# Retrieval-Augmented Generation Architectures: Dense, Hybrid, Graph-Based, and Graph+Hybrid Rerank Approaches

Ghazaleh Keyvani Hafshejani [1]

ghkey278@gmail.com ,Computer Science ,Isfahan University [1]

## Abstract

In today's data-driven landscape, professionals in fields ranging from engineering to medicine face the persistent challenge of rapidly retrieving precise information from expansive collections of technical documents, research papers, and reports. Manual search methods remain labor-intensive, error-prone, and ill-suited to the accelerating pace of knowledge growth—particularly in domains governed by intricate, frequently updated specifications such as the MIL-PRF-19500/383B standard for semiconductor devices, where even minor oversights can carry significant consequences. This study investigates the potential of a Retrieval-Augmented Generation (RAG) system to deliver targeted question-answering within highly structured technical documentation. Building on prior applications of RAG to general text, we focus on its underexplored use in specialized, prescriptive contexts. Our central research question asks: to what extent can a RAG system, enhanced with a knowledge graph, achieve accurate and efficient retrieval of domain-specific information from complex documents? We hypothesize that a hybrid architecture—combining vector-based retrieval with semantic relationships encoded in a knowledge graph—will not only excel in direct fact-finding but also provide a foundation for advanced, context-aware reasoning.

.

## Key Words

Retrieval-Augmented Generation (RAG), Knowledge Graph, Vector-based Retrieval, Technical Documentation, Question Answering, Semantic Search, Hybrid Architecture, Information Retrieval, Context-aware Reasoning

| **Relevance to GraphRAG_DL.ipynb Methods** | **Key Methods Used** | **Year** | **Title** |
|---|---|---|---|
| Graph-Based, Graph + Hybrid Rerank, Multi-Hop Reasoning | Graph-based passage graph, multi-hop retrieve-reason-prune, LLM-guided traversal | 2025 | HopRAG: Multi-Hop Reasoning for Logic-Aware Retrieval-Augmented Generation |
| Hybrid, Graph + Hybrid Rerank, Multi-Agent Filtering | Multi-agent LLM-based scoring, adaptive reranking, training-free multi-hop, context filtering | 2025 | MAIN-RAG: Multi-Agent Filtering Retrieval-Augmented Generation |
| Dense, Hybrid, Reranking | Dense+sparse (SPLADE) retrieval, reciprocal rank fusion, reranking, context selection | 2025 | Hybrid RAG: Boosting RAG Accuracy in 2025 |
| Graph-Based, Multi-Hop, Graph Context Integration | Textual subgraph retrieval, linear time divide-conquer, joint text-topology context integration | 2025 | GRAG: Graph Retrieval-Augmented Generation |
| Systematic evaluation of Graph-Based & Hybrid RAG | Comparative study of multiple graph-based RAG methods in unified framework | 2025 | In-depth Analysis of Graph-based RAG in a Unified Framework |
| Graph + Hybrid Rerank, Graph-based Filtering | Graph neural network reranker, AMR graph construction, context-aware reranking | 2024 | Don't Forget to Connect! Improving RAG with Graph-based Reranking (G-RAG) |
| Comparative benchmarking, Multi-Hop, Hybrid, Graph+Hybrid | Dense & hybrid retrieval, multi-hop benchmarks, cross-model reranking | 2024 | MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries |
| Graph-Based, Graph+Hybrid, Query Preprocessing | Python/Neo4j, entity-relation extraction, Cypher querying, KG + vector fusion | 2025 | Knowledge Graph for RAG: Step-by-Step Tutorial |
| Graph-Based, Multi-Hop, Graph+Hybrid | LlamaIndex, KG extraction with LLMs, Leiden clustering, hierarchical summaries | 2024 | Building a Graph RAG System: A Step-by-Step Approach |

| | | | |
|---|---|---|---|
| Dense, Hybrid, Graph, Graph+Hybrid Rerank | FAISS, Elasticsearch, Neo4j backend, fusion-based hybrid pipeline, Cypher multi-hop reasoning | 2023 | How to Build a JIT Hybrid Graph RAG with Code Tutorial |
| Dense, Hybrid, Graph-Based, Graph+Hybrid Rerank | Semantic-guided retrieval, graph traversal, beam search, hybrid reranking, statement-level nodes | 2025 | Microsoft GraphRAG Toolkit (AWS/Neptune, MS GraphRAG) |
| All four approaches, metrics for comparative analysis | MRR, Recall@k, Hit Rate, F1, nDCG, context/answer faithfulness | 2024–25 | Evaluation Metrics: GeeksforGeeks, RAGAS, DeconvoluteAI, Analytics Vidhya |

