Global Research Journal in Engineering, Science, Technology and Management (GRJESTM)
Volume 02, Issue 03 (May-June) 2026, pp. 370-375
GRJESTM :: An odaswa™ Journal
ISSN : 3107-3611 (Online)

TECHNICAL SERIES | TECHNOLOGY INSIGHT – TN09

Retrieval-Augmented Generation with Graph-Based Retrieval: Improving Context for Technical AI Systems

Aditya Rajput[1]* and Shreya Mehta[2]

Technology Snapshot.

Technology Domain	AI systems and Information retrieval
Core Technologies	LLMs, vector search, knowledge graphs, hybrid retrieval, RAG
System Architecture	Retrieval + graph reasoning + generation
Primary Applications	Technical support, enterprise search, research assistance, knowledge management
Key Advantages	Higher context accuracy, better reasoning, reduced hallucinations
Key Challenges	Graph construction, scalability, latency, data freshness

Keywords: Retrieval-Augmented Generation, Knowledge Graphs, Graph-Based Retrieval, Vector Search, Large Language Models, Hybrid Retrieval, Grounded Generation.

* 1. Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India.
2. Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India.

Copyright: © 2026 Aditya Rajput and Shreya Mehta. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. Published By **odaswa**. Published in Chennai, Tamil Nadu, India

1. Introduction

Large Language Models (LLMs) have demonstrated considerable capabilities in natural-language understanding, information synthesis and question answering. However, their responses may be incomplete or inaccurate when questions require specialized knowledge, current information or relationships between multiple technical concepts. Retrieval-Augmented Generation (RAG) addresses these limitations by retrieving relevant external information and supplying it to the language model before response generation [1].

Conventional RAG systems commonly use vector search to identify documents that are semantically similar to a query. Dense passage retrieval has demonstrated the effectiveness of embedding-based retrieval for open-domain question answering [2]. Nevertheless, similarity-based retrieval may not adequately capture explicit relationships between entities or connections distributed across multiple documents.

Graph-based retrieval extends this approach by incorporating knowledge graphs, in which information is represented as entities and their relationships. Recent surveys have organized GraphRAG methods around graph construction, graph-guided retrieval and graph-enhanced generation, highlighting their potential for knowledge-intensive applications [3], [4]. Graph-based methods can also support query-focused summarization across large document collections by organizing related entities and generating summaries of connected communities [5].

By combining semantic retrieval with graph traversal, graph-based RAG can retrieve both relevant documents and relationship-based context. This approach is particularly useful for technical support, scientific research, enterprise knowledge management and other applications requiring contextual reasoning across interconnected information. However, its effectiveness depends on graph quality, retrieval design and the availability of reliable source evidence [3], [4].

2. Technical Background

RAG combines two major components: a retrieval system and a generative model. The retrieval system searches an external knowledge source, while the language model uses the retrieved material to construct a response. This enables the system to incorporate information that may not be present in the model's original training data [1].

Vector databases represent documents or text segments as numerical embeddings. A query is converted into an embedding, and similarity search identifies text segments that are semantically related to the query. Dense Passage Retrieval demonstrated how learned dense representations can improve passage retrieval for open-domain question answering [2]. However, similarity alone may not reveal relationships between different entities, such as the dependency between a software component and its associated failure, or the connection between a scientific method and its reported findings.

A knowledge graph represents information using nodes and edges. Nodes may represent people, organizations, concepts, documents or technical components, while edges describe relationships between them. For example, a graph may represent relationships such as component—depends on—library or research paper—uses—method. Knowledge graphs provide an explicit representation of interconnected information, making relationships available for structured querying and traversal [3], [4].

Graph-based RAG integrates these complementary representations. Vector search identifies semantically relevant information, while graph retrieval explores connections among entities and documents. Depending on the implementation, graph-based retrieval may also incorporate entity descriptions, relationship information, graph neighborhoods and summaries of connected communities [4], [5].

A further development is the use of knowledge graphs as structured retrieval resources for multi-hop question answering. HippoRAG, for example, combines knowledge graphs with retrieval mechanisms inspired by associative memory to support retrieval across related information [6]. These approaches demonstrate that graph-based retrieval can extend beyond retrieving isolated passages to connecting evidence distributed across a knowledge base.

3. Operating Principle

A graph-based RAG workflow generally consists of the following stages.

Stage 1: Query interpretation. The user's question is analyzed to identify the main concepts, entities and information requirements. Complex queries may be decomposed into smaller retrieval tasks. The query processor can determine whether the question requires a specific fact, a multi-step relationship or a summary of information distributed across a document collection [3], [4].

