

Here, g_q represents the union of the subgraphs $g_{(q, v_i^{(q)})}$, each focusing on an entity $v_i^{(q)}$ in ϵ_q. For each query entity, three extraction strategies can be applied: *random walk-based*, *neighborhood-based*, and *structural importance-based* strategies.

a) Random Walk-based. The *random walk (RW)* algorithm [87, 116] and its variants [25, 83, 85, 94, 95] enumerate all entities in ϵ_q, and, for each entity $v_i^{(q)}$, iteratively expand a subgraph by randomly selecting edges and nodes at each step. This simple yet effective strategy gradually constructs a subgraph of the desired size, making it a commonly used baseline method [38, 45, 71, 72].

b) Neighborhood-based. This strategy aims to capture the local structure around an entity $v_i^{(q)} \in \epsilon_q$ by expanding its neighborhood. The most representative method is *K-hop subgraph-extraction (KSE)* [38], where the subgraph is constructed by including all nodes v_j and edges e_{ij} such that the shortest path distance $d(v_i^{(q)}, v_j) \leq K$. This approach ensures that the subgraph captures the immediate relational context surrounding the query entity, progressively including nodes and edges within the specified radius (i.e., K-hops).

c) Structural Importance-based. These methods [37, 82, 118] identify nodes based on their structural importance relative to an entity $v_i^{(q)}$. Nodes are ranked by importance scores, and a subset of high-ranking N_{ppr} nodes, along with their directly connected edges, are selected to form the subgraph. Prominent methods, such as *PageRank* [82] and *Personalized PageRank (PPR)* [37], compute a relevance score for each node based on topological position, link density and centrality. Specifically, PageRank assigns an importance score $S(v_j)$ to each node v_j, which is iteratively updated based on both its local connectivity and the scores of its neighboring nodes, thereby capturing its global significance. Personalized PageRank (PPR) enhances this process by "personalizing" the distribution towards the query entity $v_i^{(q)}$, ensuring that nodes closer to $v_i^{(q)}$ receive higher scores. While PPR is a widely adopted approach for SBE solutions, its computational cost scales with the size of the graph, presenting a considerable bottleneck for real-time GraphRAG pipelines. Section 4.3 discusses this efficiency challenge in detail.

3.2.2 **Semantic-Augmented Extraction (SAE).** Semantic models play a key role in facilitating subgraph-extraction by assessing the semantic relevance of graph components. These models fall into two main categories: End-to-End Models (EEMs) and Large Language Models (LLMs).

a) End-to-End Models (EEMs). EEMs process the text associated with graph components (i.e., nodes, edges, triples) and queries to compute semantic relevance. They either output statistical properties and embeddings for relevance calculation or directly provide relevance scores. Formally, for nodes, edges, or triples $v, e,$ or $\tau \in G$

[3] While subgraphs can be built from relationship $e_i^{(q)} \in \epsilon_q$, practical methods predominantly use entities (nodes) as the basis for subgraph construction [45, 65, 70, 72, 98].

Table 1: Design Solutions of LEGO-GraphRAG Framework: *The framework comprises of four groups of instances: (I) SBE & SBR; (II) SBE & I/OSAR; (III) SAE & SBR; (IV) SAE & I/OSAR. When the subgraph-extraction module is optional, an additional group, (V) SBR or I/OSAR, is included.*

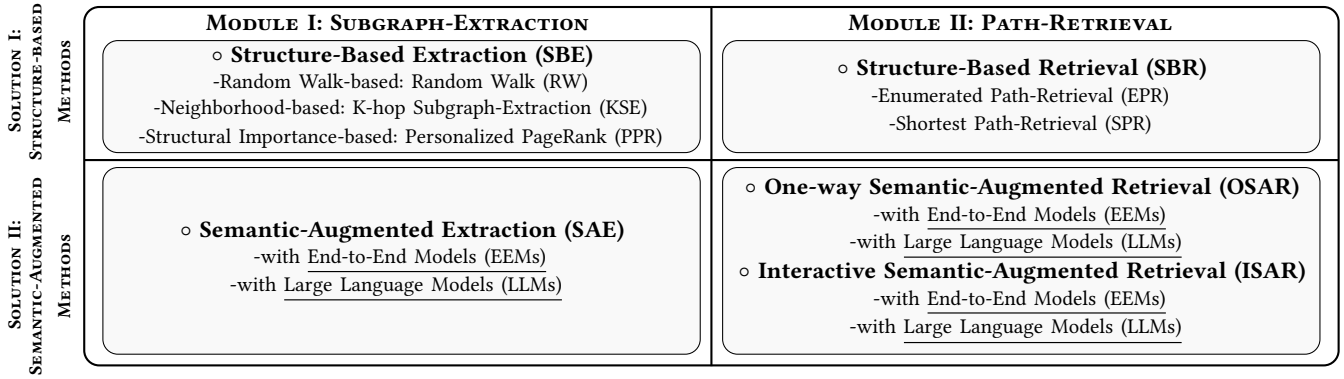

Table 2: Five Groups of Instances under the LEGO-GraphRAG Framework

Group	Subgraph-Extraction	Path-Retrieval	Implemented Instances
Structure-based Methods on Both Modules (Group (I): *SBE & SBR***)**	SBE (-RW/KSE/PPR)	SBR (-EPR/SPR)	Our Instance: No.1
Semantic-Augmented Methods on Both Modules (Group (II): *SAE & I/OSAR***)**	SAE (-EEMs/LLMs)	OSAR (-EEMs/LLMs)	Our Instances: No.2, 3, 4, 5
	SAE (-EEMs/LLMs)	ISAR (-EEMs/LLMs)	GCR (arXiv24) [72] Our Instances: No.6, 7, 8, 9
Semantic-Augmented Methods on Subgraph-Extraction (Group (III): *SAE & SBR***)**	SAE (-EEMs/LLMs)	SBR (-EPR/SPR)	RoG (ICLR24) [71] GSR (EMNLP24) [45] Our Instances: No.10, 11
Semantic-Augmented Methods on Path-Retrieval (Group (IV): *SBE & I/OSAR***)**	SBE (-RW/KSE/PPR)	OSAR (-EEMs/LLMs)	Our Instances: No.12, 13
	SBE (-RW/KSE/PPR)	ISAR (-EEMs/LLMs)	StructGPT (EMNLP23) [50] Our Instances: No.14, 15
Without Subgraph-Extraction Modules (Group (V): *SBR or I/OSAR***)**	None	SBR (-EPR/SPR)	-
	None	OSAR (-EEMs/LLMs)	KELP (ACL24) [65]
	None	ISAR (-EEMs/LLMs)	ToG (ICLR24) [98], DoG (arXiv24) [74]

and a query q, the goal is: EEMs(v or e or τ, q) $\rightarrow$ scores. Using these scores, the most relevant graph components are identified for subgraph-extraction. EEM can be categorized into two main types: *Statistical Models* [73, 90, 91] and *Embedding Models/Re-Rankers* [13, 88]. The former relies on statistical computations, while the latter consists of end-to-end neural network models to produce embeddings or rank relevance.

a).1 Statistical Models. Statistical models are foundational in text analysis and semantic modeling, relying on metrics like term frequency (TF) and inverse document frequency (IDF) [91]. Popular methods include *BM25* [90] for ranking relevance in information retrieval, *Latent Semantic Analysis (LSA)* [18] for uncovering latent semantic structures via dimensionality reduction, and *Latent Dirichlet Allocation (LDA)* [5] for topic modeling, which aids semantic similarity through topic distributions.

For example, BM25 treats textual information attached to graph components (nodes, edges, or triples) as "documents" within a corpus (the graph). During precomputation, the text of each graph component is tokenized, enabling the calculation of TF and IDF. When a query is issued, its text undergoes tokenization, and BM25 computes a relevance score for each graph component based on the query terms and their frequency distributions across the graph. Statistical models are efficient, with low computational costs and short execution times. They are suitable for simpler semantic tasks or as baselines in semantic-augmented workflows.

a).2 Embedding Models/Re-Rankers. Embedding models and re-rankers use pre-trained neural network (NN) models with smaller parameter sizes (millions) compared to LLMs (billions). Trained on large corpora, they generate dense embeddings that encode rich semantic information [88]. Embedding models (e.g., Sentence-Transformers [111]) independently generate embeddings for queries and graph components (nodes, edges, triples). Semantic similarity is computed using metrics [9, 48] like cosine similarity:

$$\text{ST}(v \text{ or } e \text{ or } \tau, q) \rightarrow \cos(\mathbf{emb}(v \text{ or } e \text{ or } \tau), \mathbf{emb}(q))$$

Re-Rankers (e.g., BGE-Reranker [13]) take combined query and graph components as inputs and output a similarity score, allowing for deeper contextual interactions:

$$\text{BGE}(v \text{ or } e \text{ or } \tau, q) \rightarrow \text{NeuralNetworks}(\mathbf{emb}(v \text{ or } e \text{ or } \tau) \oplus \mathbf{emb}(q))$$

These models generally require longer execution and pre-training times compared to statistical models. However, their balance of efficiency and semantic precision makes them a common choice for semantic modeling in GraphRAG. Domain-specific fine-tuning can further enhance these models' performance [45, 65, 71, 72], though it comes at an additional cost.

a).3 SAE with EEMs. In the subgraph-extraction module, EEMs prune nodes, edges, or triples from graph G based on their semantic relevance to the query, reducing computational overhead. However, directly applying EEMs to large graphs is costly. Thus, it is desired to have a *pre-filtering* step before applying semantic models, which

extracts a smaller subgraph $g \in G$ (e.g., via SBE methods-PPR) to focus on relevant components [45, 65]. Thus, this subgraph pruning process generally follows three key strategies: *node pruning*, *edge pruning*, and *triple pruning*.