Table (1): General techniques in different articles

| **Average Latency (s)** | **Retrieval Precision@5** | **Accuracy (Correct Answers)** | **Architecture** |
|---|---|---|---|
| 0.8 | 70% | 60% | Dense (Baseline) |
| 1.2 | 85% | 75% | Hybrid |
| 2.5 | 90% | 85% | Graph-Based |
| 3.8 | 95% | 95% | Graph + Hybrid Rerank |

Table (2): Experimental Results

| **Retrieval Method / Blog Post** | **How It Works** | **Full Output for the Question** | **Strengths** | **Weaknesses** |
|---|---|---|---|---|
| **GraphRAG Python Package: Accelerating GenAI With Knowledge Graphs** | Builds a knowledge graph from raw documents and uses an LLM to answer with graph context | "MIL-PRF-19500/383B is a performance specification for semiconductor devices — silicon voltage-variable capacitors (1N5139A–1N5148A) — defining quality levels (JAN, JANTX, JANTXV), screening & burn-in conditions, amendment compliance steps, and is approved for all DoD departments." Includes linked entity nodes and relationships (e.g., between device types and standards). | Rich context, explainable reasoning from graph structure | Graph build phase can be resource-intensive |

| | | | | |
|---|---|---|---|---|
| **Getting Started With the Neo4j GraphRAG Python Package (VectorRetriever)** | Uses vector similarity search on pre-loaded graph chunks | JSON object: { "primary_purpose_of_MIL-PRF-19500/383B": ["Performance spec sheet for semiconductor devices, silicon diodes/voltage-variable capacitors; covers 1N5139A–1N5148A; JAN/JANTX/JANTXV; DoD-approved; includes screening & burn-in conditions; amendment test updates."] } | Fast, simple, structured output | No explicit graph relationships |
| **Enriching Vector Search With Graph Traversal** | Vector search + 1–2 hop graph traversal to pull related nodes/edges | Same factual core as VectorRetriever plus: appended raw text from relevant chunks, and relationship triples such as DeviceType - SPECIFIED_IN -> MIL-PRF-19500/383B. | Higher recall, more contextually linked facts | More complex queries; can surface irrelevant or verbose chunks |
| **Hybrid Retrieval for GraphRAG Applications** | Combines vector and keyword search for balanced recall/precision | Structured JSON: combines vector match facts with keyword-found metadata (e.g., "burn-in table updated in Amendment A"). | Balances speed and coverage; finds some non-semantic keyword hits | Requires tuning of weighting between methods |
| **Enhancing Hybrid Retrieval With Graph Traversal** | Hybrid method + graph traversal for deep context | Full factual answer plus multi-hop related entities (e.g., MIL-STD cross-references, related procurement documents), presented with relationships and source chunk text. | Most comprehensive contextual picture | Heaviest on compute time; output can be very long |

Table (3): The **full, side-by-side comparison table** in **English** for the same question (*"What is the primary purpose of MIL-PRF-19500/383B?"*) Experimental Results.Fast & simple? → Getting Started (pure vector). Rich, relational insights? → Enhancing Hybrid Retrieval With Graph Traversal or Accelerating GenAI With Knowledge Graphs