Stage 2: Semantic retrieval. Vector search identifies relevant passages from indexed documents. Keyword or metadata-based search may also be used to improve retrieval for exact technical terms, identifiers and domain-specific expressions. Hybrid retrieval can combine semantic matching with lexical search to improve the coverage of relevant information [2], [7].

Stage 3: Graph-based retrieval. The system identifies relevant entities in the knowledge graph and explores their relationships. Depending on the query, graph traversal may retrieve connected components, associated events, supporting documents or multi-step relationships. Graph-based methods can also use clusters or communities of connected entities to support broader questions about a document collection [5], [6].

Stage 4: Context integration. The retrieved passages and graph-derived information are combined, ranked and filtered. Duplicate, irrelevant or contradictory information should be identified where possible before the context is passed to the language model. The retrieval orchestrator may use relevance scores, relationship strength, source reliability and query-specific constraints to determine which evidence should be retained [3], [4].

Stage 5: Grounded generation. The LLM generates a response using the retrieved context. Source references or document citations can be included to help users verify the information. The generation process should distinguish evidence explicitly supported by the retrieved material from interpretations or inferences derived from that evidence [1], [5].

Stage 6: Evaluation and verification. The generated response is checked for relevance, factual consistency and support from retrieved evidence. Automated evaluation frameworks such as RAGAS provide metrics for assessing retrieval quality, faithfulness and response relevance [8]. For high-reliability applications, automated validation and human review may also be required.

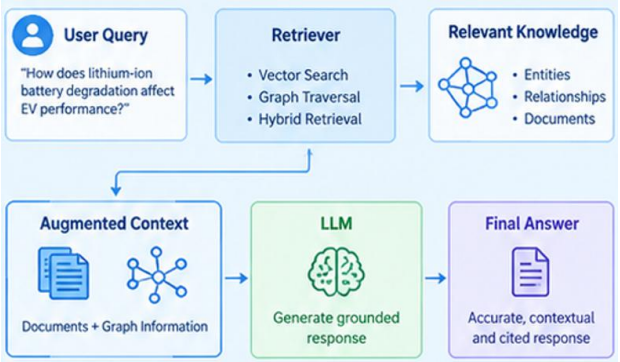

Fig. 1. Workflow of Graph-Based Retrieval-Augmented Generation (RAG), illustrating query interpretation, hybrid retrieval, graph traversal, context integration and grounded response generation.

4. System Architecture

A typical graph-based RAG architecture consists of a document-processing pipeline, a vector database, a knowledge graph, a retrieval orchestrator and an LLM.

The document-processing pipeline collects information from technical documents, research publications, manuals, databases or other

approved sources. Documents are divided into suitable text segments, which are converted into embeddings for vector search. In parallel, entities and relationships may be extracted to construct or update the knowledge graph. Links between graph entities and their original source documents are important for traceability.

During query execution, the retrieval orchestrator coordinates vector search and graph traversal. The two retrieval paths may operate in parallel or sequentially, depending on the system design. Their outputs are then ranked and combined to create the final context. The LLM uses this context to generate an answer, ideally with references to the source material.

Graph-based retrieval can improve contextual coverage, but its performance depends on the quality of the underlying graph and retrieval strategy. Incorrect entity extraction, missing relationships or outdated documents can introduce errors. Therefore, graph construction and maintenance are essential parts of the architecture rather than optional preprocessing activities.

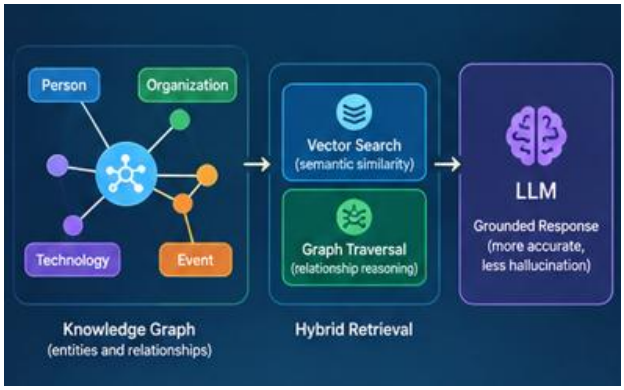

Fig. 2. Architecture of a Graph-Based RAG System

5. Advantages and Applications

Graph-based RAG offers several potential advantages over retrieval systems based exclusively on vector similarity.

Relationship-aware retrieval: The system can retrieve information connected through explicit relationships, even when the relevant details appear in different documents.

