International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Conceptual Graph RAG Models for Complex Semantic Query Processing

Gartman Ievgen

CEO and founder, Bridge.Digital, Austin, TX, USA
Email: *ievgen.g[at]bridge.digital*

Abstract: ***This article introduces a new class of models for handling complex semantic queries-Conceptual Graph RAG Models-which combine the representational strengths of Conceptual Graphs (CG) with the dynamic retrieval capabilities of the Retrieval-Augmented Generation (RAG) paradigm. The paper examines two recent architectures, Graph RAG and G-Retriever, focusing on the extraction of relevant subgraphs, their optimal construction using Prize-Collecting Steiner Tree algorithms, and integration with large language models (LLMs) via soft prompt tuning. Methods such as Prompt Tuning and LoRA are discussed for enhancing efficiency while reducing the number of trainable parameters. A comprehensive review of evaluation metrics is presented, including precision/recall in retrieval, BERTScore, Mean Reciprocal Rank (MRR), Hop-Accuracy, and hallucination detection. Benchmark datasets such as PATQA, MINTQA, and WebQSP are also analyzed. The results show that the proposed approaches deliver high answer accuracy, improved interpretability, and greater robustness against misinformation-particularly when scaled to large knowledge graphs. The insights offered in this work will be of particular interest to researchers focused on ontology formalization and knowledge representation, especially those working at the intersection of symbolic reasoning and neural retrieval-generation systems. The topic also holds practical value for system architects and engineers of enterprise semantic platforms aiming to optimize and scale complex semantic queries by integrating graph-based knowledge representations with LLMs in intelligent data processing pipelines.***

Keywords: conceptual graphs, Retrieval-Augmented Generation, Prize-Collecting Steiner Tree, soft prompt tuning, multi-hop QA, LLM+KG, explainable AI

1. Introduction

Large Language Models (LLMs) have demonstrated strong capabilities in natural language generation and understanding. However, when applied to complex question answering (QA) tasks, they face notable limitations, including insufficient reasoning capabilities, outdated or absent domain knowledge, limited context length, and a tendency toward factual hallucination [1]. Conceptual Graphs (CGs) are a formalism for graph-based knowledge modeling, in which nodes represent concepts and edges represent relationships between them. Their explicit semantics and support for multi-hop reasoning make them particularly effective for complex logic-based tasks [2]. The Retrieval-Augmented Generation (RAG) paradigm combines external knowledge retrieval with answer generation via LLMs, helping reduce hallucinations and incorrect responses [3].

In recent years, traditional RAG methods have evolved through the integration of LLMs with structured knowledge representations, such as knowledge graphs (KGs). Pan et al. [1] propose a unified roadmap for merging the strengths of LLMs and KGs, delineating the stages of subgraph retrieval, transformation into LLM-consumable formats, and the reintegration of generated outputs into the graph structure. Gao et al. [7] offer a systematization of RAG architectures, knowledge indexing strategies, and dynamic graph handling methods, while also pointing out the lack of standardized benchmarks for evaluating the quality of generation in semantically complex queries. Perozzi et al. [8] introduce a method for encoding structured data into formats directly consumable by LLMs—preserving graph topology via specialized encoders that translate nodes and edges into semantic role-based textual templates.

Regarding empirical RAG models, Lewis et al. [3] formalized the RAG architecture for knowledge-intensive QA tasks, combining retrieval and generation into a unified pipeline that improves accuracy on questions requiring external knowledge. He et al. [2] proposed the G-Retriever model, designed to interpret textual graph descriptions through a two-stage process: initial subgraph retrieval followed by semantic node representation refinement based on the query. This approach achieved state-of-the-art results in graph-based QA tasks. Sen et al. [10] addressed complex chain-of-thought questions by introducing KG-augmented LMs that leverage graph-based ontological knowledge through hybrid attention and planning mechanisms for long-form reasoning. Zhao et al. [11] presented GraphText, a model that transforms graph representations into textual form—treating edges and nodes as sentences and fragments—thus enabling the use of any state-of-the-art LLM without architectural changes. While simplifying integration, this transformation introduces challenges in result interpretability.