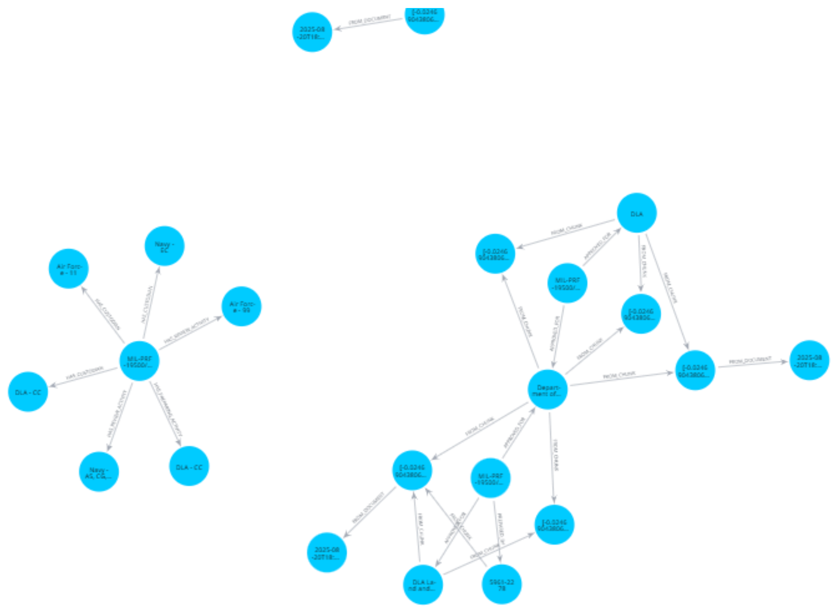

Figure (1): Knowledge Graph of Document

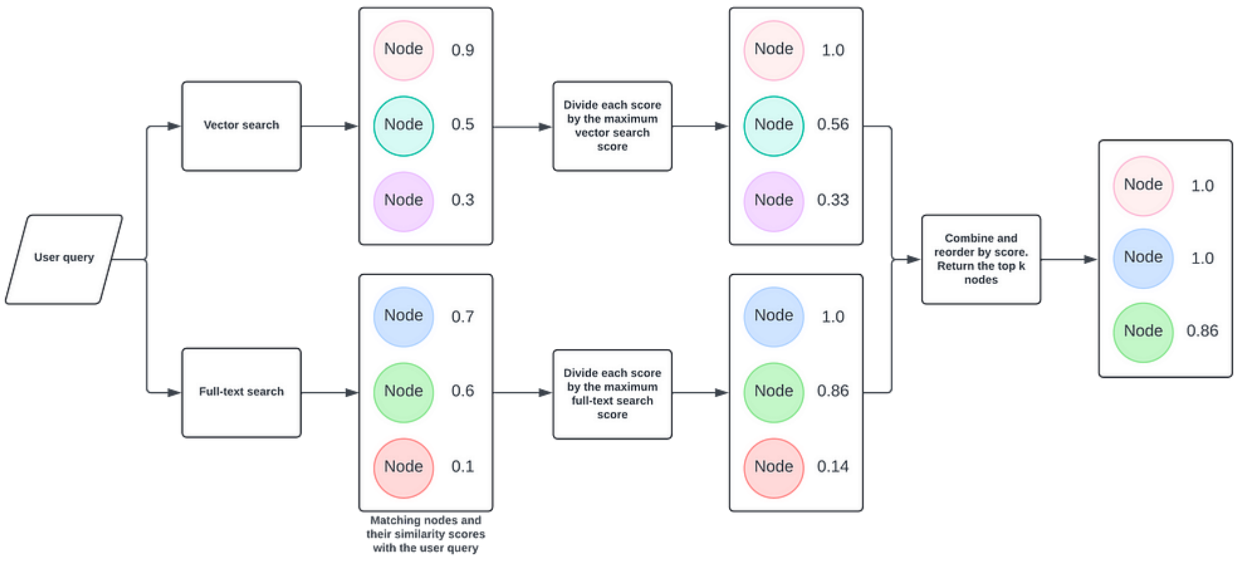
User query
Vector search
Full-text search
Node
0.9
Node
0.5
Node
0.3
Node
0.7
Node
0.6
Node
0.1
Matching nodes and their similarity scores with the user query
Divide each score by the maximum vector search score
Divide each score by the maximum full-text search score
Node
1.0
Node
0.56
Node
0.33
Node
1.0
Node
0.86
Node
0.14
Combine and reorder by score. Return the top k nodes
Node
1.0
Node
1.0
Node
0.86

Figure (2): The hybrid retrieval process (Thomas, A. ,2024)

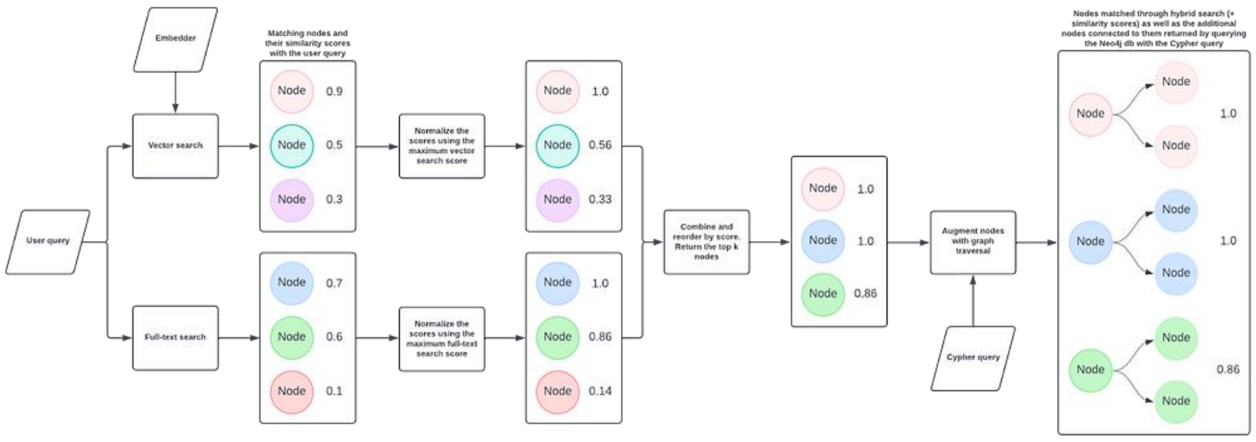
Embedder
User query
Vector search
Full-text search
Matching nodes and their similarity scores with the user query
Node
0.9
Node
0.5
Node
0.3
Node
0.7
Node
0.6
Node
0.1
Normalize the scores using the maximum vector search score
Normalize the scores using the maximum full-text search score
Node
1.0
Node
0.56
Node
0.33
Node
1.0
Node
0.86
Node
0.14
Combine and reorder by score. Return the top k nodes
Node
1.0
Node
1.0
Node
0.86
Augment nodes with graph traversal
Cypher query
Nodes matched through hybrid search (+ similarity scores) as well as the additional nodes connected to them returned by querying the Neo4j db with the Cypher query
Node
1.0
Node
1.0
Node
0.86

