Технічні науки

UDC 004.8:004.652.1

Gartman Ievgen

CEO and founder, Bridge.Digital,

(Austin, TX, USA)

INFLUENCE OF GRAPH KNOWLEDGE TOPOLOGY ON THE EFFICIENCY OF RAG PROCESSES

Summary. *This paper examines how the topological characteristics of knowledge graphs affect the performance of Retrieval-Augmented Generation (RAG) systems built atop large language models (LLMs). As LLM "hallucinations" remain a critical challenge, integrating structured knowledge via graph representations promises to improve response accuracy. We set out to identify which graph-topology metrics and their combinations most strongly influence RAG effectiveness, and to quantify their effects on both answer quality and system throughput. By surveying and comparing existing studies, we demonstrate that an optimally structured graph—when paired with hybrid indexing—yields higher answer precision without increasing the LLM's contextual load, outperforming traditional RAG approaches. Our findings will be of interest to AI researchers and engineers leveraging LLMs and graph databases for high-precision RAG workflows, as well as architects of advanced question-answering and summarization platforms who seek an ideal trade-off between semantic richness and computational efficiency.*

Key words: *Retrieval-Augmented Generation, knowledge graph, graph topology, retrieval granularity, hybrid indexing, large language models, multi-hop reasoning.*

Introduction. Large language models (LLMs) have revolutionized natural language understanding and generation in recent years. Yet they remain prone to

so-called "hallucinations," producing plausible but incorrect statements, and often lack up-to-date domain knowledge. Without grounding in external knowledge sources, LLMs are vulnerable to factual errors and outdated information, underscoring the need to integrate dedicated knowledge stores to boost answer accuracy. In the e-commerce and retail sectors, the accuracy of product information representation, personalization of commercial offers, relevance of catalog search results, and speed of handling customer requests are key factors that directly affect conversion rates and the cultivation of long-term customer loyalty. However, traditional data-processing and analysis methods typically lack the scalability and adaptability needed to manage high volumes of information and complex interrelationships. In this context, the Graph-RAG architecture emerges as a promising solution for a wide array of challenges—from intelligent search and recommendation-system development to the automation of customer support and in-depth analysis of user intent.

This paper aims to fill that gap by analyzing how specific graph-topology parameters, alone and in combination, govern RAG effectiveness.

Our central hypothesis is that selecting the right mix of topological features—whether extracting shortest paths or k-hop subgraphs, leveraging hybrid "graph + vector" indexes, or enabling incremental updates—can substantially raise answer accuracy and coverage without increasing the LLM's contextual footprint compared to conventional RAG pipelines.

Materials and methods. A foundational thread in this line of work emphasizes adaptive, graph-aware retrieval. Chen J. et al. [6] and later Chen R. [1] argue for models that dynamically traverse knowledge graphs to fetch domain facts in real time, thereby anchoring generated text firmly in structured information. In a related survey, Guo and Zhao [13] review generative techniques for crafting graph topologies themselves, highlighting how design-time graph structure impacts downstream retrieval.

Beyond retrieval, several studies investigate end-to-end architectures for graph-enhanced RAG. Jiang X. et al. [2] introduce Ragraph, a unified framework that trains over graph embeddings and retrieval signals to elevate both extraction and generation quality. He X. et al. [10] propose G-Retriever, a method that refines graph comprehension for question answering, demonstrating significant gains in graph-based reasoning.

Wang Y. et al. [12] further show that tailoring retrieval to key topological metrics—such as node centrality and subgraph connectivity—yields more accurate and coherent text outputs. Bahr L. et al. [3] use knowledge graphs to trace faults in industrial workflows, while Oh J. H. et al. [14] employ synthetic brain networks to pinpoint disease markers. Zhang C. et al. [11] leverage graph neural nets to inform architectural design decisions, illustrating the broad applicability of graph-enriched generation.