Another active area of research focuses on prompting and adapting LLMs using graph-derived information. Tian et al. [5] introduced the concept of Graph Neural Prompting, where GNN-generated prompts encode the graph's logic and context, enabling LLMs to solve logic-driven and semantically rich queries more effectively. Li et al. [4] presented GraphAdapter, a method for tuning multimodal (vision-language) models using dual graphs—one for visual data and one for knowledge—each encoded separately and fused during generation. This showcases the generalizability of graph encoders beyond textual tasks.

Alongside engineering solutions, some studies focus on foundational adaptation techniques for LLMs. Hu et al. [9] proposed LoRA, a low-rank adaptation method that enables efficient fine-tuning of LLMs through small adaptation

matrices, reducing training overhead—an approach applicable to graph-based RAG systems. Qian et al. [6] examined LLM potential in property prediction tasks, comparing them to classical GNNs. While effective at generating property descriptions, LLMs showed limited applicability in domains requiring explicit graph structure, such as molecular modeling.

This literature review reveals several thematic clusters:

- Methodological and integrative studies on combining LLMs and graphs [1, 7, 8]
- Algorithmic advancements in graph-based RAG for QA tasks [2, 3, 10, 11]
- Prompt engineering and model adaptation using graph inputs [4, 5, 9]
- Application of LLMs in graph-reliant domains [6]

A notable tension emerges between two core approaches: transforming graphs into text for easier LLM integration (which risks losing semantic precision) and preserving graph topology (which demands complex encoders and high computational costs). Another gap lies in the lack of robust evaluation frameworks for RAG systems handling deep semantic queries—there are no widely accepted benchmarks that assess logical depth or the structural alignment of generated answers with the original graph. Furthermore, little attention is given to the dynamic updating of knowledge in real time or handling evolving graphs, which is crucial for applications requiring continual information integration.

The aim of this study is to analyze conceptual graph-based RAG models used for processing complex semantic queries.

This research introduces a conceptual synthesis and systematic classification of Conceptual Graph RAG architectures—from baseline Graph RAG models to PCST-optimized G-Retriever enhanced with soft prompt tuning and LoRA adaptation. It also formalizes a unified set of metrics and benchmarks for comprehensive evaluation of RAG systems in terms of quality, scalability, and dynamic knowledge integration.

The central hypothesis is that integrating conceptual graphs with the RAG paradigm—via optimized subgraph retrieval and soft prompt tuning of LLMs—can significantly enhance answer accuracy and reasoning explainability for complex semantic queries.

The study is based on a comparative analysis of existing research in this field, drawing insights from both conceptual frameworks and empirical results.

2. Theoretical Foundations and Unification Paradigms of Conceptual Graphs and RAG

Conceptual Graphs (CG) are a graph-based formalism for knowledge representation, where nodes denote concepts and edges encode semantic relations between them. Due to their explicit structure, CGs offer a flexible medium for representing and integrating facts from diverse sources [2].
When CGs are used as background knowledge, the core task becomes identifying relevant subgraphs and incorporating them into the context provided to the LLM. Knowledge fusion and RAG frameworks enable the alignment of linguistic embeddings with graph-based representations through two principal strategies:

- Knowledge Integration and Fusion: This approach adaptively selects relevant graph fragments to fine-tune the LLM. For example, KG-Adapter injects graph-specific adapter layers into the LLM architecture, allowing it to process graph-augmented inputs.
- Retrieval-Augmented Generation on Graphs: In architectures like Graph RAG, subgraphs are retrieved and trimmed using graph neural networks (GNNs), and their embeddings are combined with textual vectors to support answer generation [3].

Conceptual Graphs also function as reasoning scaffolds, guiding LLMs through permitted inference paths in multi-hop logic tasks. Figure 1 below outlines representative approaches to CG-LLM integration.