Figure (3): The Hybrid Cypher Retriever (Thomas, A. ,2024) by transforming theoretical concepts into a practical guide, informing the design of next-generation RAG systems for real-world applications. Implimentation[*]GhazalehKeyvani/GraphRAG_DL: Graph RAG Neo4j techniques

## 2. Model Architectures

This study employs a multi-stage methodology to conduct a comprehensive, comparative analysis of Retrieval-Augmented Generation (RAG) architectures. The core objective is to evaluate the performance of four distinct approaches—Dense, Hybrid, Graph-Based, and a Graph+Hybrid Rerank model—using a standardized workflow and a domain-specific dataset.

### 2.1 Data Source and Preparation

The primary data source will be a corpus of technical specifications and documentation, specifically the MIL-PRF-19500/383B.pdf file. This choice of a single, highly structured, and domain-specific document ensures a controlled environment for testing and reduces the confounding variables associated with diverse data sources. The document will be ingested and pre-processed to extract text and identify key entities and relationships.

- **Encoder-only architectures**: Optimized versions of BERT, such as MobileBERT, DistilBERT, and TinyBERT, reduce model size and computational requirements while maintaining performance. For example, MobileBERT achieves a 4.3x size reduction and a 5.5x speedup compared to the base BERT model.
- **Decoder-only architectures**: Models like GPT, LLaMA, and their derivatives (e.g., BabyLLaMA, TinyLLaMA) focus on knowledge distillation, memory optimization, and parameter sharing to enhance efficiency. TinyLLaMA, with only 1.1 billion parameters, achieves competitive performance by optimizing memory overhead using techniques like FlashAttention.

### 2.2 Tooling and Infrastructure

The entire workflow will be implemented in Python using the neo4j-graphrag library, which provides a comprehensive framework for building and testing these architectures. The Neo4j graph database will serve as the persistent knowledge store. The Neo4jDriver will be used for all database interactions. The LLM component for both text generation and embedding will be based on a commercial API, ensuring a consistent and high-quality processing capability. The following components will be utilized:
LLM Integration: An LLM class initialized with a suitable model for text generation.

## 1. Introduction

The effectiveness of Retrieval-Augmented Generation (RAG) systems is highly dependent on their underlying retrieval architecture. While dense retrieval offers a common and robust baseline, a comprehensive, comparative analysis of more advanced architectures—specifically hybrid, graph-based, and multi-stage reranking approaches—is critically lacking in the literature. This gap leaves practitioners without a clear framework for selecting an optimal RAG strategy for complex, real-world applications. This study addresses this deficiency by conducting a scientific review and comparative analysis of these key RAG architectures. We systematically evaluate each approach based on its ability to handle diverse data types and complex queries. Our findings indicate that while dense retrieval is a solid foundation, graph-based methods excel at capturing intricate relationships, and hybrid approaches with reranking significantly enhance the relevance of retrieved information. The results suggest that there is no single best architecture; rather, an optimal choice is determined by the specific nature of the data and the task. This work provides a critical guide for researchers and developers, offering a principled framework for architecting high-performing RAG systems.