- Node Pruning (NP): Nodes $v \in g$ are ranked by semantic relevance scores to query q. Top N_v nodes are retained, ensuring connected edges are included: $g_{(q,v)} = \{v \mid v \text{ in top } N_v\} \cup \{e \mid e \text{ connects } g_{(q,v)}\}$.
- Edge Pruning (EP): Edges $e \in g$ are scored, with top N_e edges retained. Disconnected nodes are removed: $g_{(q,e)} = \{e \mid e \text{ in top } N_e\} \cup \{v \mid v \text{ connected by } g_{(q,e)}\}$.
- Triple Pruning (TP): Top N_τ triples $\tau \in g$ are selected, forming a minimal subgraph containing these triples: $g_{(q,\tau)} = \{\tau \mid \tau \text{ in top } N_\tau\}$.

Batch processing can be used for EEMs to efficiently compute relevance scores for all graph components in g, streamlining subgraph-extraction:

$$SE(G, q, \epsilon_q), EEMs \rightarrow g_q = g_{(q,v/e/\tau)}$$

b) Large Language Models (LLMs). LLMs, such as Llama [102, 103, 105] , Qwen [117], and GPT [7, 80], are large pre-trained language models known for their ability to capture nuanced semantics and contextual understanding due to extensive training on large-scale corpora. Unlike EEMs, LLMs provide flexible evaluation of graph components (nodes, edges, triples) through prompt-based interactions. For instance, to evaluate the node relevance to a query, LLMs can generate relevance scores or directly a ranked list of entities. Formally, this process is represented as:

$$\text{LLMs}(v \text{ or } e \text{ or } \tau, q, \text{Prompt}) \rightarrow \text{scores or ranked list}$$

In practice, LLM outputs often fall short of prompt requirements due to output length limits and inherent limitations (e.g., hallucination [44]). Generated entity lists (from subgraph-extraction module) or reasoning paths (from path-retrieval module) may be sparse or low-quality. Mitigation approaches include: prompt tuning [76], fine-tuning [15] for long-context understanding, or adopting more capable LLMs. Moreover, we explore some general purpose strategies to address this challenge in Section 5.3.

While fine-tuning LLMs on domain-specific knowledge improves performance for targeted queries, it is more resource-intensive than tuning embedding models or re-rankers[4] and may generalize poorly across scenarios [61]. LLMs also have high inference costs due to the model size and autoregressive decoding. Hence, we recommend using them selectively and offer non-LLM alternatives in Section 4.2. Cost-efficient fine-tuning methods (e.g., LoRA [40], QLoRA [19]) and inference optimizations (e.g., pruning [115], quantization [47], KV caching [49], parallel decoding [12], and semantic compression [68]) further reduce overhead.

b).1 SAE with LLMs. The subgraph extraction process using LLMs mirrors that of EEMs and can be formalized as follows:

$$SE(G, q, \epsilon_q), LLMs, Prompts \rightarrow g_q = g_{(q,v/e/\tau)}$$

Even on subgraph g, LLM-based semantic evaluation remains costly and faces token constraints, making it hard to process g with a single LLM call. To mitigate this, an additional *pre-filtering* step is applied, where candidate components within g can be ranked and pruned via lightweight heuristics (e.g., random selection or EEM-based similarity selection) before invoking LLMs. Subsequently, the refined g can be passed to LLMs for evaluation within a single call.[5]

b).2 Other Methods with LLMs and Limitations. In certain scenarios, LLMs fine-tuned on domain knowledge can act as agents [70, 71] to generate extraction rules (e.g., SPARQL queries [1, 43, 70]). For example, in response to a query, RoG [71] utilizes LLMs fine-tuned on domain-specific knowledge graphs to identify key relationships between entities, which are then used to construct SPARQL queries that extract relevant triples, forming the query-relevant subgraph. The subgraph containing these triples ultimately serves as the query-relevant subgraph. While LLMs offer direct subgraph-extraction without pruning, these approaches are limited by high computational costs, reliance on accurate templates, and reduced generalizability for diverse queries. Fine-tuning LLMs for domain-specific tasks further adds to the resource burden, making this approach less scalable for subgraph-extraction on large-scale graphs.

3.3 Design Solutions of Path-Retrieval

The design solution of the path-retrieval module is comprised of *structure-based retrieval* (**SBR** in Section 3.3.1), *one-way semantic-augmented retrieval* (**OSAR** in Section 3.3.2), and *interactive semantic augmented retrieval* (**ISAR** in Section 3.3.3).

3.3.1 **Structure-Based Retrieval (SBR)**. These methods operate on either the original graph or extracted subgraphs, leveraging graph algorithms to identify reasoning paths. Starting from the query-relevant entities, these algorithms iteratively traverse node connections to generate candidate paths, utilizing techniques such as Breadth-First Search (BFS), Depth-First Search (DFS), Dijkstra's algorithm, and their variants. The procedure is formally defined as:

$$PR(G \text{ or } g_q, q, \epsilon_q) \rightarrow \mathcal{P}_q = \bigcup\nolimits_{v_i^{(q)} \in \epsilon_q} \mathcal{P}_{(q,v_i^{(q)})}$$

Here, $\mathcal{P}_q$ represents the union of the reasoning path sets $\mathcal{P}_{(q,v_i^{(q)})}$, each focusing on an entity $v_i^{(q)}$ in ϵ_q. Two primary approaches are applied to extract paths for each query entity: *enumerated path-Retrieval* and *shortest path-Retrieval*.

a) Enumerated Path-Retrieval (EPR). EPR [65] enumerates all possible paths from an entity $v_i^{(q)} \in \epsilon_q$ to every reachable node within the graph or subgraph. It captures diverse reasoning chains, providing additional context for LLMs, although it may introduce information that is not related to the query, potentially adding noise to the reasoning process of the LLMs.

b) Shortest Path-Retrieval (SPR). SPR [36, 92, 97, 112, 121] identifies all shortest paths from an entity $v_i^{(q)} \in \epsilon_q$ to reachable nodes, ensuring that the extracted paths are concise and directly relevant to the query.

3.3.2 **One-way Semantic-Augmented Retrieval (OSAR)**. Similar to the SAE methods in the subgraph-extraction module, the OSAR methods leverage semantic models to evaluate and select the N_p most relevant paths from a candidate path set $\mathcal{P} \in G$ or g_q,[6] to construct a refined set of reasoning paths $\mathcal{P}_q$ that are most relevant

[4] Details about fine-tuning for both EEMs and LLMs are in the technical report B.1.

[5] Multiple LLM calls can be used to evaluate all graph components in g, but this incurs substantial overhead during subgraph-extraction and is typically avoided.

[6] $\mathcal{P}$ is typically obtained from SBR methods, such as EPR or SPR.

to the query q. Path semantic relevance can be evaluated using either EEMs or LLMs, as outlined in Section 3.2.2. Note that when using LLMs to evaluate and select reasoning paths, it may not be feasible to input all paths in $\mathcal{P}$ in a single call. In such cases, random selection or EEMs can be used as a preliminary filter to reduce the number of candidate paths.[7]

3.3.3 **Interactive Semantic-Augmented Retrieval (ISAR).** Unlike OSAR, which separates path generation and evaluation, ISAR integrates both within an interactive framework. Using interactive search algorithms [81, 96, 106] combined with semantic models, ISAR directs the search process toward semantically relevant graph regions from the outset. By interactively and dynamically evaluating and prioritizing path extensions based on their relevance to the query, ISAR effectively narrows the search space during execution.

For example, in Beam Search, [8] which is widely used in GraphRAG workflow [16, 98], a fixed number of candidate paths (beams) are explored iteratively to identify the most relevant reasoning path for a query entity $v_i^{(q)} \in \epsilon_q$. At each step, potential path extensions are evaluated for semantic relevance using a greedy strategy, and the top-B paths are retained. This process continues until a stopping criterion is met, e.g., reaching the maximum path length L or exhausting relevant extensions. The final output consists of the most relevant reasoning paths discovered during the search.

In ISAR, semantic models for evaluating path extensions can be EEMs, LLMs, or a hybrid of both to combine their strengths. Based on practical exploration, we propose three hybrid strategies:

- **LLMs Refinement (LLMs-R, in ISAR-EEMs):** After the ISAR-EEMs search process, LLMs are used to refine the retrieved paths by reducing redundancy and improving semantic coherence.
- **EEMs Pre-filter (EEMs-PF, in ISAR-LLMs):** At each step of the LLMs-based search, EEMs pre-rank and filter candidate extensions before passing them to the LLMs, reducing input size and improving efficiency.
- **EEMs Supplement (EEMs-S, in ISAR-LLMs):** At each step of path expansion, if the LLMs generate too few or low-quality candidates, EEMs are used to supplement the expansion with top-ranked relevant paths.