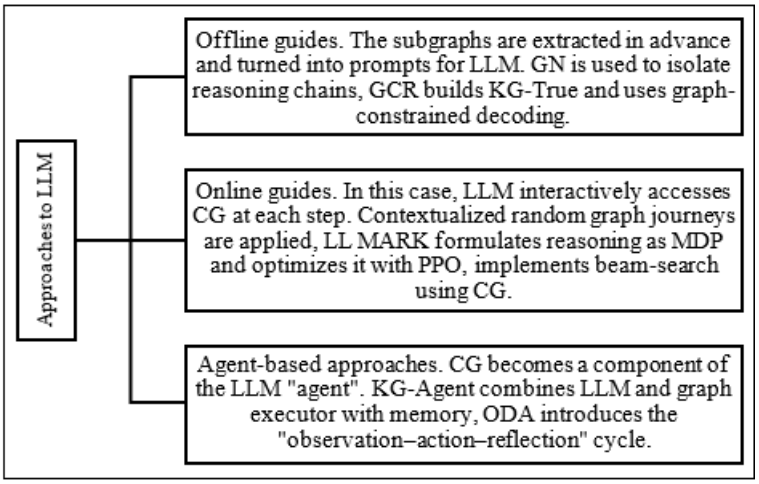

Figure 1: Approaches in LLM [1, 3]

Integrating CGs into both the filtering and final answer generation stages helps improve the accuracy and reliability of responses by enabling:

- Filtering and Validation: Methods such as ACT-Selection filter candidate answers based on Wikidata entity types; KG-Rank reorders outputs according to graph-based

relationships; and KGR retrofits LLM-generated outputs using graph alignment.
- Answer Refinement: Techniques like EFSUM reformulate LLM outputs based on CG-based summaries; InteractiveKBQA orchestrates iterative queries to graph databases; and LPKG trains LLMs in structured planning using CG templates [4].

The main paradigms for unifying CG and RAG architectures are summarized in Table 1.

Table 1: The Main Paradigms of Unification of CG and RAG [2, 5, 6]

Paradigm	Description	Example Methods
CG as Background Knowledge	Integration of relevant CG subgraphs into LLMs through knowledge fusion or RAG	InfuserKI, KG-Adapter, Graph RAG
CG as Reasoning Guidance	Providing structured inference paths for LLMs via CGs; includes both online and offline graph dialogue	EtD, GCR, Oreo, LLM-ARK, ToG
CG as Validators and Refiners	Filtering, re-ranking, and refining LLM outputs through factual validation and CG-grounded correction	ACT-Selection, KG-Rank, KGR, EFSUM, InteractiveKBQA, LPKG

This classification highlights three complementary ways to leverage Conceptual Graphs in RAG-based systems: as foundational knowledge, as navigational guides for reasoning, and as mechanisms for validating and refining final outputs. Together, these paradigms establish a solid theoretical basis for the structured development of Conceptual Graph RAG Models.

3. Architectures of Contemporary Conceptual Graph RAG Models

The Graph RAG method was originally proposed as a way to integrate Retrieval-Augmented Generation directly with graph structures, bypassing the conventional reliance on textual fragments [1, 7]. During knowledge preparation, embeddings are generated for nodes and edges of the underlying ontological graph using Graph Neural Networks (GNNs). For each textual query, the k-nearest elements of the graph are retrieved based on cosine similarity, followed by the construction of a candidate subgraph: irrelevant edges are pruned using predefined thresholds, and the remaining components are refined through GNN processing. This subgraph is then linearized into textual form and concatenated with the original query, serving as augmented context for the large language model (LLM)—a design that substantially reduces hallucination risk.

Despite notable improvements in answer quality, Graph RAG struggles to scale to large graphs. The reliance on basic kNN retrieval limits its coverage, and rigid pruning thresholds can eliminate structurally important connections, reducing the completeness of generated answers.

To overcome these limitations, G-Retriever advances the Graph RAG framework by operating on textual graphs and formulating subgraph selection as a Prize-Collecting Steiner Tree (PCST) optimization problem. During indexing, node and edge attributes are encoded using a pretrained language model (e.g., SentenceBERT) into embeddings of dimension d. Upon receiving a query x_s, its embedding z_s is computed. The top-k nodes V_k and edges E_k are selected based on cosine similarity ($\cos(z_s, z_n)$, $\cos(z_s, z_e)$). Each element is assigned a prize value prize = k – i, where i is its rank; others receive zero. The objective becomes finding a connected subgraph S that maximizes the sum of prize values minus the edge costs [2, 3].

