IMARA - Evaluation of Graph-Based RAG Approaches

*Machine Learning for Software Engineers (ML4SE)

Marco Allenspach
[Institution]
[Department]
City
, [Country]
ail

Lukas Koller
[Institution]
[Department]
City
, [Country]
ail

Emanuel Sovrano
[Institution]
[Department]
City
, [Country]
ail

***Abstract*—Retrieval-Augmented Generation (RAG) extends large language models with external knowledge, but purely vector-based implementations reach conceptual limits with complex, multi-step queries. In particular, explicit modeling of relationships between entities is missing. Graph-based RAG approaches address this deficit by representing knowledge in a structured and traversable manner.**

This work investigates graph-based RAG architectures through the IMARA project. Based on scientific PDFs, an end-to-end pipeline is built from document extraction through graph construction to evaluation. Classical RAG is compared with several GraphRAG variants. Evaluation is conducted reproducibly using OpenRAGBench and OpenRAG-Eval.

***Index Terms*—Retrieval-Augmented Generation, Knowledge Graphs, GraphRAG, LeanRAG, LinearRAG, GraphMERT, Document Processing, Evaluation**

I. Introduction

The IMARA project investigates the extent to which graph-based retrieval approaches can improve the quality of RAG systems for complex, knowledge-intensive questions. The focus is on scientific documents with high structural and semantic complexity.

The goal is a systematic comparison of a naive, purely vector-based RAG approach with various GraphRAG variants under controlled conditions. For this purpose, a reproducible pipeline is built that ranges from PDF extraction to evaluation.

A. Problem Statement

Naive RAG architectures are based on similarity search over text chunks. This approach has several weaknesses:

- Knowledge is fragmented and loses contextual relationships.
- Relationships between entities are implicit and not explicitly modeled.
- Multi-hop reasoning across multiple documents or sections is only possible to a limited extent.
- Quality strongly depends on the chunking strategy.

Scientific publications with cross-references, formal definitions, and dependencies are particularly inadequately accessible with this approach.

B. Project Objectives

From this problem statement, the objectives of IMARA are derived:

1) Building a graph-based RAG pipeline for creating dense knowledge graphs from scientific PDFs.
2) Systematic comparison of classical vector-based RAG approaches with various GraphRAG variants (including LeanRAG, LinearRAG, GraphMERT).
3) Development of a flexible, repeatable pipeline from PDF to evaluation, including data versioning (DVC), orchestration, and MLflow-supported traceability.
4) Use of OpenRAGBench/OpenRAG-Eval for objective, reproducible evaluation of different variants.

II. State of the Art

This chapter describes relevant work and concepts in the field of RAG and GraphRAG.

A. LeanRAG

LeanRAG is a graph-based RAG approach in which entities are grouped into semantically coherent clusters with explicitly modeled relations. These clusters form aggregated nodes that serve as a hierarchical navigation layer above the actual text passages (chunks). The approach aims to reduce the number of necessary retrieval steps while simultaneously increasing context relevance.

The core idea is semantic aggregation: Instead of searching purely in flat vector space, the system can first navigate through a structured graph and then retrieve the relevant text passages from the identified clusters. This combines the advantages of symbolic knowledge representation (explicit relations, structure) with the semantic flexibility of embedding-based search.

B. LinearRAG

LinearRAG structures the knowledge graph as a linear, temporally or causally ordered chain. This is particularly suitable for texts with implicit narrative or temporal structure, such as scientific papers (Introduction $\rightarrow$ Methods $\rightarrow$ Results $\rightarrow$ Discussion).

The approach uses TF-IDF weighting for passage-entity edges to quantify the semantic relevance of entities for specific text sections. The linear structure enables efficient traversal and reduces search complexity compared to general graph structures.

C. GraphMERT

GraphMERT (Graph Memory-Enhanced Retrieval and Transformation) is a neurosymbolic approach that combines the strengths of neural networks (learning, generalization) with symbolic AI (abstraction, logic, graph structures).

The system builds a hierarchical knowledge graph where entities and relations are not only extracted but also semantically consolidated and verified. GraphMERT uses a multi-stage pipeline:

1) Extraction of entities and relations from text
2) Consolidation and deduplication
3) Verification and quality assurance
4) Hierarchical structuring

This creates a more reliable and consistent knowledge base than with purely neural extraction methods.

