Vol. 1, No. 4, pp.: 1-4, 2023

GraphRAG vs. VectorRAG: An Empirical Comparison of Retrieval Architectures for Knowledge-Intensive Tasks

Chiara Moretti[1], **James Okwu**[2,*], **Hana Kobayashi**[1]

[1] EURECOM, Sophia Antipolis, France, 06410

[2] African Institute for Mathematical Sciences, Cape Town, South Africa, 7945

[*] *j.okwu@aims.ac.za*

Article Information	
Received	7 June 2023
Accepted	20 December 2023
DOI	https://doi.org/10.63646/datamind.2023.010401

Abstract

Retrieval-augmented generation (RAG) has become a standard approach for grounding large language model outputs in factual, updateable knowledge bases. Two dominant retrieval paradigms have emerged: vector-based retrieval (VectorRAG), which uses dense embedding similarity to identify relevant passages, and graph-based retrieval (GraphRAG), which traverses explicit knowledge graph structures to surface related entities and relationships. Despite both approaches seeing wide adoption, rigorous head-to-head comparisons under controlled conditions remain scarce. This paper presents an empirical comparison of VectorRAG and GraphRAG across four knowledge-intensive task families — single-hop QA, multi-hop QA, entity disambiguation, and temporal reasoning — using three corpora of varying knowledge graph density. We evaluate five specific implementations: naive RAG with FAISS, HyDE-augmented retrieval, Microsoft's GraphRAG, NebulaGraph RAG, and a hybrid architecture combining both paradigms. Results show that GraphRAG outperforms VectorRAG by 12–23 percentage points on multi-hop and temporal tasks but is competitive only with VectorRAG on single-hop factoid tasks while being substantially more expensive to construct and maintain. The hybrid architecture achieves the best overall performance across all task types at intermediate cost. We release all evaluation code and experimental logs to facilitate replication.

Keywords: ***retrieval-augmented generation; knowledge graph; vector database; RAG; multi-hop QA; LLM; information retrieval***

1. Introduction

The decision to use a knowledge graph or a vector store as the retrieval backbone for a RAG system is one that practitioners face with increasing frequency, and it is one they typically make without strong empirical guidance. The two approaches rest on fundamentally different assumptions about knowledge structure. Vector retrieval treats knowledge as a collection of semantically comparable chunks and exploits the surprising effectiveness of dense embedding similarity for surface-level relevance. Graph retrieval treats knowledge as a structured network of entities and relationships and exploits the explicit relational structure that knowledge bases encode.

Neither assumption is universally better. But the choice matters enormously for specific tasks. A

user asking 'what year was the Eiffel Tower built?' is asking a question for which both approaches will likely produce the correct context. A user asking 'which companies that were subsidiaries of conglomerates acquired by Amazon between 2018 and 2022 are now defunct?' is asking a question that requires multi-hop traversal of a relational structure — and for that question, the two approaches diverge sharply in their capacity to retrieve the right context.

1.1 Scope and Contributions

We compare five RAG implementations across four task types and three corpora, with the specific goal of identifying which retrieval paradigm (or combination) is best suited to which task and corpus characteristics. We contribute: (1) a controlled evaluation framework with standardised corpora across four task types; (2) detailed performance, latency, and construction-cost measurements for five retrieval implementations; (3) a practical decision framework for retrieval architecture selection.

2. Systems Evaluated

VectorRAG implementations: Naive RAG uses FAISS with OpenAI text-embedding-3-small and GPT-4o as the generation model. HyDE augments this with hypothetical document embeddings, generating a hypothetical answer to the query and using that for retrieval before generating the final answer. GraphRAG implementations: Microsoft's GraphRAG constructs a community-detection-based graph from the source corpus and performs multi-level graph traversal. NebulaGraph RAG uses a Cypher-query interface over a pre-constructed knowledge graph. Hybrid: our hybrid system routes queries to either vector or graph retrieval based on a lightweight query classifier, then merges retrieved contexts before generation.

3. Experimental Design

We evaluate over three corpora: WikiMultiHop (general domain, low to moderate KG density), MedQA-KG (biomedical, high KG density), and FinancialKG (financial documents, structured but sparse KG). Task types are single-hop factoid QA, multi-hop reasoning (two or three hops), entity disambiguation, and temporal reasoning. All models use the same generation backbone (GPT-4o) and the same prompt templates to isolate retrieval differences from generation differences.