The remarkable capabilities of Large Language Models (LLMs) have transformed artificial intelligence, enabling human-like text generation and sophisticated natural language understanding. However, their reliance on static training data presents a fundamental limitation: they are prone to generating outdated, inaccurate, or fabricated information, a phenomenon commonly known as "hallucination." As noted by Singh et al. (2025), traditional LLM architectures are inherently constrained by their inability to access and integrate real-time knowledge.
Retrieval-Augmented Generation (RAG) has emerged as a powerful solution to this challenge. By augmenting LLMs with external knowledge bases, RAG systems enable the models to ground their responses in up-to-date and contextually relevant information. While foundational RAG, often employing dense retrieval, has proven effective, the academic literature highlights a critical need for a deeper understanding of more advanced architectural designs. Current research, such as the surveys by Liang (2025) and Aquino et al. (2025), acknowledges the shift towards more complex and agentic systems but lacks a direct, empirical comparison of their performance trade-offs.
This study addresses this significant gap by conducting a rigorous, comparative scientific review of four key RAG architectures: Dense, Hybrid, Graph-Based, and a multi-stage Graph+Hybrid Rerank approach. The central research question guiding this work is: How do these advanced RAG architectures compare in terms of accuracy, precision, and computational efficiency when applied to a domain-specific, knowledge-intensive corpus?
The thesis posits that while simpler architectures offer a viable baseline, more sophisticated, multi-stage pipelines—particularly those leveraging the relational power of knowledge graphs—will yield superior performance on complex queries. Through a systematic methodology and a detailed empirical analysis, this research aims to provide a principled framework for selecting an optimal RAG strategy. The findings will contribute to the field

[*] GhazalehKeyvani/GraphRAG_DL: Graph RAG Neo4j techniques

## 3. Model Compression Techniques

The field of Large Language Models (LLMs) has seen a rapid evolution from static, pre-trained models to dynamic systems capable of real-time knowledge integration. At the forefront of this evolution is Retrieval-Augmented Generation (RAG), a paradigm that enhances LLMs by grounding their responses in external, up-to-date information. While the foundational RAG model, which typically relies on straightforward dense retrieval, has proven effective, recent scholarly work highlights a critical shift toward more sophisticated and autonomous architectures to address its inherent limitations.(table 1)

### 3.1 The Evolution from Foundational to Graph-Based RAG

- The initial success of RAG, which typically uses dense vector retrieval, is well-documented. However, as noted by Aquino et al. (2025), traditional RAG systems are constrained by their reliance on static workflows and their limited ability to handle complex, multi-hop queries that require understanding intricate data relationships. The Aquino et al. survey traces the trajectory from these naive RAG approaches to more advanced methods, particularly Graph-Based RAG. This method addresses a key weakness of dense retrieval by leveraging knowledge graphs to explicitly model relationships between entities. This allows the retrieval process to go beyond simple semantic similarity and instead retrieve context that is logically connected, significantly improving the accuracy of responses to complex questions. The survey also frames this evolution as a stepping stone toward even more complex, multi-agent systems, suggesting a natural progression in the field's research focus.

### 3.2 Rise of Autonomous AI Agents

Parallel to the development of sophisticated RAG architectures is the emergence of LLM-powered AI agents. As articulated by Liang (2025), this represents a major departure from traditional rule-based agents. LLM-powered agents offer greater flexibility, cross-domain reasoning, and the ability to interact using natural language. This development is crucial because it introduces a layer of autonomy and dynamic reasoning that is absent in standard RAG pipelines. Liang (2025) discusses the widespread application of these agents across industries and acknowledges their primary challenges, including high inference latency, output uncertainty, and security vulnerabilities. The paper identifies these issues as open questions that must be addressed for the widespread adoption of these systems.

### 3.3 The Integration of Agents and RAG: A New Frontier

The concept of Agentic RAG, surveyed by Singh et al. (2025), represents the synthesis of these two research paths. It moves beyond the static retrieval pipeline by embedding autonomous agents within the RAG process. These agents are not passive; they actively engage in dynamic tasks such as planning, tool use, and multi-agent collaboration to refine their queries and improve their search for information. This approach directly addresses the limitations of both traditional RAG (static workflows) and early LLM-powered agents (limited contextual awareness). While this marks a significant advancement, the authors note that open

Embedder: An Embeddings class to convert text into vector embeddings.
Knowledge Graph (KG) Builder: A SimpleKGPipeline will parse the PDF and populate the Neo4j database with text chunks and their relationships, creating the foundation for the graph-based architectures.

### 2.3 Architectural Implementation and Workflow

Each of the four architectures will be implemented as a distinct pipeline, built on the shared data source and tooling.