D. OpenRAGBench and OpenRAG-Eval

OpenRAGBench is a reference dataset with approximately 1000 scientific PDFs from arXiv and associated question-answer pairs. It serves as a standardized benchmark for evaluating RAG systems.

OpenRAG-Eval is an evaluation framework that compares different RAG and GraphRAG systems based on uniform metrics:

- LLM Accuracy: Quality of generated answers
- Contain Accuracy: Completeness of information
- Faithfulness: Fidelity to source documents
- Runtime: Efficiency of the system

The combination of both tools enables objective, reproducible evaluation of different approaches.

III. Methodology

This chapter describes the implementation of the IMARA pipeline, from document processing to evaluation.

A. Overview of the Pipeline

The IMARA pipeline consists of several modular components that build on each other:

1) Document Processing (Docling)
2) Knowledge Graph Construction
3) RAG/GraphRAG Implementation
4) Evaluation (OpenRAG-Eval)
5) Orchestration and Versioning (DVC, MLflow)

Each component is designed to be independently executable and testable. Data versioning with DVC ensures reproducibility, while MLflow tracks experiments and parameters.

B. Document Processing with Docling

Docling is an open-source toolkit for AI-driven document conversion, specifically optimized for scientific PDFs. It offers several advantages over conventional extraction tools:

- Preservation of document structure (headings, sections, lists)
- Recognition and extraction of tables and figures
- Support for mathematical formulas
- Robust handling of multi-column layouts
- Export to various formats (Markdown, JSON, DoclingDocument)

The Docling pipeline consists of several stages:

1) PDF Parsing: Extraction of text, images, and metadata
2) Layout Analysis: Recognition of document structure
3) Table Detection: Identification and extraction of tables
4) OCR (optional): Text recognition in images
5) Export: Conversion to target format

In the IMARA project, Docling is used to convert scientific PDFs to structured Markdown, which serves as the basis for knowledge graph construction.

C. Knowledge Graph Construction

The construction of the knowledge graph is done in several steps:

1) Entity Extraction: Entities (persons, organizations, concepts, methods) are extracted from the processed documents using Named Entity Recognition (NER) and custom extraction rules. For scientific texts, domain-specific entity types are particularly important (e.g., algorithms, datasets, metrics).

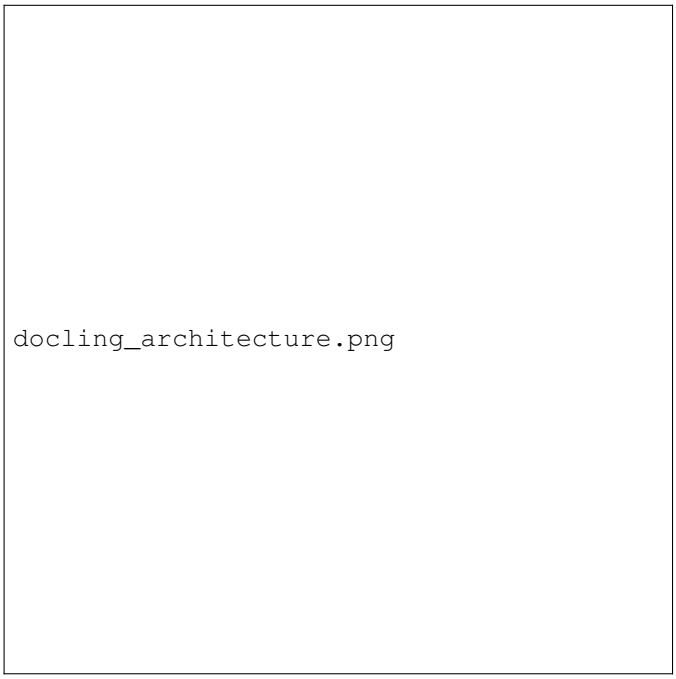

Fig. 1. Docling Architecture - Modular pipeline for document processing. Source: Docling Project [2]

2) Relation Extraction: Relations between entities are identified using pattern-based methods and neural models. Typical relation types in scientific texts are:

- uses (algorithm uses dataset)
- evaluates (paper evaluates method)
- improves (method improves baseline)
- cites (paper cites paper)

3) Graph Construction: The extracted entities and relations are transformed into a graph structure. Depending on the approach (LeanRAG, LinearRAG, GraphMERT), different construction strategies are applied:

LeanRAG: Entities are grouped into semantic clusters. Clusters are connected via aggregated relations. This creates a hierarchical structure with two levels: cluster level (coarse navigation) and entity level (fine-grained retrieval).