Additionally, for small-scale graphs, LLM-agent-based ISAR methods [74] can generate reasoning paths through single- or multi-turn interactions, if LLMs are fine-tuned on domain-specific knowledge. Despite domain-specific effectiveness, this approach suffers from limited generalizability and efficiency due to its reliance on fine-tuned LLMs, context length constraints, and poor scalability. Hence, it is not our focus of the framework.

4 EXPERIMENT

Building upon the LEGO-GraphRAG framework, we develop an implementation that facilitates the seamless integration of modules and methods for constructing GraphRAG instances.[9] To assess the effectiveness of the LEGO-GraphRAG framework and our implementation, we create versions of our framework based on the core ideas of three state-of-the-art instances—RoG [71], KELP [65], and ToG [98]—and align the experimental setups to ensure a fair comparison. As shown in Table 3, the instances implemented within our framework exhibit performance comparable to their original counterparts across two datasets. For KELP, our implementation outperforms the original due to the effective integration of two modules and the optimization of the employed methods.

Table 3: Existing Instances vs. LEGO-GraphRAG Instances

GraphRAG Instances	WebQSP		CWQ	
	Hits@1	Recall	Hits@1	Recall
RoG [71] (RoG planning w/ChatGPT)	81.51	71.60	52.68	48.51
LEGO-RoG (RoG planning w/ChatGPT)	82.79	64.41	56.06	49.76
KELP [65] (one-hop w/gpt-4o-mini)	31.06	-	14.16	-
LEGO-KELP (one-hop w/gpt-4o-mini)	77.36	63.99	48.65	43.88
ToG [98] (w/Llama3-8B)	59.76	43.05	36.97	32.69
LEGO-ToG (w/Llama3-8B)	66.44	44.77	40.26	33.63

As shown in Table 2, the existing instances are far from covering the four groups of GraphRAG instances within our framework. Thus, we construct 15 distinct GraphRAG instances (numbered and labeled in Table 2) by selecting representative methods from subgraph-extraction and path-retrieval modules for a detailed empirical study (Section 4.2). In addition, we conduct separate evaluations of the various methods, strategies, and semantic models utilized in the two modules (Sections 4.3 and 4.4).

4.1 Experiment Settings

Graph and GraphRAG Query Datasets. We leverage Freebase [6], a large-scale, multi-domain knowledge base (e.g., on finance, law, sports) widely used in GraphRAG research [45, 50, 51, 70–72, 77, 98, 108, 121], along with four established query datasets, WebQSP [120], CWQ [100], GrailQA [30], and WebQuestions [16], covering diverse and challenging GraphRAG scenarios. Following standard settings [38, 51, 71, 98], we construct dataset-specific graphs, including all triples reachable within the maximum reasoning hops from query entities (100M nodes, 300M edges) and sample 1,000 test queries per dataset (one-hop: multi-hop is 1:1). All experiments (Sections 4.2–4.4) are conducted based on these datasets. We also include MetaQA [125], a non-Freebase dataset built on Wiki-Movies [78], a single-domain base focused on movies (40K nodes, 130K edges), with 1,000 test queries to assess cross-base and cross-domain transferability of our framework.[10]

Metrics. Our empirical study is based on three key metrics: quality, efficiency, and cost. *a) Quality. a).1 For the subgraph-extraction*, F1 Score serves as the primary evaluation metric, balancing Precision and Recall to reflect overall extraction quality.[11] However, since this module directly influences the retrieval space for downstream modules, we introduce a minimum Recall threshold to ensure sufficient coverage of relevant entities and relations. Relying solely on high F1 Score may favor Precision at the cost of

[7] Alternatively, multiple LLM calls can be used to process all reasoning paths, improving quality at the cost of efficiency. This feature is supported in our implementation (see technical report B.6 and our open-source repository).

[8] See the technical report B.8 for detailed algorithm

[9] Our implementation is built in Python and integrates libraries such as Transformers [114], iGraph [17], PyTorch[2], and vLLM [58] to support flexible configuration of algorithms, models, and reasoning. The semantic models used are publicly available on the Hugging Face [46].

[10] Further details of datasets are in our technical report B.2 and B.3.

[11] Precision = $|\mathbf{C}_q \cap \mathbf{A}_q|/|\mathbf{C}_q|$, Recall = $|\mathbf{C}_q \cap \mathbf{A}_q|/|\mathbf{A}_q|$, where $\mathbf{C}_q$ and $\mathbf{A}_q$ represent the predicted and ground truth sets, respectively. The F1 score is the harmonic mean of Precision and Recall: F1 score = $(2 \cdot \text{Precision} \cdot \text{Recall})/(\text{Precision} + \text{Recall})$

missing important components. In practice, a fixed Recall threshold (e.g., around 60%) is usually sufficient to ensure downstream performance, after which F1 Score can guide subgraph-extraction. We also explore adaptive thresholding based on query complexity, graph density, downstream tasks, and historical logs, balancing coverage and efficiency.[12] *a).2 For the path-retrieval*, the F1 score[11] measures the alignment between the retrieved reasoning paths and the ground truth answers. For sets containing the same number of reasoning paths, a higher F1 score indicates better quality. *a).3 For the GraphRAG instance*, we adopt Hits@1 as the main evaluation metric, following prior work [3, 50, 62, 63, 75, 98]. It measures the proportion of outputs that exactly match the ground truth.[13] We also report two commonly used complementary metrics—F1 score and LLM-based evaluation. Both show similar trends to Hits@1 across different GraphRAG instances and do not affect our experimental conclusions.[14] b) Efficiency. We record the runtime of both modules in the GraphRAG instance per query. c) Cost. We track token cost and peak GPU memory usage of the GraphRAG instance on EEMs and LLMs, reflecting processing load and computational resource demands, respectively.

Settings for Graph Instance Experiments. For GraphRAG instances in this study, we selected methods and semantic models that balance efficiency and quality within their respective categories, based on experiments on subgraph-extraction and path-retrieval modules (Sections 4.3 and 4.4). To ensure fair comparisons, we adjusted parameters within each instance so that all instances ultimately retrieved the same number of reasoning paths.[15] The key experimental settings are as follows, and detailed settings for each instance are provided in supplemental material: *a) EEMs&LLMs.* A sentence-Transformer (ST) [111] and Qwen2-72B [117] are employed as the EEMs and LLMs implementations, respectively. Note that for all experiments in this paper, the batch size of EEMs is set to 64 and the max input token of LLMs is set to 16k. *b) SBE.* The PPR algorithm is applied, with a maximum of nodes $N_{ppr} = 1,000$ retained. *c) SAE.* Edge pruning (EP) is used for EEMs and LLMs based on subgraphs extracted by PPR, with a maximum of edges $N_e = 64$ retained. *d) SBR.* The Dijkstra algorithm is employed for SPR implementation. *e) OSAR.* Select $N_p = 32$ reasoning paths from those retrieved using SPR, leveraging both EEMs and LLMs. *f) ISAR.* The Beam Search algorithm is selected for implementation and combined with both EEMs and LLMs. The beam width is set to $B = 8$, and the maximum path length retained is $L = 4$. *g) Generation.* We utilize various LLMs with differing architectures and scales .[16]

Settings for Module Experiments. To evaluate the quality and efficiency of methods in subgraph-extraction and path-retrieval modules, we implement representative methods from each method category, and conduct independent experiments. The settings are as follows: *a) EEMs&LLMs Selection.* A variety of EEMs, including statistical model, embedding model, and reranker, are employed. Specifically, we adopt BM25 as a representative statistical model, a sentence-Transformer (ST) [111] as a widely used embedding model, BGE-Reranker (BGE) [13] as an effective re-ranker, and Qwen2-72B [117] as the LLM implementation.

b) Settings for the Subgraph-Extraction Module. b).1 SBE. We employ PPR as a structural importance-based method, RW as a random walk-based algorithm, and KSE as a neighborhood-based strategy. *b).2 SAE.* Node Pruning (NP), Edge Pruning (EP), and Triple Pruning (TP) are applied across all selected EEMs and LLMs, based on subgraphs extracted by SBE, with PPR selected due to its superior quality and efficiency in our evaluations.

c) Settings for the Path-Retrieval Module. We extract 3,000 subgraphs from four evaluation datasets using methods implemented in the subgraph-extraction module. From each dataset, 250 test queries (1,000 total) are sampled, and three subgraphs per query are generated using SBE, SAE-EEM, and SAE-LLM, with the most effective implementations selected based on our experiments. These subgraphs are used to evaluate the methods in the path-retrieval module independently. The settings for each method are as: *c).1 SBR.* The Dijkstra algorithm is employed for SPR, while the DFS algorithm is employed for EPR. *c).2 OSAR.* We use all selected EEMs and LLMs to perform semantic evaluation and refinement on the paths obtained through SPR (a superior SBR method based on our evaluation). *c).3 ISAR.* We combine beam search algorithm ($B = 8$ and $L = 4$) with all selected EEMs and LLMs.