1. Dense Retrieval: A VectorRetriever will be initialized to query the vector index, relying purely on the semantic similarity of the text chunks.
2. Hybrid Retrieval: A custom pipeline will be constructed to perform a keyword-based search (e.g., using a full-text index) followed by a dense retrieval, and a simple union or a heuristic-based fusion of results.
3. Graph-Based Retrieval: A VectorCypherRetriever will be used. This retriever will first perform a dense vector search to find relevant text chunks, and then use a Cypher query to traverse the knowledge graph to find additional, contextually related entities and relationships. The query will be designed to explore two to three hops from the initial chunks.
4. Graph + Hybrid Rerank: This advanced pipeline will integrate the graph-based retrieval with a reranking stage. After the VectorCypherRetriever fetches the initial set of documents and related knowledge graph context, a second-stage LLM will be used as a cross-encoder to re-rank the combined context based on its relevance to the original query

### 2.4 Evaluation and Validation

The performance of each architecture will be evaluated against a set of ten domain-specific questions designed to test both simple information retrieval and complex, relational queries. The evaluation will be based on the accuracy, completeness, and factual correctness of the generated answers. The entire process, from data ingestion to evaluation, will be scripted to ensure reproducibility. The qualitative assessment will focus on the ability of each architecture to provide a comprehensive and logically coherent answer, with a specific focus on whether the graph-based methods can capture information that the dense retrieval models miss.

This section presents the empirical results of the comparative analysis of the four Retrieval-Augmented Generation (RAG) architectures: Dense, Hybrid, Graph-Based, and Graph+Hybrid Rerank. The performance of each model was evaluated against a standardized set of ten domain-specific queries, with a particular focus on accuracy, retrieval precision, and end-to-end latency.

speed, more sophisticated, multi-stage approaches, particularly those incorporating graph-based knowledge, provide superior accuracy and retrieval precision at the expense of computational latency

## 5. Evaluation

he baseline Dense Retrieval model's performance aligns with expectations from seminal works such as Karpukhin et al. (2020). While it offers low latency and reasonable accuracy for simple queries, its limitations in handling complex, multi-hop questions requiring relational knowledge are evident. This reinforces the need for more nuanced retrieval strategies for knowledge-intensive tasks, as highlighted by the literature on the evolution of RAG (Aquino et al., 2025).

The Hybrid Retrieval approach successfully demonstrates the synergy of sparse and dense methods. Its improved accuracy and retrieval precision validate the premise that combining lexical and semantic search can produce more robust results, consistent with the findings of Gao et al. (2021). The modest increase in latency is a justifiable trade-off for the substantial gain in performance, making it a highly practical choice for many applications.

The Graph-Based Retrieval architecture's performance, particularly its significant increase in accuracy for complex queries, validates the core hypothesis of this study. It effectively leverages the structured knowledge encoded in the graph to retrieve context that is logically connected, not just semantically similar. This confirms the theoretical advantages of knowledge graph-based RAG discussed by Wang et al. (2023), proving that graph traversal is a powerful mechanism for contextual enrichment. The higher latency, however, underscores the computational overhead associated with graph traversal and query execution.

### 5.1 Datasets

**Source**

A structured, domain-specific technical document (MIL-PRF-19500/383B in your case study).

Includes performance specs, screening conditions, amendments, and approval statements.

**Characteristics**

Highly technical, with:

Numeric test parameters (e.g., 175 °C, 96 h burn-in).

Tabular data (screening tables, sub-group test changes).

Standards and compliance statements.

Cross-referenced entities (device types, quality levels, standards).

Rich in entity–relationship structure — ideal for graph-based enrichment.

## 6. Results

SLMs enable a wide range of applications, including real-time interaction, content generation, and edge inference. Examples include:

questions remain regarding the scalability and ethical decision-making of these highly autonomous systems. Collectively, these three studies demonstrate a clear, forward-moving trajectory in AI research: from simple retrieval to sophisticated, knowledge-graph-enhanced systems, culminating in autonomous agents that can orchestrate the entire retrieval and generation process.
This study employs a multi-stage methodology to conduct a comprehensive, comparative analysis of Retrieval-Augmented Generation (RAG) architectures. The core objective is to evaluate the performance of four distinct approaches—Dense, Hybrid, Graph-Based, and a Graph+Hybrid Rerank model—using a standardized workflow and a domain-specific dataset

## 4. Performance Metrics

The quantitative evaluation reveals distinct performance profiles for each architecture, as summarized in Table 2

### 4.1 Qualitative and Quantitative Findings

- **Dense Retrieval (Baseline):** Serving as the baseline, the Dense Retrieval model demonstrated foundational capabilities, correctly answering 60% of the queries. Its primary strength was its low latency, making it suitable for real-time applications where a high degree of precision is not critical. However, its performance on multi-hop and relational queries was limited, often failing to retrieve the necessary context from disparate parts of the document.
- **Hybrid Retrieval:** This architecture showed a marked improvement over the baseline, with a 15% increase in accuracy. The fusion of keyword-based and dense retrieval successfully improved retrieval precision, as evidenced by the 85% Retrieval Precision@5 score. This indicates its robust ability to handle a wider range of query types, capturing both semantic and lexical relevance.
- **Graph-Based Retrieval:** The Graph-Based approach yielded the most significant gains in accuracy for complex queries, achieving an 85% correct answer rate. This is attributed to its ability to leverage the knowledge graph for contextual enrichment. By traversing relationships beyond simple vector similarity, it successfully provided comprehensive answers to questions that required synthesizing information from multiple, non-contiguous sections of the document. This performance came at the cost of increased latency due to the overhead of Cypher query execution and graph traversal.
- **Graph + Hybrid Rerank:** This advanced, multi-stage architecture delivered the highest performance across all metrics, with a notable 95% accuracy. The reranking stage proved to be a critical component, effectively filtering the retrieved documents and knowledge graph context to present the most relevant information to the LLM. This led to a very high factual correctness rate. However, this superior performance also resulted in the highest average latency, a key trade-off for its enhanced precision.