To simplify graph use in RAG, several teams have built modular toolkits. Rani M. et al. [7] add a "robust retrieval" layer that selects search strategies based on graph density. Ngangmeni J. and Rawat D. B. [8] release GraphRAG, an out-of-the-box wrapper that auto-constructs a literature graph and highlights pivotal citations. Edge D. et al. [9] apply graph-augmented summarization to produce concise, query-focused abstracts, showing that global graph metrics boost coherence.

Datasets and benchmarks have also evolved to test graph-RAG resilience. Yu W. et al. [4] present IFQA, an open QA corpus with counterfactual twists that stress-test systems' ability to navigate alternate graph topologies. Sun H., Bedrax-Weiss T., and Cohen W. W. [5] demonstrate iterative retrieval cycles across both knowledge bases and text, laying groundwork for multi-hop graph strategies.

Existing research in the Graph-RAG domain primarily focuses on general methods for improving accuracy [1, 2, 6] or on narrowly specialized application scenarios such as error analysis [3] and medical diagnosis [14]. However, a clear gap exists in the systematic investigation of these technologies' potential

specifically for eCommerce tasks. In particular, it is necessary to analyze how different knowledge-graph topologies affect a Graph-RAG system's ability to generate personalized product recommendations and to provide detailed, context-dependent responses to complex product queries.

Results and discussion. When integrating knowledge graphs (KG) into a RAG system, the graph's structure is key. It defines how knowledge is stored, indexed, and updated. Typically, KG uses an entity-relation graph composed of entity nodes and relation edges, which model domain semantics. This works well for logical inference across multiple steps. But, as the graph grows denser, scaling issues arise with deep traversal and indexing [3, 12]. A Document-Structure KG (DSG) addresses this by considering text fragments as nodes and edges that reflect semantic proximity. For clarity, the key characteristics are summarized in Table 1.

Table 1

Key types of knowledge graphs and their topological characteristics

KG Type	Nodes / Edges	Topology	Pros	Cons	An example of an application in e-Commerce
Entity – Relation	diverse entities; 1:1, 1:N, N:M relations	heterogeneous, medium density	clear semantics; supports multi-hop inference	search complexity grows in large graphs	modeling a product catalog of thousands of SKUs with their attributes, categories, and brands, as well as relations between products (e.g., "frequently bought together," "similar products") and customer data (purchase histories, preferences) for personalized recommendations
Document-Structure	`text fragments; edges ≈ contextual similarity`	homogeneous, shallow	simple to build; low indexing cost	loses fine-grained semantic connections	building a graph from an extensive corpus of customer reviews and product overviews—nodes represent individual reviews or their semantic segments, edges denote semantic similarity or shared aspects (e.g., "camera quality," "battery life"), enabling rapid retrieval of relevant opinions for specific user queries
Community-	community nodes;	wide & shallow	drastically reduces	no detail within	capturing behavioral data for large user groups (e.g., "shoppers interested in eco-

KG Type	Nodes / Edges	Topology	Pros	Cons	An example of an application in e-Commerce
Summary	inter-community edges		prompt context size	aggregated communities	friendly products") or aggregated characteristics of product categories to identify trends and inform broad marketing strategies
Hetero/Dynamic KG	multi-typed nodes/edges; metadata	evolving topology	flexible updates; rich semantics	high complexity in management and indexing	modeling a continuously updated catalog with diverse product types and user interactions (views, purchases, reviews) to deliver real-time, up-to-date recommendations and search results

When constructing KGs for RAG systems, topology dictates the trade-off between semantic expressivity and computational efficiency. Some RAG methods employ native graph-traversal algorithms (BFS/DFS) without preliminary vectorization: while semantically exact, these traversals become prohibitively slow on graphs exceeding 10^6 nodes.

Hybrid schemes combine a "graph" index for structural queries with a "vector" index for textual attributes, balancing precision and performance. In such architectures, new nodes and edges are simply appended to the native graph index, and only the new elements require embedding. This design supports real-time KG updates without full re-indexing.