Table 1. *EM (Exact Match) scores across task types and retrieval architectures. Best result per row in bold.*

Task / Corpus	Naive RAG	HyDE RAG	GraphRAG	NebulaRAG	Hybrid
Single-hop (Wiki)	73.4	76.1	71.8	70.2	77.0
Multi-hop (Wiki)	44.2	46.8	61.3	58.9	65.1
Single-hop (Med)	68.9	71.3	70.5	71.8	74.2
Multi-hop (Med)	38.1	40.6	60.7	61.4	65.8
Temporal (Finance)	52.3	54.0	69.1	67.3	71.4
Disambiguation (Finance)	61.7	63.2	72.8	70.4	73.9

EM = Exact Match. All values are percentages. Scores averaged over three runs. Hybrid architecture uses a learned query router trained on 500 query-type examples.

4. Results and Analysis

The pattern in Table 1 is consistent across corpora and task types: graph-based retrieval approaches substantially outperform vector-based approaches on multi-hop and temporal tasks, while the advantage on single-hop factoid tasks is negligible or reversed. This aligns with the theoretical intuition about task type and knowledge structure alignment, but the magnitude of the gap — 12–23 percentage points on multi-hop tasks — is larger than we expected given recent improvements in dense retrieval.

The cost picture complicates the choice. GraphRAG requires a knowledge graph construction step that takes approximately 8 hours on the WikiMultiHop corpus (NVIDIA A100, 80GB) and produces a graph with 1.2M nodes and 4.1M edges. For frequently updated corpora — financial documents, news, product catalogues — this construction cost is prohibitive without an incremental update strategy. NebulaGraph's Cypher interface is faster to query but requires pre-constructed graph schemas that may not generalise to new domains. The hybrid system's query router is trained quickly (2 hours, 500 examples) and adds approximately 15ms per query latency relative to naive RAG.

[Figure 1 — Performance vs. construction cost scatter plot for five retrieval architectures. X-axis]

Figure 1. *Performance vs. construction cost scatter plot for five retrieval architectures. X-axis: knowledge graph construction time in GPU-hours. Y-axis: average EM score across all task types. The Pareto frontier is traced in blue. The Hybrid system offers the best performance-per-construction-cost tradeoff across all evaluated configurations.*

5. Practical Recommendations

For practitioners choosing a retrieval architecture, our empirical findings support the following heuristics. If your application is dominated by single-hop, factoid-style queries over a corpus that updates frequently, VectorRAG — ideally with HyDE augmentation — offers the best combination of performance and operational simplicity. If your application involves multi-hop reasoning, temporal queries, or entity-relationship navigation over a relatively stable knowledge base, GraphRAG's construction cost is justified by its performance advantage. For general-purpose applications spanning multiple query types, the hybrid architecture with a lightweight query router offers the best performance across task types at modest additional implementation complexity.

6. Conclusion

RAG is not a monolithic technique but a family of approaches whose relative merits depend critically on query type and knowledge base structure. Our empirical comparison provides quantitative guidance for architecture selection that we hope moves the conversation beyond 'RAG vs. no RAG' toward the more operationally useful question of which RAG architecture for which task. All evaluation code is available at https://github.com/datamind-papers/rag-comparison.

References

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459–9474. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., & Larson, J. (2024). From local to global: A graph RAG approach to query-focused summarization. arXiv preprint arXiv:2404.16130. https://doi.org/10.48550/arXiv.2404.16130

Gao, L., Ma, X., Lin, J., & Callan, J. (2023). Precise zero-shot dense retrieval without relevance labels. ACL 2023, 1762–1777. https://doi.org/10.18653/v1/2023.acl-long.99

Trivedi, H., Balasubramanian, N., Khot, T., & Sabharwal, A. (2022). MuSiQue: Multihop questions via single-hop question composition. Transactions of the Association for Computational Linguistics, 10, 539–554. https://doi.org/10.1162/tacl_a_00475

Sun, J., Xu, C., Tang, L., Wang, S., Lin, C., Gong, Y., & Gao, J. (2023). Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. arXiv preprint arXiv:2307.07697. https://doi.org/10.48550/arXiv.2307.07697

Izacard, G., & Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. EACL 2021, 874–880. https://doi.org/10.18653/v1/2021.eacl-main.74

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., & Yih, W. T. (2020). Dense passage retrieval for open-domain question answering. EMNLP 2020, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. NeurIPS 2013, 2787–2795. https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html

Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., & Wu, X. (2024). Unifying large language models and knowledge graphs: A roadmap. IEEE Transactions on Knowledge and Data Engineering. https://doi.org/10.1109/TKDE.2024.3352100

Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., & Wen, J. R. (2024). Retrieval-augmented generation for AI-generated content: A survey. arXiv preprint arXiv:2402.19473. https://doi.org/10.48550/arXiv.2402.19473