4.2 Evaluation of GraphRAG Instances

We first present the results of all GraphRAG instances across various metrics and analyze key observations.[17]

a) Reasoning Performance. As shown in Figure 2, the reasoning performance of different GraphRAG instances across four datasets follows a consistent trend. With the increase in reasoning model size and capabilities (i.e., from 7B LLMs to 72B LLMs), all instances show significant performance improvements. Next, we analyze the instances according to the groups defined in Table 2.

a).1 Structural Methods on Both Modules (Group (I)). Instance No.1 of Group (I) exhibits the worst overall performance. For example, its Hits@1 is only 0.49 (WebQSP with Qwen2-7B), while most other instances are close to or exceed 0.6. This indicates that structure-based methods alone yield baseline solutions, necessitating semantic models for further improvement. Notably, Instance No.1 outperforms certain semantically augmented instances on specific datasets, such as No.6, No.8, and No.14 (CWQ with Llama3.3-70B). It shows that while semantic augmented methods generally perform better, exceptions depend on query complexity, domain, and integration of semantic models, as discussed later.

a).2 Semantic Augmentation on Both Modules (Group (II)). In Group (II), instances using EEMs for subgraph-extraction (e.g., No.2, 3, 6, 7) consistently outperform those using LLMs (e.g., No.4, 5, 8, 9) across GrailQA, and all reasoning models. This indicates that LLMs are unsuitable for subgraph-extraction when both modules are semantically augmented, as they may overly prune graph information, losing critical intermediate information in the reasoning paths, a limitation seen in instances like GCR [72], which compromises stability and practicality.

[12] Detailed strategies are available in our technical report B.5.
[13] Note the gap between the evaluation of path-retrieval and that of the GraphRAG instance, which arises from the ability of LLMs to utilize the reasoning paths.
[14] Detailed results are provided in the technical report B.4.
[15] In this paper, we follow prior work [45, 71, 77] by setting the number of reasoning paths to 32. Note that for instances integrating LLMs in OSAR and ISAR, the number of reasoning paths may be fewer than 32 due to the uncertainty in their outputs.
[16] Glm4-9B [29], Llama3.3-70B [103], Qwen2-7B and Qwen2-72B [117]
[17] We supplement the evaluation with a case study comparing representative reasoning paths and answers (see technical report B.12).

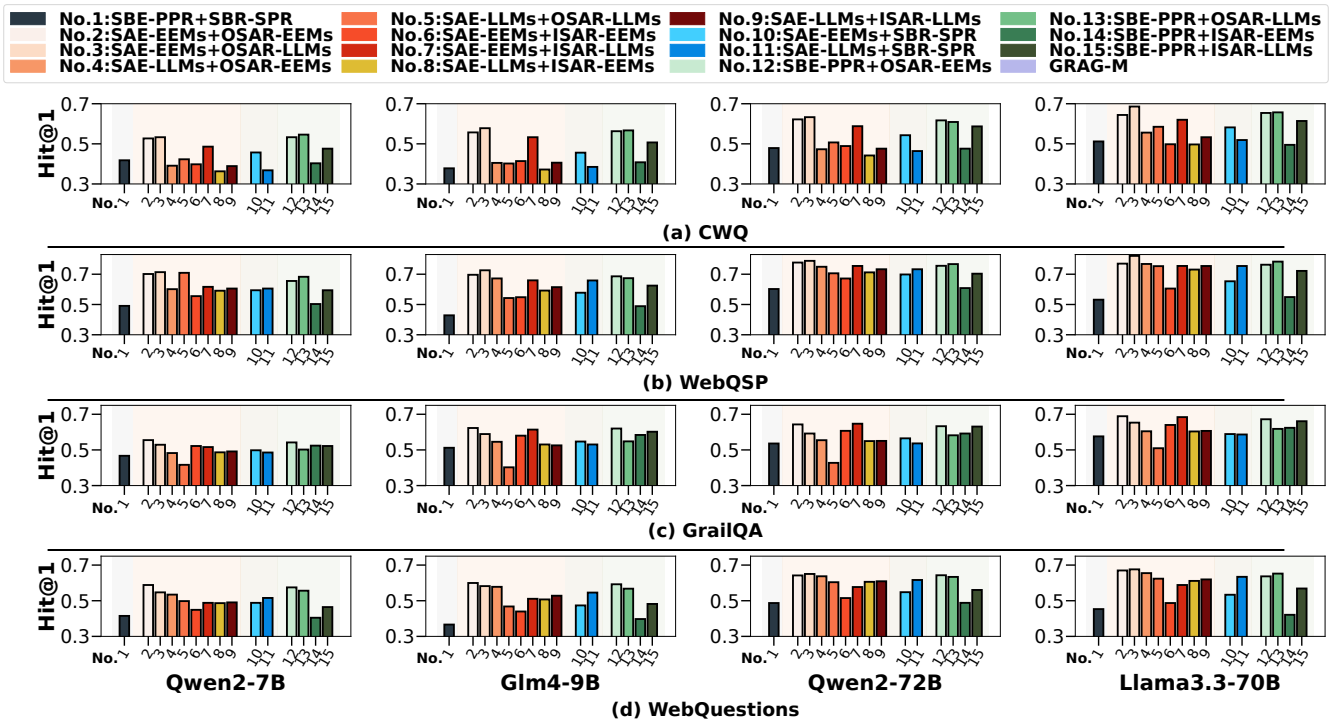

Figure 2: Reasoning Results of LEGO-GraphRAG Instances Across Four Datasets

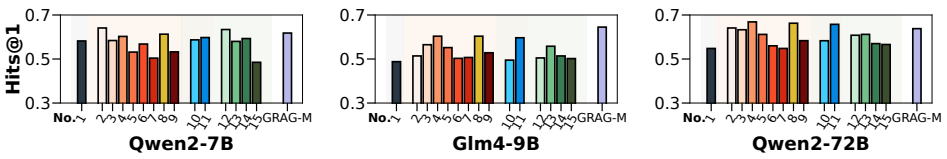

Figure 3: Results of LEGO-GraphRAG Instances and GraphRAG-M Instance on MetaQA Dataset

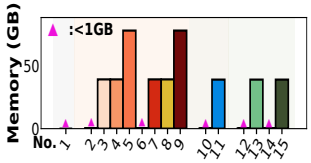

Figure 4: Peak GPU Memory Usage

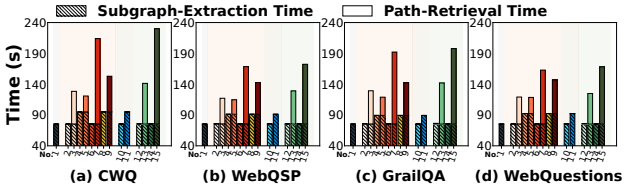

Figure 5: Runtime of SE and PR Modules for GraphRAG Instances

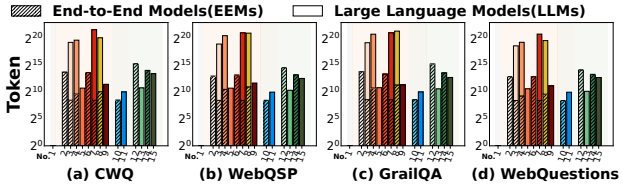

Figure 6: Token Costs for EEMs and LLMs in GraphRAG Instances

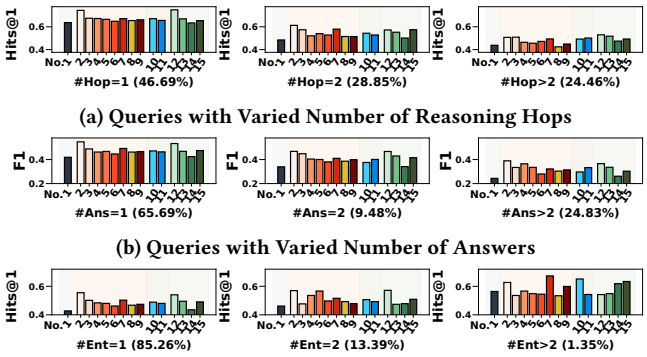

Figure 7: Instance Performance w.r.t. Queries

Also, Instances No.2 and 3 in Group (II) achieve the best overall performance, consistently ranking in the top-3 in all reasoning models (WebQSP). This highlights the effectiveness of combining EEMs for subgraph-extraction with OSAR for path-retrieval, a strategy yet unexplored by existing methods like GSR [45], which pairs EEMs with structure-based path-retrieval. Notably, among Group (II) instances using EEMs for subgraph-extraction (i.e., No.2, 3, 6, 7), Instance No.6 performs the worst. For example, its performance is similar to that of Instance No.1, which uses only structure-based methods (CWQ with all reasoning models). This suggests that ISAR pairs better with LLMs than EEMs, as EEMs' weaker semantics can yield noisy scores on long paths and small score gaps, making pruning unreliable. One way to improve ISAR-EEMs is to enhance the EEMs themselves, for example, through customization or fine-tuning for specific domains or graphs. In addition, we propose two

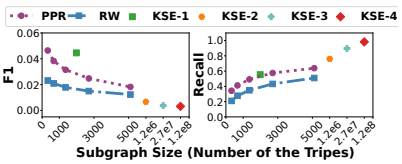

Figure 8: Results w.r.t. Subgraph Size (SBE)

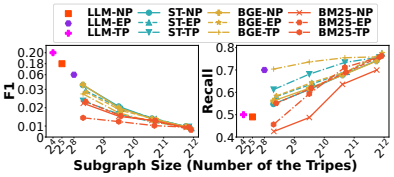

Figure 9: Results w.r.t. Subgraph Size (SAE)