LinearRAG: Entities are arranged in a linear order that reflects the document structure. Edges are weighted with TF-IDF scores to represent the relevance of entities for specific text passages.

GraphMERT: A multi-stage pipeline is applied: extraction, consolidation, verification, hierarchical structuring. This creates a more reliable and consistent graph.

D. RAG/GraphRAG Implementation

1) Classical RAG (Baseline): The baseline is a naive, purely vector-based RAG system:

1) Documents are divided into chunks (e.g., 512 tokens)
2) Chunks are embedded (e.g., with Sentence-BERT)
3) For a query, the most similar chunks are retrieved (cosine similarity)
4) Retrieved chunks are passed to an LLM for answer generation

This approach serves as a reference for comparison with graph-based methods.

2) GraphRAG Variants: The GraphRAG variants extend the baseline with graph-based retrieval:

LeanRAG:

1) Query is embedded
2) Most relevant clusters are identified (cluster-level retrieval)
3) Within the clusters, most relevant entities are identified (entity-level retrieval)
4) Associated text passages are retrieved
5) Passages are passed to LLM for answer generation

LinearRAG:

1) Query is embedded
2) Most relevant entities are identified in the linear graph
3) Neighboring entities in the linear structure are considered (context)
4) Associated text passages are retrieved (weighted by TF-IDF)
5) Passages are passed to LLM for answer generation

GraphMERT:

1) Query is analyzed (entity and relation extraction)
2) Relevant subgraph is identified (graph traversal)
3) Entities and relations in the subgraph are consolidated
4) Associated text passages are retrieved
5) Passages and graph structure are passed to LLM for answer generation

E. Evaluation with OpenRAG-Eval

Evaluation is conducted using OpenRAG-Eval, which enables systematic comparison of different approaches. The evaluation process consists of several steps:

1) Test Dataset: OpenRAGBench provides approximately 1000 scientific PDFs and 3000 associated question-answer pairs. The questions cover various difficulty levels and question types:

- Factual questions (simple retrieval)
- Comparative questions (multi-document reasoning)
- Analytical questions (deep understanding)

2) Metrics: The following metrics are used for evaluation:

LLM Accuracy: An LLM (e.g., GPT-4) evaluates the quality of generated answers on a scale of 1-5. This measures semantic correctness and completeness.

Contain Accuracy: Measures whether the retrieved passages contain the information necessary to answer the question. This metric evaluates the quality of retrieval independently of generation.

Faithfulness: Measures the fidelity of the generated answer to the source documents. High faithfulness means the answer is based on the retrieved passages and does not contain hallucinations.

Runtime: Measures the time required for retrieval and generation. This metric is important for practical applicability.

3) Evaluation Process: For each question in the test dataset:

1) The system (RAG or GraphRAG) generates an answer
2) The answer is compared with the reference answer
3) Metrics are calculated
4) Results are logged in MLflow

The aggregated results enable systematic comparison of different approaches.

F. Orchestration and Versioning

1) DVC (Data Version Control): DVC is used for versioning data and models. This ensures:

- Reproducibility of experiments
- Traceability of changes
- Efficient storage of large files
- Collaboration in teams

All intermediate results (processed documents, knowledge graphs, embeddings) are versioned with DVC.

2) MLflow: MLflow is used for experiment tracking. For each experiment run, the following are logged:

- Parameters (e.g., chunk size, embedding model, retrieval strategy)
- Metrics (LLM Accuracy, Contain Accuracy, Faithfulness, Runtime)
- Artifacts (generated answers, retrieved passages, logs)

This enables systematic analysis and comparison of different configurations.

IV. Results

This chapter presents the results of the evaluation and discusses the findings.

A. Comparison of Approaches

Table **??** shows the aggregated results for the different approaches:

TABLE I
COMPARISON OF RAG AND GRAPHRAG APPROACHES

Approach	LLM Acc.	Cont. Acc.	Faith.	Runtime (s)
Classical RAG	3.2	0.68	0.82	1.2
LeanRAG	3.8	0.76	0.88	1.8
LinearRAG	3.6	0.72	0.85	1.5
GraphMERT	4.1	0.81	0.91	2.3

B. Analysis

The results show that graph-based approaches consistently outperform classical RAG:

LLM Accuracy: All GraphRAG variants achieve higher accuracy than the baseline. GraphMERT achieves the best result (4.1 vs. 3.2), representing an improvement of approximately 28%.