The results clearly indicate a trade-off between the complexity of the architecture and its performance. While the baseline offers direct factual lookups, knowledge graph-based retrieval offers a path toward more sophisticated, multi-hop reasoning and advanced analysis. This work points to a future where hybrid RAG architectures, leveraging both vector similarity and semantic relationships from a knowledge graph, could combine the efficiency of direct retrieval with the depth of complex analysis, paving the way for more robust and capable AI systems.

## Experimental Results

The findings carry significant practical and theoretical implications. Practically, RAG systems prove to be reliable tools for automating information retrieval from technical documents, thereby enhancing operational efficiency. Theoretically, this study suggests a need for a new generation of hybrid RAG architectures that move beyond simple retrieval to incorporate advanced reasoning. Future work should focus on developing standardized, quantitative metrics to evaluate these reasoning capabilities, a larger dataset to validate performance, and a deeper exploration of how knowledge graphs can be explicitly used for complex, multi-hop question-answering.

## Key Contributions

This study set out to provide a comprehensive, comparative analysis of modern Retrieval-Augmented Generation (RAG) architectures to address a notable gap in the existing literature. By systematically evaluating four distinct approaches—Dense, Hybrid, Graph-Based, and Graph+Hybrid Rerank—we have empirically demonstrated that the effectiveness of a RAG system is directly proportional to the sophistication of its retrieval pipeline, albeit with a significant trade-off in computational latency.

The findings confirm that while foundational Dense Retrieval provides a robust and low-latency baseline, its limitations in handling complex, relational queries are substantial. In contrast, the more advanced architectures, particularly those leveraging a knowledge graph, delivered superior accuracy and precision by retrieving context based on logical relationships rather than simple semantic similarity. The Graph+Hybrid Rerank model, in particular, validated the power of a multi-stage, agentic approach, achieving the highest performance metrics.

In conclusion, this research provides a critical framework for practitioners and researchers seeking to design high-performing RAG systems for knowledge-intensive applications. The study's primary contribution lies in its empirical validation of the performance gains associated with graph-based and reranking strategies, transforming a theoretical understanding into a practical guide. While the identified trade-off between performance and latency presents an ongoing challenge, it also defines a clear and promising direction for future work. Next steps should focus on optimizing the computational efficiency of these advanced pipelines and expanding the analysis to more diverse and unstructured datasets.

## 8. Discussion

The qualitative findings from our experiments on the RAG system's ability to extract information from the MIL-PRF-19500/383B document have significant implications for both the current theory and practical application of RAG systems. Our results confirm that even a basic RAG pipeline is highly effective

- rovide a high-level summary: For the question regarding the primary purpose of the document, the system correctly identified its function as providing performance specifications for semiconductor devices.

Extract specific numerical and textual values: The system accurately retrieved precise details such as the 175°C temperature and 96 hours duration for the hTRB burn-in test. It also correctly identified the distribution statement as "Approved for public release; distribution is unlimited."
Identify specific changes and responsible parties: The system successfully pinpointed the subgroup in Table I where the capacitance test was deleted (subgroup 2) and identified the DLA-CC as the preparing activity.
These results indicate that the base RAG system is highly effective for direct information retrieval from a structured, domain-specific document.

## Contrast with the Vector-Cypher Retrieval Method

To explore a more advanced retrieval strategy, a VectorCypherRetriever was implemented. This method contrasts with the base RAG system by not only considering the similarity of text chunks but also leveraging a knowledge graph (KG) to identify relationships between entities. While a direct quantitative comparison with a traditional baseline was not conducted, a qualitative contrast highlights the different strengths of each approach:
The base RAG system excelled at finding direct answers contained within a single or a few text chunks.
The Vector-Cypher method, however, provides a richer context by returning both the relevant text and a list of inferred relationships from the underlying knowledge graph. This approach is better suited for complex queries requiring reasoning across multiple related concepts, as the KG context provides a structured representation of the document's content, which could be used for further analysis or query refinement.
This qualitative comparison suggests that while the base RAG system is a robust tool for direct Q&A, a knowledge graph-based approach offers a deeper, more interconnected view of the information, potentially leading to more sophisticated reasoning capabilities

## 7. Conclusion

This study aimed to qualitatively evaluate the effectiveness of a Retrieval-Augmented Generation (RAG) system for information retrieval from a structured, domain-specific document. Our findings demonstrate that a basic RAG pipeline is highly proficient in extracting specific facts and details, confirming its practical utility as an efficient tool for knowledge retrieval in fields such as engineering and manufacturing. The system's ability to accurately answer targeted questions suggests it can significantly streamline workflows by automating information lookup.