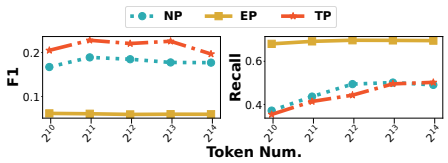

Figure 10: Results w.r.t. Token Cost (SAE-LLMs)

complementary strategies: **triple-level scoring**, which evaluates only the newly added step at each expansion to reduce error accumulation, and **dynamic pruning**, which adjusts the number of retained paths based on score distribution, helping preserve good candidates when scores are close.[18]

a).3 Semantic Augmentation on one Module (Group (III)&(IV)). We observe that the performance of instances in Group IV (i.e., Instances No.12-15) generally surpasses that of Group III (i.e., Instances No.10, 11). This suggests that if only one module is semantically augmented, prioritizing the path-retrieval module over the subgraph-extraction module is more effective. Using structure-based methods for subgraph extraction remains efficient and sufficient, unlike existing instances such as RoG [71] and GCR [72], which use LLMs (Llama-2-7b, Llama-3.1-8b) for subgraph-extraction and encounter performance and efficiency bottlenecks. Additionally, Group (II) instances do not consistently outperform Groups (III) and (IV), implying that semantically augmenting both modules is unnecessary in resource-limited contexts. Moreover, as shown in Figure 3, results on MetaQA align with findings from Freebase-based datasets. The small scale and single-domain nature of the graph lead to uniformly high scores across all instances. Notably, Instance No.14 (ISAR-EEMs) performs comparably to or slightly better than Instance No.15 (ISAR-LLMs), suggesting that EEMs can be a viable and more efficient alternative to LLMs in such settings.

b) Runtime. Figure 5 shows the average runtime of different GraphRAG instances for subgraph-extraction and path-retrieval modules on a single query. A key observation is that subgraph-extraction is the primary bottleneck in GraphRAG runtime of all instances. For example, Instance No.1, using structure-based methods, requires approximately 70 seconds for subgraph-extraction (i.e., PPR algorithm) but less than 0.1 seconds for path-retrieval. Since subgraph-extraction is a crucial but time-intensive step in the GraphRAG workflow, it poses significant challenges for the practical development of GraphRAG systems on large-scale graphs like Freebase. Unfortunately, most existing instances, such as KELP [65], ToG [98], and DoG [74], overlook the efficiency of this process.

Additionally, in the path-retrieval, methods using LLMs (especially under ISAR) exhibit longer runtimes compared to those using EEMs. For example, Instance No.12 (OSAR-EEMs) and No.14 (ISAR-EEMs) complete path-retrieval in under 1.15 seconds and 0.19 seconds, respectively, while Instance No.13 (OSAR-LLMs) and No.15 (ISAR-LLMs) take 66.64 seconds and 122.43 seconds, respectively, on GrailQA. This shows that using the ISAR-LLMs methods in the path-retrieval may result in unacceptably low query efficiency.

c) Cost. Figures 6 and 4 show the average token overhead (EEMs and LLMs) and peak GPU memory usage (average of four datasets) across GraphRAG instances. Instances mixing EEMs and LLMs (e.g., No.3, 4, 7, 8) incur higher token costs than those using the same model type (e.g., No.2, 5, 6, 9), with EEM-only setups being the most economical. GPU usage remains low (<1GB) for EEM-only instances but rises sharply for LLM-based ones, peaking at 80GB when LLMs are used in both modules (e.g., No.5, 9). The results underscore the performance–cost trade-offs of semantic augmentation and the importance of corresponding optimization.

d) Analysis by Query Type. We analyze performance across different query types by varying reasoning hops, number of answers, and number of query entities. As shown in Figure 7a, performance consistently declines with more reasoning hops, especially for structure-based methods (e.g., Instance No.1), which struggle on multi-hop queries. Figure 7b shows a similar drop from single- to multi-answer queries, where semantic methods are more resilient. Figure 7c shows improved performance with more query entities, likely due to increased chances of retrieving relevant paths—even benefiting structure-only instances like No.1.

e) Integrating Microsoft GraphRAG. To evaluate compatibility with Microsoft GraphRAG, we implement an instance (GRAG-M), which follows its workflows: performing community detection on the graph and precomputing textual summaries for each community. On the MetaQA dataset, GRAG-M uses the reasoning paths from Instance 12 (which showed strong performance in our experiments) and provides both the paths and summaries as input to the LLMs. As shown in Figure 3, GRAG-M performs well on Qwen2-7B and Glm4-9B. Gains diminish with Qwen2-72B, where the larger model already captures sufficient context, reducing the benefit of precomputed summaries.

4.3 Evaluation of Subgraph-Extraction Module

Structure-Based Extraction (SBE). Figure 8 shows the variation in F1 score and Recall for the three SBE methods (RW, PPR, and KSE) as the extracted subgraph size changes. The F1 score for all methods decreases as the subgraph size increases, while their Recall consistently improves. Across all subgraph sizes, PPR consistently outperforms RW in both F1 score and Recall, demonstrating the superior effectiveness of PPR over RW. Additionally, RW struggles to achieve a Recall above 60%, rendering it nearly unusable in GraphRAG. Moreover, PPR offers greater flexibility compared to KSE, enabling precise control over subgraph size through hyperparameters. In contrast, KSE relies on hierarchical extraction, which lacks such flexibility.[19] These advantages establish PPR as the current optimal solution for SBE. As Table 4 demonstrates, PPR's computational costs become prohibitive for large graphs.

[18] Detailed discussion and results are available in our technical report B.8.

[19] Many prior studies [45, 50, 71, 72] first apply KSE to extract 2-hop or 4-hop subgraphs, followed by PPR to distill them into smaller, query-relevant subgraphs.

Table 4: PPR Runtime w.r.t. Graph Size

Nodes	Ave. Edges	Ave. Time (s)
100000	431414.5	0.58
1000000	5567538.1	1.37
10000000	48865879.2	11.5
100176641	298458255	75.29

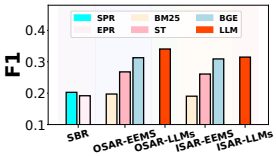

Figure 11: F1 Score of Methods in Path-Retrieval Module

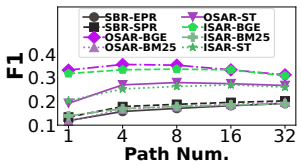

Figure 12: F1 Score vs. # of Paths (for Methods of Path-Retrieval)

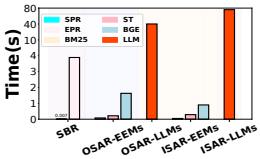

Figure 13: Runtime of Methods in Path-Retrieval (WebQSP)

Semantic-Augmented Extraction (SAE). Figure 9 shows the Recall and F1 score of different semantic model combinations with three subgraph pruning strategies under the SAE methods, as the subgraph size varies. Among the three semantic models, the BGE and ST significantly surpass BM25 across all pruning strategies and subgraph scales. BGE is slightly better in quality, while ST is even better in efficiency, as discussed later. Considering the trade-off between quality and efficiency, we propose to use ST as the representative EEMs. For the three pruning strategies, TP exhibits the best quality at smaller subgraph scales, while NP performs the worst. However, as subgraph size increases, the F1 score and Recall of EP, TP, and NP converge. Ultimately, EP slightly outperforms TP, while TP marginally surpasses NP. Thus, TP is recommended when the size of the extracted subgraph is small; otherwise, EP.

Due to the inherent uncertainty in LLM outputs, SAE-LLMs methods often struggle to control the size of subgraphs extracted, typically producing smaller subgraphs that can reduce path-retrieval quality (see **Finding 2**). Figure 9 shows the F1 score and Recall for subgraphs generated using SAE-LLMs under three pruning strategies, plotted against the average subgraph size. Among the three pruning strategies, EP outperforms NP and TP for SAE-LLMs. Figure 10 further shows the trends in F1 score and Recall as the number of tokens input into the LLMs varies for NP, TP and EP. Recall for NP consistently surpasses that of both TP and EP, maintaining a gap of around 0.2. Although EP underperforms in terms of F1 score due to the larger subgraphs it extracts, the relatively low Recall of NP and EP may diminish their effectiveness in path-retrieval. As the number of tokens increases, Recall for NP and TP initially rises, then declines, while EP remains nearly constant, indicating that EP achieves more effective subgraph extraction at a lower token cost for the SAE-LLMs methods. The efficiency of SAE methods is related to the semantic models used. Runtime is influenced by factors such as model parameters, algorithmic complexity, and data volume. In general, the runtime for processing a graph component follows this order: LLMs > BGE > ST > BM25, as shown in Figure 13.