The selected optimal subgraph is then flattened into a linear textual sequence, embedded using a Text Embedder and passed through an MLP projection to form a graph embedding. This is combined with the query tokens and input into the LLM. Importantly, LLM weights remain frozen; training is limited to the GNN and MLP layers, enabling efficient task-specific adaptation. G-Retriever's key advantages include high scalability (via controlled subgraph size), strong robustness against hallucinations (by grounding generation in retrieved facts), and high explainability (via return of the PCST subgraph). In terms of system optimization, two main directions are distinguished:
- Soft Prompt Tuning: Prompt tokens are concatenated with the text embedding and fine-tuned without modifying LLM weights. An analogous graph-based variant is implemented in GraphToken [8], where the entire graph embedding serves as a prompt. However, the lack of node selection limits its scalability.
- Parameter-Efficient Fine-Tuning (PEFT) with LoRA (Low-Rank Adaptation): Hu et al. [9] demonstrated that introducing sparse low-rank matrices into the model allows training a small subset of parameters while keeping the LLM core intact. Combined with PCST-based retrieval in G-Retriever, LoRA provides performance gains over soft-prompt tuning alone [9].

Table 2 presents a comparison of key features across the main graph-based architectures.

Table 2: Comparison of Graph-Based RAG Architectures [2, 7, 8, 9]

Characteristic	Graph RAG	G-Retriever	GraphToken
Graph Type	Knowledge Graphs (e.g., ontologies)	Textual graphs	Conceptual Graphs / Structured Graphs
Retrieval Method	kNN via GNN embeddings	kNN + PCST optimization	Whole-graph embedding
Subgraph Size	Threshold-based pruning	Optimized PCST subgraph	All nodes via soft prompt
LLM Integration	concat(subgraph, query)	soft-prompt (MLP → LLM)	soft-prompt (graph embedding)
Scalability	Moderate	High	Low
Robustness to Hallucinations	Moderate	Strong	Weak
Explainability	Limited	High (PCST subgraph returned)	Low

In summary, contemporary Conceptual Graph RAG models have evolved from simple subgraph extraction and concatenation (Graph RAG) to more advanced optimization-based architectures like G-Retriever, which incorporate PCST algorithms and soft prompt tuning. These developments improve scalability, interpretability, and robustness against hallucinations—key requirements for handling complex semantic QA tasks.

4. Evaluating the Applicability of Graph-Based RAG Models

To comprehensively evaluate Conceptual Graph RAG models, the literature outlines four main categories of metrics:

- Retrieval Stage Metrics: These assess the quality of extracting relevant graph fragments. Core metrics include subgraph-level precision and recall, as well as context relevance, which measures how well the content of the subgraph aligns with the intent of the query.
- Answer Generation Metrics: These evaluate the quality of the final textual response. Standard measures include BERTScore and Mean Reciprocal Rank (MRR) to assess linguistic and semantic similarity to reference answers, along with faithfulness (factual correctness) and answer relevance (logical alignment with the query).
- Reasoning Metrics: A key indicator is Hop-Acc, which captures the proportion of correct transitions within a multi-hop reasoning chain.
- Hallucination Metrics: For systems with explicit graph references, these include the proportion of Valid Nodes, Valid Edges, and fully accurate Cited Subgraphs (Fully Valid Graphs) [2, 10].

Table 3 presents benchmark datasets commonly used to evaluate LLM+KG QA methods.