When integrating knowledge graphs into RAG systems, it is essential to choose the right level of semantic and structural granularity for the entities you extract, and to formalize the extraction process itself—from path-finding algorithms and pattern templates to methods for aggregating contextual links. The amount of detail in the extracted subgraph directly determines both the volume and semantic richness of the context passed to the LLM, which in turn has a direct impact on the accuracy and completeness of the system's answers [14].

At the most minimal granularity, only individual entities or single edges are retrieved. This approach delivers extremely high throughput—often O(1)–O(log N) in a vector index—while transmitting very little data, but it cannot capture

complex inter-entity relationships. An example of this is PullNet [5, 8], which operates purely on entity retrieval for straightforward question answering tasks.

Table 2 illustrates the main levels of extraction granularity in Graph-RAG systems.

Table 2

The main levels of extraction granularity in Graph-RAG systems

Granularity	Description	Typical Use Cases	Complexity
Nodes / edges	Individual entities or single edges	Fact retrieval: obtaining a specific product attribute (price, stock availability); fact checking (e.g., "Is Phone X waterproof?")	O(1)–O(log N) (vector lookup)
Triples	Single facts ⟨h,r,t⟩	One-step question answering (e.g., "Which smartphones support 5G?" "What material is this bag made of?")	O(N)
Paths	Sequential chains of node–edge–node	Multi-hop reasoning: finding products by complex criteria ("a laptop for a designer with a > 15″ screen, > 16 GB RAM, and an NVIDIA GPU"); analyzing a user's click sequence to infer intent	NP-hard
Subgraphs (k-hop)	k-radius neighborhood around a node	Summarization and complex QA; personalized recommendation generation ("Since you viewed product A and purchased product B, you may like product C, which is often bought with B and shares attributes with A"); comparative product analysis	$O(D^k)$

When processing a client query such as "Recommend a good gaming phone under $500" the local retrieval layer concentrates on nodes corresponding to gaming-specific attributes—high-performance processor, sufficient RAM, and a high refresh-rate display—while ensuring each candidate meets the price constraint. The global analysis layer aggregates user ratings to pinpoint models with the strongest positive trend in gamer satisfaction. The hybrid approach implemented in LightRAG [11, 13] then combines this initial extraction of key entities (phone models, gaming specifications, and price range) with context expansion through the integration of relevant user reviews and expert evaluations, thereby delivering a comprehensive solution for selecting the optimal product.

Depending on the research goal and the desired precision of information retrieval from graph indexes, three fundamental workflows are generally distinguished. The simplest, one-shot retrieval, issues a direct query against a vector or graph index without any downstream filtering. This delivers very low latency but can return irrelevant results [2; 7].

To improve precision, multistage retrieval appends ranking or pruning steps to the raw hit set. GraphRAG [9], for instance, post-processes retrieved candidates into ordered "community summaries" via a large language model. Likewise, GRAG [1] applies graph-neural-network masking over the extracted subgraph to eliminate low-relevance nodes. A more sophisticated pipeline, G-Retriever [10], first selects pivotal nodes and edges, then synthesizes an optimal Steiner subgraph around them.

Table 3 summarizes these graph-fragment transformation and context-enhancement techniques.

Table 3

Methods of Graph Fragment Transformation and Context Enhancement [1; 2; 4; 6]

Method	Category	Description	Example Implementations
Serial numbering and direct verbalization	Conversion	Assign indexes + textual attributes to convey order and basic context	G-Retriever
Node sequences	Conversion	Linear chains of nodes and relations for multi-hop reasoning	GraphChain-of-Thought; ToG
Embedding text attributes	Conversion	Include full entity/relation descriptions (key–value) in the prompt	LightRAG; KGP
Subgraph summarization	Conversion	LLM-generated summaries of communities or k-hop subgraphs	GraphRAG,
Hierarchical descriptions (tree traversal)	Conversion	Nested text structure: root → neighbors → …	GRAG