Table 5: Results of Hybrid ISAR Combining EEMs and LLMs

Method	CWQ	WebQSP	GrailQA	WebQuestion
SBE+ISAR-EEMs	0.183	0.157	0.395	0.16
SBE+ISAR-EEMs (LLMs-R)	**0.271**	**0.219**	**0.379**	**0.175**
SBE+ISAR-LLMs	0.249	0.264	0.371	0.249
SBE+ISAR-LLMs (EEMs-PF)	0.271	0.316	0.396	0.245
SBE+ISAR-LLMs (EEMs-S)	**0.310**	**0.373**	**0.435**	**0.276**

4.4 Evaluation of Path-Retrieval Module

Figure 11 shows the F1 score of different methods in path-retrieval module. It shows that OSAR outperforms ISAR in all semantic models. SPR method slightly outperforms EPR and I/OSAR with BM25, but underperforms all I/OSAR solutions with ST, BGE or LLM, which suggests that the use of poorer semantic models to aid enhanced retrieval may lead to a loss of quality. We further investigate the variation in the F1 score of the path-retrieval module when the number of reasoning paths changes, as shown in Figure 12. Since the number of paths output by LLM is inherently uncertain, methods using LLM are excluded. It shows that the F1 score of SBR and I/OSAR with BM25 gradually increases with the number of paths, yet their F1 score remains low-level, consistently trailing I/OSAR with BGE and ST by approximately 0.1. The gap widens to 0.2 or more when fewer paths are retrieved. In contrast, the performance of I/OSAR with ST and BGE initially improves and then declines as the number of retained paths increases, with OSAR generally outperforming ISAR. Note that F1 scores may be higher for smaller sets of reasoning paths, as they tend to prioritize precision. A slight decrease in F1 for larger reasoning path sets is not indicative of lower quality but rather reflects the broader scope of the retrieved paths.

Figure 13 shows the runtime of different methods in the path-retrieval module (WebQSP). Among the SBR solutions, EPR has the longest runtime due to the large number of paths it retrieves, while SPR reduces computation time by retrieving only a single path for each node. The EPR-based OSAR method becomes impractical because its excessive data volume results in prohibitive runtime overhead when used with semantic models. As a result, only SPR is employed in this context. Among the three semantic models, both BGE and ST demonstrate significant quality improvements over BM25 combined with I/OSAR. BGE achieves marginally higher quality, whereas ST exhibits superior efficiency.

Table 5 reports the results of hybrid ISAR methods combining EEMs and LLMs. All three methods outperform single-model baselines on most datasets, confirming the benefit of combining model strengths. Notably, the EEMs-S strategy yields the best results, suggesting that combining EEMs and LLMs helps reduce the risk of missing important paths during expansion.

5 FINDINGS AND DISCUSSION

We summarize key findings from our empirical study and discuss promising ways for addressing GraphRAG's two key bottlenecks: subgraph-extraction efficiency and LLM generation quality.

5.1 Summary of Key Findings

Finding 1: GraphRAG Trade-off Structure. GraphRAG entails multi-aspect trade-offs among quality (Hits@1), efficiency (runtime), and cost (token and GPU usage). This trade-off spans two modules (subgraph-extraction vs. path-retrieval), two method types (structure-based vs. semantic-augmented), and two types of semantic models (EEMs vs. LLMs).

Finding 2: Module-level Trade-offs. Subgraph-extraction is the main efficiency bottleneck for large-scale graphs. Overuse of semantic augmentation (e.g., using LLMs) in subgraph-extraction can compromise downstream path-retrieval quality. Targeting semantic augmentation in the path-retrieval module is a more cost-effective strategy to maintain high quality.

Finding 3: Method-level Trade-offs. Structure-based methods offer higher efficiency and lower cost, but semantic-augmented methods are essential for quality, especially for complex (two- or multi-hop) queries. One-way semantic methods (OSAR) balance quality and efficiency better than interactive methods (ISAR).

Finding 4: Model-level Trade-offs. EEMs are more cost effective in token and GPU usage, while LLMs generally provide better quality, particularly for interactive methods (ISAR). However, LLMs can exhibit unstable generation in some cases, affecting consistency. A balanced approach includes applying LLM-based augmentation in path-retrieval or EEM-based augmentation in both modules.

Finding 5: A Promising yet Underexplored Strategy. Based on our analysis, the effective approach for GraphRAG would be the one that combines structure based methods for subgraph-extraction with OSAR for path-retrieval. Despite its potential, this strategy remains underexplored in existing works. Also, it opens up opportunities for further optimizations for handling multi-hop and multi-answer queries, as well as improving overall efficiency.

5.2 Efficiency of the Subgraph-Extraction

We explore promising directions for faster subgraph-extraction, supported by empirical analysis, including limitations, trade-offs, and research opportunities.[20] Below, we briefly outline each direction.

Computational acceleration yields significant efficiency improvement for SBE methods (e.g., PPR) through approximate and distributed algorithms (Table 6). Approximate methods may lower Recall; distributed solutions depend on system setup.

Precomputation-based acceleration pre-generates subgraphs (e.g., via clustering or community detection) to scale down the search space for the extraction process in order to reduce runtime overhead. As shown in Table 7, our implemented prototype demonstrates the potential of this direction, but it also incurs preprocessing overhead and may limit the ability to access relevant information spread across different subgraphs.

Vector database-based acceleration encodes graph components (e.g., nodes or triples) into embedding vectors and stores them in a vector database, thus efficiently identifying query-relevant components based on semantic similarity, followed by subgraph construction centered on them. Table 7 shows our prototype achieves notable speedups with high Recall. However, the retrieved components often form multiple weakly connected subgraphs, which may hinder multi-hop reasoning.

5.3 Generation Quality of LLMs

For the issue of LLMs generating fewer or lower-quality components, we take the path-retrieval module as a example and explore two effective strategies: **Multi-round LLMs (M-LLMs)** conduct iterative refinement via repeated LLM calls. **EEMs supplementation (EEMs-S)** compensates for insufficient outputs by incorporating top-ranked candidate paths from EEMs. Table 8 shows both strategies improve performance across most datasets (applied in the SBE+OSAR-LLMs instance), though with added computational overhead of multiple model calls and EEM-based ranking steps.[21]

[20] See technical report B.7 for details.

Table 6: Computational acceleration of PPR

Approximate PPR (Freebase)			Distributed PPR	
Method	**Recall**	**Ave. Time (s)**	**Method**	**Speedup**
RBS [107]	0.31	**0.51**	HGPA [31]	3.4–4.1×
Fora [110]	0.18	7.61	PAFO [109]	58.7×
TopPPR [113]	0.44	42.08	Delta-Push [39]	**123–162**×
Standard PPR	**0.96**	75.29	**Standard PPR**	1× (baseline)

Table 7: Performance and Cost Analysis of the SE Acceleration Methods on Freebase (About 100M Nodes, 300M Edges)

Method	Pre-time	Online-time	Recall	F1	WCC
Precomputation	19373s	19.02s	0.52	0.0021	**1**
Vector Database	14570s	**0.85s**	0.65	0.0023	12.86
Standard PPR	**0s**	75.29s	**0.96**	**0.0038**	**1**

Table 8: Performance of strategies designed to mitigate the limitations of LLM-generated outputs

Method	CWQ	WebQSP	GrailQA	WebQuestions
SBE+OSAR-LLMs	0.304	0.401	0.401	0.328
SBE+OSAR-LLMs (M-LLMs)	0.349	**0.438**	0.381	0.348
SBE+OSAR-LLMs (EEMs-S)	**0.381**	0.427	**0.476**	**0.353**

6 CONCLUSION

In this paper, we introduce LEGO-GraphRAG, a unified framework for the modular analysis and design of GraphRAG instances. By dividing the GraphRAG retrieval process into distinct modules and identifying corresponding design solutions, LEGO-GraphRAG provides a viable approach for building advanced GraphRAG systems. Building upon the LEGO-GraphRAG framework, we conduct extensive empirical studies on large-scale real-world graphs and diverse GraphRAG query sets, leading to key findings: **1)** GraphRAG must balance quality, efficiency, and cost across three aspects: modules, method types, and semantic models. **2)** Extracting query-relevant subgraphs is the primary efficiency bottleneck for GraphRAG on large-scale graphs and remains underexplored. **3)** Graph structural information enables efficient solutions for GraphRAG, while semantic information is crucial for improving the quality of complex queries. The optimal solution should integrate structural information to identify query-relevant subgraphs and leverage semantic information to retrieve reasoning paths.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 62472400, Grant 62072428, Grant 62271465, in part by the Suzhou Basic Research Program under Grant SYG202338, and in part by the HK RGC grants 12202024, R1015-23, and C1043-24GF. Yukun Cao and Zengyi Gao contributed equally to this work. Xike Xie is the corresponding author.

[21] More details and analysis are in technical report B.9.

REFERENCES

[1] Ibrahim Abdelaziz, Razen Harbi, Zuhair Khayyat, and Panos Kalnis. 2017. A survey and experimental comparison of distributed SPARQL engines for very large RDF data. *Proc. VLDB Endow.* 10, 13 (Sept. 2017), 2049–2060.

[2] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, and et al. Chourdia. 2024. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS 24)*. ACM.

[3] Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. arXiv:2306.04136 [cs.CL]

[4] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing(EMNLP 2013)*, David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (Eds.). Association for Computational Linguistics, Seattle, Washington, USA, 1533–1544.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. 3, null (March 2003), 993–1022.

[6] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data(SIGMOD 2008)*. 1247–1250.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems(NeurIPS 2020)* 33 (2020), 1877–1901.

[8] Yukun Cao, Shuo Han, Zengyi Gao, Zezhong Ding, Xike Xie, and S. Kevin Zhou. 2024. GraphInsight: Unlocking Insights in Large Language Models for Graph Structure Understanding. arXiv:2409.03258 [cs.CL]

[9] Dhivya Chandrasekaran and Vijay Mago. 2022. Evolution of Semantic Similarity - A Survey. *ACM Comput. Surv.* 54, 2 (2022), 41:1–41:37.