Improved multi-hop question answering: Questions requiring several linked facts can benefit from graph traversal, provided the necessary entities and relationships are represented accurately.

Contextual grounding: Combining retrieved passages with structured relationships can provide the language model with more informative context and help reduce unsupported responses.

Traceability: When retrieved graph elements are linked to source documents, the system can provide references that support verification and auditing.

Potential applications include technical documentation assistants, enterprise knowledge search, software dependency analysis, scientific literature exploration, engineering troubleshooting and domain-specific research support. In geospatial and remote-sensing workflows, for example, a graph could connect satellite sensors, datasets, processing methods, geographic regions and published findings, helping researchers retrieve related technical information across multiple sources.

These benefits are not automatic. Graph-based retrieval does not eliminate hallucinations, and an incomplete or inaccurate knowledge graph may negatively affect response quality.

Implementation Challenges and Future Directions

The construction and maintenance of a high-quality knowledge graph can require significant processing and domain expertise. Entity recognition, relationship extraction, entity resolution and schema design are particularly challenging when documents contain ambiguous terminology or inconsistent naming conventions.

Scalability is another concern. Large graphs may contain millions of nodes and relationships, making traversal, filtering and ranking computationally expensive. A practical implementation must balance retrieval depth against response latency and resource consumption. Excessive graph traversal may introduce irrelevant information, while overly

restrictive traversal may omit important connections.

Information freshness is also important. Technical documents, software dependencies, scientific findings and organizational procedures change over time. The system should support controlled updates, source versioning and mechanisms for identifying outdated or conflicting information.

Evaluation should measure more than the fluency of generated responses. Useful metrics include retrieval precision and recall, contextual relevance, answer correctness, faithfulness to retrieved evidence, citation accuracy, latency and computational cost. Comparisons with conventional vector-based RAG should use representative queries and consistent evaluation conditions.

Future developments are likely to focus on dynamic knowledge graphs, more efficient hybrid retrieval, improved entity and relationship extraction, graph-aware reranking and stronger evidence verification. Domain-specific retrieval strategies may further improve performance in technical environments where terminology, provenance and relationship accuracy are critical.

6. Conclusion

Graph-based Retrieval-Augmented Generation extends conventional RAG by combining semantic document retrieval with structured relationships represented in knowledge graphs. This combination can provide richer context for complex queries, support multi-hop information retrieval and improve the traceability of generated answers. Its practical effectiveness depends on accurate graph construction, efficient retrieval orchestration, reliable source management and systematic evaluation. Graph-based RAG is therefore a promising architecture for technical AI systems, particularly where information is distributed across interconnected documents and structured knowledge sources.

References

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems, Vol. 33, pp. 9459–9474, 2020.

[2] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen and W.-T. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

[3] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang and S. Tang, "Graph Retrieval-Augmented Generation: A Survey," arXiv preprint arXiv:2408.08921, 2024. doi: 10.48550/arXiv.2408.08921.

[4] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, M. Halappanavar, R. A. Rossi, S. Mukherjee, X. Tang, Q. He, Z. Hua, B. Long, T. Zhao, N. Shah, A. Javari and J. Tang, "Retrieval-Augmented Generation with Graphs (GraphRAG)," arXiv preprint arXiv:2501.00309, 2025. doi: 10.48550/arXiv.2501.00309.

[5] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. Osazuwa Ness and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv preprint arXiv:2404.16130, 2024. doi: 10.48550/arXiv.2404.16130.

[6] G. Gutierrez, Y. Shu, Y. Gu, M. Yasunaga and Y. Su, "HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models," Advances in Neural Information Processing Systems, 2024.

[7] C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval. Cambridge University Press, 2008.

[8] S. Es, J. James, L. Espinosa-Anke and S. Schockaert, "RAGAS: Automated Evaluation of Retrieval Augmented Generation," arXiv

preprint arXiv:2309.15217, 2023. doi: 10.48550/arXiv.2309.15217.

[9] H. Han, L. Ma, H. Shomer, Y. Wang, Y. Lei, K. Guo, Z. Hua, B. Long, H. Liu, C. C. Aggarwal and J. Tang, "RAG vs. GraphRAG: A Systematic Evaluation and Key Insights," arXiv preprint arXiv:2502.11371, 2025.

[10] M. Cheng, Y. Luo, J. Ouyang, Q. Liu, H. Liu, L. Li, S. Yu, B. Zhang, J. Cao, J. Ma and E. Chen, "A Survey on Knowledge-Oriented Retrieval-Augmented Generation," arXiv preprint arXiv:2503.10677, 2025.