Method	Category	Description	Example Implementations
Multi-stage ranking and filtering	Context enhancement	Generate answers per subgraph, rank by LLM score, pick top k	GraphRAG
Soft prompts via GNN embeddings	Context enhancement	Prepend learnable GNN embeddings before the text without changing the LLM itself	G-Retriever; GRAG
Masking low-relevance elements	Context enhancement	Use GNN to hide nodes/edges deemed irrelevant before generation	GRAG

However, it faces the following limitations: 1) Dependence on the completeness and freshness of the underlying knowledge graph—otherwise, semantic search accuracy and recommendation quality degrade; 2) High computational cost and latency when extracting deep subgraphs (k-hop), which can impair response times in real-world e-commerce settings; 3) Complexity of integrating and maintaining the graph infrastructure, requiring substantial resources for updates, monitoring, and version coordination; 4) Limited transparency of the model's internal decision logic, making it difficult to interpret results or diagnose incorrect behavior and necessitating additional auditing and validation mechanisms.

Conclusion. This paper has examined how the topological characteristics of knowledge graphs impact the performance of Retrieval-Augmented Generation systems. Our findings demonstrate that Graph-RAG architectures deliver more accurate and reliable answers—significantly reducing hallucination risks and offering richer semantic understanding of queries. At the same time, we have identified key challenges, notably: Scalability when handling dynamic or highly heterogeneous graph structures; Chunking strategies for multimodal inputs, where existing methods may not yield optimal context granularity; Dependence on the underlying LLM and embedding model, which can constrain the overall robustness and generality of the approach.

References

1. R. Chen, "Retrieval-Augmented Generation with Knowledge Graphs: A Survey," in Proceedings of the Computer Science Undergraduate Conference 2025 @ XJTU, pp. 1–8, 2025.

2. X. Jiang et al., "Ragraph: A General Retrieval-Augmented Graph Learning Framework," Advances in Neural Information Processing Systems, 37, pp. 29948–29985, 2024.

3. L. Bahr et al., "Knowledge Graph Enhanced Retrieval-Augmented Generation for Failure Mode and Effects Analysis," Journal of Industrial Information Integration, 45, p. 100807, 2025.

4. W. Yu et al., "Ifqa: A Dataset for Open-Domain Question Answering under Counterfactual Presuppositions," arXiv preprint arXiv:2305.14010, 2023.

5. H. Sun, T. Bedrax-Weiss & W. W. Cohen, "Pullnet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text," arXiv preprint arXiv:1904.09537, 2019.

6. J. Chen et al., "Benchmarking Large Language Models in Retrieval-Augmented Generation," Proceedings of the AAAI Conference on Artificial Intelligence, 38 (16), pp. 17754–17762, 2024.

7. M. Rani et al., "To Enhance Graph-Based Retrieval-Augmented Generation (RAG) with Robust Retrieval Techniques," in 2024 18th International Conference on Open Source Systems and Technologies (ICOSST), IEEE, pp. 1–6, 2024.

8. J. Ngangmeni & D. B. Rawat, "Swamped with Too Many Articles? GraphRAG Makes Getting Started Easy," AI, 6 (3), p. 47, 2025.

9. D. Edge et al., "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv preprint arXiv:2404.16130, 2024.

10. 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.

11. C. Zhang et al., "End-to-End Generation of Structural Topology for Complex Architectural Layouts with Graph Neural Networks," Computer-Aided Civil and Infrastructure Engineering, 39 (5), pp. 756–775, 2024.

12. Y. Wang et al., "Topology-Aware Retrieval Augmentation for Text Generation," in Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, pp. 2442–2452, 2024.

13. X. Guo & L. Zhao, "A Systematic Survey on Deep Generative Models for Graph Generation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 45 (5), pp. 5370–5390, 2022.

14. J. H. Oh et al., "Graph-Based Conditional Generative Adversarial Networks for Major Depressive Disorder Diagnosis with Synthetic Functional Brain Network Generation," IEEE Journal of Biomedical and Health Informatics, 28 (3), pp. 1504–1515, 2023.