[10] Harrison Chase. 2022. *LangChain.* https://github.com/langchain-ai/langchain

[11] Hanzhu Chen, Xu Shen, Qitan Lv, Jie Wang, Xiaoqi Ni, and Jieping Ye. 2024. SAC-KG: Exploiting Large Language Models as Skilled Automatic Constructors for Domain Knowledge Graph. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics(ACL 2024) (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 4345–4360.

[12] Hao Mark Chen, Wayne Luk, Ka Fai Cedric Yiu, Rui Li, Konstantin Mishchenko, Stylianos I Venieris, and Hongxiang Fan. 2024. Hardware-aware parallel prompt decoding for memory-efficient acceleration of llm inference. *arXiv preprint arXiv:2405.18628* (2024).

[13] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2402.03216 [cs.CL]

[14] Yongrui Chen, Huiying Li, Guilin Qi, Tianxing Wu, and Tenggou Wang. 2022. Outlining and Filling: Hierarchical Query Graph Generation for Answering Complex Questions over Knowledge Graphs. arXiv:2111.00732 [cs.AI]

[15] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307* (2023).

[16] Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. UHop: An Unrestricted-Hop Relation Extraction Framework for Knowledge-Based Question Answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 345–356.

[17] Gábor Csárdi and Tamás Nepusz. 2006. The igraph software package for complex network research. *InterJournal, Complex Systems* (2006), 1695.

[18] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990).

[19] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms, 2023. 2 (2023).

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[21] Jialin Dong, Bahare Fatemi, Bryan Perozzi, Lin F. Yang, and Anton Tsitsulin. 2024. Don't Forget to Connect! Improving RAG with Graph-based Reranking. arXiv:2405.18414 [cs.CL]

[22] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130 [cs]

[23] Wenfei Fan. 2022. Big graphs: challenges and opportunities. *Proc. VLDB Endow.* 15, 12 (Aug. 2022), 3782–3797.

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

[25] Yasuhiro Fujiwara, Makoto Nakatsuji, Makoto Onizuka, and Masaru Kitsuregawa. 2012. Fast and exact top-k search for random walk with restart. 5, 5 (Jan. 2012), 442–453.

[26] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise Zero-Shot Dense Retrieval without Relevance Labels. arXiv:2212.10496 [cs]

[27] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL]

[28] Yunfan Gao, Yun Xiong, Meng Wang, and Haofen Wang. 2024. Modular rag: Transforming rag systems into lego-like reconfigurable frameworks. *arXiv preprint arXiv:2407.21059* (2024).

[29] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, and Jiadai Sun et al. 2024. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. arXiv:2406.12793

[30] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW 2021)*. Association for Computing Machinery, New York, NY, USA, 3477–3488.

[31] Tao Guo, Xin Cao, Gao Cong, Jiaheng Lu, and Xuemin Lin. 2017. Distributed Algorithms on Exact Personalized PageRank. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD 2017)*. Association for Computing Machinery, New York, NY, USA, 479–494.

[32] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. arXiv:2410.05779 [cs.IR]

[33] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. arXiv:2405.14831 [cs.CL]

[34] Haoyu Han, Harry Shomer, Yu Wang, Yongjia Lei, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, and Jiliang Tang. 2025. RAG vs. GraphRAG: A Systematic Evaluation and Key Insights. arXiv:2502.11371 [cs.IR]

[35] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. 2025. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309* (2025).

[36] Mohamed S. Hassan, Walid G. Aref, and Ahmed M. Aly. 2016. Graph Indexing for Shortest-Path Finding over Dynamic Sub-Graphs. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD 2016)*. Association for Computing Machinery, New York, NY, USA, 1183–1197.

[37] Taher H. Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the 11th International Conference on World Wide Web* (Honolulu, Hawaii, USA) *(WWW 2002)*. Association for Computing Machinery, New York, NY, USA, 517–526.

[38] Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving Multi-hop Knowledge Base Question Answering by Learning Intermediate Supervision Signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (Virtual Event, Israel) *(WSDM 2021)*. Association for Computing Machinery, New York, NY, USA, 553–561.

[39] Guanhao Hou, Qintian Guo, Fangyuan Zhang, Sibo Wang, and Zhewei Wei. 2023. Personalized PageRank on evolving graphs with an incremental index-update scheme. *Proceedings of the ACM on Management of Data(SIGMOD 2023)* 1, 1 (2023), 1–26.

[40] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[41] Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2024. A Survey of Knowledge Enhanced Pre-Trained Language Models. *IEEE Transactions on Knowledge and Data Engineering* 36, 4 (2024), 1413–1430.

[42] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. GRAG: Graph Retrieval-Augmented Generation. arXiv:2405.16506 [cs]

[43] Jiewen Huang, Daniel J. Abadi, and Kun Ren. 2011. Scalable SPARQL querying of large RDF graphs. *Proc. VLDB Endow.* 4, 11 (Aug. 2011), 1123–1134.

[44] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. arXiv:2311.05232 [cs.CL]

[45] Wenyu Huang, Guancheng Zhou, Hongru Wang, Pavlos Vougiouklis, Mirella Lapata, and Jeff Z. Pan. 2024. Less is More: Making Smaller Language Models Competent Subgraph Retrievers for Multi-hop KGQA. arXiv:2410.06121 [cs.CL]

[46] Hugging Face. 2025. Hugging Face: The AI community building the future. Accessed: 2025-01-01.

[47] Erik Johannes Husom, Arda Goknil, Merve Astekin, Lwin Khin Shar, Andre Kåsen, Sagar Sen, Benedikt Andreas Mithassel, and Ahmet Soylu. 2025. Sustainable LLM Inference for Edge AI: Evaluating Quantized LLMs for Energy Efficiency, Output Accuracy, and Inference Latency. *arXiv preprint arXiv:2504.03360* (2025).

[48] Paul Jaccard. 1912. THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE. *New Phytologist* 11, 2 (1912), 37–50.

[49] Patrick Jaillet, Jiashuo Jiang, Chara Podimata, and Zijie Zhou. 2025. Online Scheduling for LLM Inference with KV Cache Constraints. *arXiv preprint arXiv:2502.07115* (2025).

[50] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing(EMNLP 2023)*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 9237–9251.

[51] Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2022. UniKGQA: Unified Retrieval and Reasoning for Solving Multi-hop Question Answering Over Knowledge Graph. In *The Eleventh International Conference on Learning Representations*.

[52] Sylvio Barbon Junior, Paolo Ceravolo, Sven Groppe, Mustafa Jarrar, Samira Maghool, Florence Sèdes, Soror Sahri, and Maurice van Keulen. 2024. Are Large Language Models the New Interface for Data Pipelines?. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments(BiDEDE 2024), Santiago, Chile, June 9-15, 2024*, Philippe Cudré-Mauroux, Andrea Kö, and Robert Wrembel (Eds.). ACM, 6:1–6:6.

[53] Arijit Khan, Sourav S. Bhowmick, and Francesco Bonchi. 2017. Summarizing static and dynamic big graphs. *Proc. VLDB Endow.* 10, 12 (Aug. 2017), 1981–1984.

[54] Arijit Khan and Sameh Elnikety. 2014. Systems for big-graphs. *Proc. VLDB Endow.* 7, 13 (Aug. 2014), 1709–1710.

[55] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. arXiv:2004.12832 [cs]

[56] Hanieh Khorashadizadeh, Fatima Zahra Amara, Morteza Ezzabady, Frédéric Ieng, Sanju Tiwari, Nandana Mihindukulasooriya, Jinghua Groppe, Soror Sahri, Farah Benamara, and Sven Groppe. 2024. Research Trends for the Interplay between Large Language Models and Knowledge Graphs. arXiv:2406.08223 [cs.AI]

[57] Richard E. Korf. 1985. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artif. Intell.* 27 (1985), 97–109.

[58] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles(SOSP 2023)*.

[59] Yunshi Lan and Jing Jiang. 2020. Query Graph Generation for Answering Multi-hop Complex Questions from Knowledge Bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics(ACL 2020)*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 969–974.

[60] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs.CL]

[61] Mufei Li, Siqi Miao, and Pan Li. 2024. Simple Is Effective: The Roles of Graphs and Large Language Models in Knowledge-Graph-Based Retrieval-Augmented Generation. arXiv:2410.20724

[62] Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot In-context Learning on Knowledge Base Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics(ACL 2023) (Volume 1: Long Papers), Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 6966–6980.

[63] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024. Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources. arXiv:2305.13269 [cs]

[64] Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, Huaidong Xiong, Lin Yuan, Jun Xu, Zaoyang Wang, Zhiqiang Zhang, Wen Zhang, Huajun Chen, Wenguang Chen, and Jun Zhou. 2024. KAG: Boosting LLMs in Professional Domains via Knowledge Augmented Generation. arXiv:2409.13731 [cs.CL]

[65] Haochen Liu, Song Wang, Yaochen Zhu, Yushun Dong, and Jundong Li. 2024. Knowledge Graph-Enhanced Large Language Models via Path Selection. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*. 6311–6321.

[66] Jerry Liu. 2022. *LlamaIndex*. https://github.com/jerryjliu/llama_index

[67] Shuhao Liu, Yang Liu, and Wenfei Fan. 2024. PrismX: A Single-Machine System for Querying Big Graphs. *Proc. VLDB Endow.* 17, 12 (Nov. 2024), 4485–4488.