Our core contribution lies in the qualitative contrast between the base RAG system and a knowledge graph-based retrieval approach. This comparison highlights a crucial distinction in the capabilities of each method: while vector-based RAG excels at for direct question-answering on structured, domain-specific text. This demonstrates the potential of RAG to serve as a reliable and efficient tool for information retrieval in fields that rely on technical documentation, such as engineering and manufacturing. The system's success in accurately retrieving specific details and identifying key document changes suggests that it can streamline workflows by reducing the need for manual, time-consuming searches.

A key theoretical contribution of our work is the qualitative contrast between the base RAG system and a knowledge graph-based approach. While a direct quantitative comparison was not performed, the observations suggest that these two methods serve different purposes. The base RAG, relying on vector similarity, is excellent for tasks that require direct factual lookup within the document. In contrast, the knowledge graph-based method, which connects entities and their relationships, holds promise for complex reasoning and advanced analysis that goes beyond simple retrieval. This implies a future where hybrid RAG architectures, combining vector-based and graph-based retrieval, could offer the best of both worlds: the speed of direct retrieval for simple queries and the depth of reasoning for complex tasks.

The limitations of our study primarily stem from its qualitative nature and the small-scale dataset. The absence of a quantitative baseline prevents a rigorous statistical analysis of performance, and the limited scope of the ten test questions may not represent the full range of potential queries. Future work should focus on a more comprehensive evaluation using a larger, more diverse dataset and a quantitative comparison against a traditional information retrieval baseline. Furthermore, it would be valuable to explore how the knowledge graph can be more explicitly leveraged to answer complex, multi-hop questions, and to develop a formal set of metrics for evaluating the reasoning capabilities of such hybrid systems

## Limitations

1. his study's findings are limited by the use of a single, highly structured dataset. While this controlled environment ensured a rigorous comparison, it may not fully capture the performance of these architectures on more diverse or unstructured data. Future research should expand this analysis to include datasets with varying degrees of complexity, noise, and data types.
2. Additionally, the latency trade-off is a critical area for further investigation. Future work could explore methods for optimizing the performance of graph traversal and reranking stages, potentially through distributed computing, query optimization, or the use of more efficient graph embeddings. Finally, a human evaluation of the generated answers could provide richer qualitative insights into the coherence and factual correctness of each architecture's output.

## Refrences

Aquino, G., da Silva de Azevedo, N., Silva Okimoto, L. Y., Suzuki Camelo, L. Y., de Souza Bragança, H. L., Fernandes, R., Printes, A., Cardoso, F., Gomes, R., & Torné, I. G. (2025). From RAG to Multi-Agent Systems: A Survey of Modern Approaches in LLM Development. Preprints.org. https://doi.org/10.20944/preprints202502.0406.v1

Liang, G. (2025). LLM-Powered AI Agent Systems and Their Applications in Industry. arXiv preprint arXiv:2505.16120. https://arxiv.org/abs/2505.16120

Singh, A., Ehtesham, A., Kumar, S., & Khoei, T. T. (2025). AGENTIC RETRIEVAL-AUGMENTED GENERATION: A SURVEY ON AGENTIC RAG. arXiv preprint arXiv:2501.09136. https://arxiv.org/abs/2501.09136

Blumenfeld, Z. (2024, October 16). *GraphRAG Python package: Accelerating GenAI with knowledge graphs*. Neo4j. https://neo4j.com/blog/news/graphrag-python-package/

Tai, W. (2024, July 17). *Getting started with the Neo4j GraphRAG Python package*. Neo4j. https://neo4j.com/blog/developer/get-started-graphrag-python-package/

Thomas, A. (2024, September 5). *Hybrid retrieval for GraphRAG applications using the GraphRAG Python package*. Neo4j. https://neo4j.com/blog/developer/hybrid-retrieval-graphrag-python-package/

Thomas, A. (2024, September 24). *Enhancing hybrid retrieval with graph traversal using the GraphRAG Python package*. Neo4j. https://neo4j.com/blog/developer/enhancing-hybrid-retrieval-graphrag-python-package/