Table 3: The Main LLM+KG Benchmarks and Their Characteristics [2, 10]

Dataset	Category	Graph Type	Metric
PATQA	Complex QA	Temporal multi-hop QA	Accuracy
MINTQA	Complex QA	Multi-hop (new/tail knowledge)	Accuracy
MedQA	Complex QA	Medical exam QA	Accuracy
WebQSP	KBQA	Freebase subgraphs (2-hop)	Hit@1
CAQA	KBQA	Attribution QA over KG	Accuracy
CR-LT KGQA	KBQA	Commonsense + long-tail KGQA	Accuracy
KGs+LLMs for QA	SQL QA	Enterprise SQL databases	Accuracy
XplainLLM	Explainable QA	Grounded explanations for LLMs	–
LLM-KG-Bench	KG Engineering	Scalable KG engineering tasks	–

The KAG framework (AntGroup) exemplifies domain-knowledge augmented generation, combining graph-based and vector-based retrieval to enhance response quality. This architecture supports both structured ontological relationships and semantic text embeddings—especially critical in domains such as e-government and e-health, where accuracy and completeness are essential.

The Graph RAG demo from NebulaGraph showcases a hybrid setup that includes (1) an NLP2Cypher engine for translating natural language into graph queries, (2) conventional vector-based RAG, and (3) a dedicated Graph-vector RAG module for complex multi-hop queries. This configuration enables advanced semantic navigation across large graphs and accommodates topology-aware reasoning.

In e-commerce, the Retrieval-Augmented Customer Service QA system illustrates how incorporating a knowledge graph boosts chatbot accuracy and answer justification. Demonstrated at SIGIR, its architecture fuses vector indexing with graph-based search across product data and business rules—substantially improving the customer experience while reducing the risk of misinformation [2, 7].

Personalized recommendations and analytics powered by CG-RAG models are actively deployed in media platforms for relevant content delivery and in medical information systems to support diagnostic decisions. Knowledge graphs contribute not only to output explainability but also to quality assurance by enabling expert inspection of intermediate reasoning steps [11].

These examples underline the wide applicability of combining large language models with structured graph representations. Such hybrid systems deliver enhanced reliability, richer explainability, and improved accuracy across diverse QA domains.

5. Conclusion

This study substantiates the necessity and feasibility of integrating Conceptual Graphs with Retrieval-Augmented Generation for addressing complex, semantically rich question-answering tasks involving multi-hop and context-dependent reasoning. A comparative analysis of Graph RAG and G-Retriever architectures demonstrates G-Retriever's advantages in scalability, interpretability, and robustness against hallucinations. Optimization methods such as Prompt Tuning and LoRA are shown to reduce the number of trainable parameters without compromising model accuracy.

A comprehensive review of metrics and benchmarks further supports the generalizability and effectiveness of the proposed approach across various domains. Looking ahead, future research directions include developing dynamic, learnable subgraph retrieval modules and expanding into multimodal CG-RAG scenarios incorporating audio and video attributes.

References

[1] S. Pan et al., "Unifying large language models and knowledge graphs: A roadmap," IEEE Transactions on

Knowledge and Data Engineering, 36 (7), pp. 3580–3599, 2024.
[2] X. He et al., "G-retriever: Retrieval-augmented generation for textual graph understanding and question answering," Advances in Neural Information Processing Systems, 37, pp. 132876–132907, 2024.
[3] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," Advances in Neural Information Processing Systems, 33, pp. 9459–9474, 2020.
[4] X. Li et al., "GraphAdapter: Tuning vision-language models with dual knowledge graph," Advances in Neural Information Processing Systems, 36, pp. 13448–13466, 2023.
[5] Y. Tian et al., "Graph neural prompting with large language models," Proceedings of the AAAI Conference on Artificial Intelligence, 38 (17), pp. 19080–19088, 2024.
[6] C. Qian et al., "Can large language models empower molecular property prediction?," arXiv preprint arXiv:2307.07443, pp. 1–10, 2023.
[7] Y. Gao et al., "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, 2, p. 1, 2023.
[8] B. Perozzi et al., "Let your graph do the talking: Encoding structured data for LLMs," arXiv preprint arXiv:2402.05862, pp. 1–8, 2024.
[9] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," ICLR, 1 (2), p. 3, 2022.
[10] P. Sen, S. Mavadia, A. Saffari, "Knowledge graph-augmented language models for complex question answering," Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE), pp. 1–8, 2023.
[11] J. Zhao et al., "GraphText: Graph reasoning in text space," arXiv preprint arXiv:2310.01089, pp. 1–9, 2023.