[68] Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Bo Li, Xuming Hu, and Xiaowen Chu. 2025. ChunkKV: Semantic-Preserving KV Cache Compression for Efficient Long-Context LLM Inference. *arXiv preprint arXiv:2502.00299* (2025).

[69] Peter A. Lofgren, Siddhartha Banerjee, Ashish Goel, and C. Seshadhri. 2014. FAST-PPR: scaling personalized pagerank estimation for large graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD 2014)*. Association for Computing Machinery, New York, NY, USA, 1436–1445.

[70] Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Luu Anh Tuan. 2024. ChatKBQA: A Generate-then-Retrieve Framework for Knowledge Base Question Answering with Fine-tuned Large Language Models. arXiv:2310.08975 [cs]

[71] Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *The Twelfth International Conference on Learning Representations*.

[72] Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. 2024. Graph-constrained Reasoning: Faithful Reasoning on Knowledge Graphs with Large Language Models. arXiv:2410.13080 [cs.CL]

[73] Yuanhua Lv and ChengXiang Zhai. 2011. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (Glasgow, Scotland, UK) *(CIKM 2011)*. Association for Computing Machinery, New York, NY, USA, 7–16.

[74] Jie Ma, Zhitao Gao, Qi Chai, Wangchun Sun, Pinghui Wang, Hongbin Pei, Jing Tao, Lingyun Song, Jun Liu, Chen Zhang, and Lizhen Cui. 2024. Debate on Graph: a Flexible and Reliable Reasoning Framework for Large Language Models. arXiv:2409.03155 [cs.CL]

[75] Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, and Jian Guo. 2024. Think-on-Graph 2.0: Deep and Interpretable Large Language Model Reasoning with Knowledge Graph-guided Retrieval. arXiv:2407.10805 [cs]

[76] Yansheng Mao, Yufei Xu, Jiaqi Li, Fanxu Meng, Haotong Yang, Zilong Zheng, Xiyuan Wang, and Muhan Zhang. 2025. LIFT: Improving Long Context Understanding of Large Language Models through Long Input Fine-Tuning. *arXiv preprint arXiv:2502.14644* (2025).

[77] Costas Mavromatis and George Karypis. 2024. GNN-RAG: Graph Neural Retrieval for Large Language Model Reasoning. arXiv:2405.20139 [cs]

[78] Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing(EMNLP 2016), Austin, Texas, USA, November 1-4, 2016*. 1400–1409.

[79] Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop Reading Comprehension through Question Decomposition and Rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics(ACL 2019)*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 6097–6109.

[80] OpenAI. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[81] PENG SI OW and THOMAS E. MORTON. 1988. Filtered beam search in scheduling. *International Journal of Production Research* 26, 1 (1988), 35–62.

[82] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

[83] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. 2004. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Seattle, WA, USA) *(KDD 2004)*. Association for Computing Machinery, New York, NY, USA, 653–658.

[84] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36, 7 (2024), 3580–3599.

[85] Serafeim Papadias, Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2022. Space-efficient random walks on streaming graphs. *Proc. VLDB Endow.* 16, 2 (Oct. 2022), 356–368.

[86] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921* (2024).

[87] Rayleigh. 1905. The Problem of the Random Walk. 72, 1866 (aug 1905), 318.

[88] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing(EMNLP 2019)*. Association for Computational Linguistics.

[89] Nils Reimers and Iryna Gurevych. 2020. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing(EMNLP 2020)*.

Association for Computational Linguistics.

[90] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (April 2009), 333–389.

[91] Stephen E. Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *J. Documentation* 60 (2004), 503–520.

[92] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving Multi-hop Question Answering over Knowledge Graphs Using Knowledge Base Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics(ACL 2020)*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 4498–4507.

[93] Kevin Scott. 2024. Behind the Tech.

[94] Yingxia Shao, Shiyue Huang, Xupeng Miao, Bin Cui, and Lei Chen. 2020. Memory-Aware Framework for Efficient Second-Order Random Walk on Large Graphs. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD 2020)*. Association for Computing Machinery, New York, NY, USA, 1797–1812.

[95] Jianbing Shen, Yunfan Du, Wenguan Wang, and Xuelong Li. 2014. Lazy Random Walks for Superpixel Segmentation. *IEEE Transactions on Image Processing* 23, 4 (2014), 1451–1462.

[96] Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. 2024. A contrastive framework for neural text generation. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NeurIPS 2022)*. Curran Associates Inc., Red Hook, NY, USA, Article 1566, 14 pages.

[97] Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*. 2380–2390.

[98] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. In *The Twelfth International Conference on Learning Representations*.

[99] Shixuan Sun, Yuhang Chen, Shengliang Lu, Bingsheng He, and Yuchen Li. 2021. ThunderRW: an in-memory graph random walk engine. *Proc. VLDB Endow.* 14, 11 (July 2021), 1992–2005.

[100] Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics(ACL 2018): Human Language Technologies, Volume 1 (Long Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 641–651.

[101] Yixuan Tang and Yi Yang. 2024. MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries. arXiv:2401.15391 [cs.CL]

[102] Llama team. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[103] Llama team. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI]

[104] Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics(ACL 2021): Human Language Technologies*. Association for Computational Linguistics, Online, 296–310.

[105] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[106] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models. arXiv:1610.02424 [cs.AI]

[107] Hanzhi Wang, Zhewei Wei, Junhao Gan, Sibo Wang, and Zengfeng Huang. 2020. Personalized pagerank to a target node, revisited. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining(KDD 2020)*. 657–667.

[108] Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-Driven CoT: Exploring Faithful Reasoning in LLMs for Knowledge-intensive Question Answering. arXiv:2308.13259 [cs]

[109] Runhui Wang, Sibo Wang, and Xiaofang Zhou. 2019. Parallelizing approximate single-source personalized pagerank queries on shared memory. *The VLDB Journal* 28, 6 (2019), 923–940.

[110] Sibo Wang, Renchi Yang, Xiaokui Xiao, Zhewei Wei, and Yin Yang. 2017. FORA: Simple and Effective Approximate Single-Source Personalized PageRank. In *SIGKDD 2017*. 505–514.

[111] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *Advances in Neural Information Processing Systems(NeurIPS 2020)*, Vol. 33. Curran Associates, Inc., 5776–5788.

[112] Ye Wang, Qing Wang, Henning Koehler, and Yu Lin. 2021. Query-by-Sketch: Scaling Shortest Path Graph Queries on Very Large Networks. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD 2021)*. Association for Computing Machinery, New York, NY, USA, 1946–1958.

[113] Zhewei Wei, Xiaodong He, Xiaokui Xiao, Sibo Wang, Shuo Shang, and Ji-Rong Wen. 2018. Topppr: top-k personalized pagerank queries with precision guarantees on large graphs. In *Proceedings of the 2018 International Conference on Management of Data(SIGMOD 2018)*. ACM, 441–456.

[114] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing(EMNLP 2020): System Demonstrations*. Association for Computational Linguistics, Online, 38–45.

[115] Shangyu Wu, Hongchao Du, Ying Xiong, Shuai Chen, Tei-wei Kuo, Nan Guan, and Chun Jason Xue. 2025. EvoP: Robust LLM Inference via Evolutionary Pruning. *arXiv preprint arXiv:2502.14910* (2025).

[116] Yubao Wu, Ruoming Jin, and Xiang Zhang. 2014. Fast and unified local search for random walk based k-nearest-neighbor query in large graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (Snowbird, Utah, USA) *(SIGMOD 2014)*. Association for Computing Machinery, New York, NY, USA, 1139–1150.

[117] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, and et al. Chengyuan Li. 2024. Qwen2 Technical Report. *arXiv preprint arXiv:2407.10671* (2024).

[118] Mingji Yang, Hanzhi Wang, Zhewei Wei, Sibo Wang, and Ji-Rong Wen. 2024. Efficient Algorithms for Personalized PageRank Computation: A Survey. *IEEE Trans. on Knowl. and Data Eng.* 36, 9 (March 2024), 4582–4602.

[119] Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic Parsing for Single-Relation Question Answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics(ACL 2014) (Volume 2: Short Papers)*, Kristina Toutanova and Hua Wu (Eds.). Association for Computational Linguistics, Baltimore, Maryland, 643–648.

[120] Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics(ACL 2016) (Volume 2: Short Papers)*, Katrin Erk and Noah A. Smith (Eds.). Association for Computational Linguistics, Berlin, Germany, 201–206.

[121] Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint Decoding of Answers and Logical Forms for Question Answering over Knowledge Bases. arXiv:2210.00063 [cs]

[122] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than Retrieve: Large Language Models are Strong Context Generators. In *The Eleventh International Conference on Learning Representations*.

[123] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. 2025. A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models. arXiv:2501.13958 [cs.CL]

[124] Taolin Zhang, Dongyang Li, Qizhou Chen, Chengyu Wang, Longtao Huang, Hui Xue, Xiaofeng He, and Jun Huang. 2024. R4: Reinforced Retriever-Reorder-Responder for Retrieval-Augmented Large Language Models. arXiv:2405.02659 [cs.CL]

[125] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational Reasoning for Question Answering with Knowledge Graph. In *AAAI*.

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

[127] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223* (2023).

[128] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models. arXiv:2310.06117 [cs]