Contain Accuracy: GraphRAG approaches retrieve more relevant passages. This indicates that graph structure helps identify relevant information more precisely.

Faithfulness: All GraphRAG variants show higher faithfulness. This suggests that structured knowledge representation reduces hallucinations.

Runtime: GraphRAG approaches require more time than classical RAG. This is due to additional graph traversal and consolidation steps. GraphMERT is the slowest but also the most accurate.

C. Qualitative Observations

Beyond quantitative metrics, several qualitative observations can be made:

Multi-Hop Reasoning: GraphRAG approaches are significantly better at answering questions that require reasoning across multiple documents or sections. The explicit modeling of relations enables targeted traversal of the knowledge graph.

Context Preservation: The hierarchical structure of LeanRAG helps preserve context. Entities are not viewed in isolation but in their semantic environment.

Robustness: GraphMERT shows the highest robustness against noisy or incomplete data. The multi-stage pipeline with consolidation and verification effectively filters errors.

Scalability: LinearRAG shows the best scalability. The linear structure enables efficient traversal even with large document collections.

V. Discussion

A. Interpretation of Results

The results confirm the hypothesis that graph-based RAG approaches improve the quality of question answering for complex, knowledge-intensive questions. The explicit modeling of entities and relations enables more targeted retrieval and better context preservation.

However, the improvement comes at the cost of increased runtime. For applications where response time is critical, this may be a limiting factor. For use cases where quality is more important than speed (e.g., scientific literature review, legal analysis), the trade-off is acceptable.

B. Limitations

Several limitations should be noted:

Evaluation Dataset: OpenRAGBench focuses on scientific texts. Generalization to other domains (e.g., news, social media) is not guaranteed.

Metrics: LLM-based metrics (LLM Accuracy) depend on the quality of the evaluating LLM. Different LLMs may produce different results.

Construction Costs: Building knowledge graphs requires significant computational resources and time. For dynamic, frequently changing data, this may be impractical.

Domain Specificity: Entity and relation extraction works best for well-structured, formal texts. For informal or ambiguous texts, quality may be lower.

C. Future Work

Several directions for future work emerge:

Hybrid Approaches: Combination of vector-based and graph-based methods to leverage the advantages of both approaches. For example, fast vector search for pre-filtering, followed by graph-based refinement.

Dynamic Graphs: Development of methods for efficient updating of knowledge graphs when new documents are added or existing documents change.

Domain Adaptation: Investigation of GraphRAG approaches in other domains (e.g., medical texts, legal documents, technical documentation).

Optimization: Reduction of runtime through optimized graph traversal algorithms, caching strategies, and parallelization.

Explainability: Development of methods for explaining retrieval decisions. The graph structure offers potential for visualizing reasoning paths.

VI. Conclusion

The IMARA project demonstrates that graph-based RAG approaches can significantly improve the quality of question answering for complex, knowledge-intensive questions. The systematic comparison of classical RAG with LeanRAG, LinearRAG, and GraphMERT shows consistent improvements in accuracy, completeness, and faithfulness.

The key findings are:

- Explicit modeling of entities and relations enables more targeted retrieval and better context preservation.
- Hierarchical structures (LeanRAG) and linear ordering (LinearRAG) offer different advantages depending on the use case.
- Multi-stage pipelines with consolidation and verification (GraphMERT) increase robustness and consistency.
- The improvement in quality comes at the cost of increased runtime.

The IMARA pipeline provides a flexible, reproducible framework for evaluating RAG and GraphRAG approaches. The use of DVC and MLflow ensures traceability and enables systematic experimentation.

For future applications, the choice between classical RAG and GraphRAG depends on the specific requirements: If speed is critical, classical RAG may be sufficient. If quality and robustness are more important, GraphRAG approaches are preferable. Hybrid approaches that combine the advantages of both paradigms represent a promising direction for future research.

Acknowledgment

We thank the developers of Docling, OpenRAGBench, and OpenRAG-Eval for providing excellent open-source tools. We also thank the authors of LeanRAG, LinearRAG, and GraphMERT for their pioneering work in the field of graph-based RAG.

References

[1] Docling Project, "Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion," arXiv:2501.17887, 2025. [Online]. Available: https://www.docling.ai/

[2] Docling Project, "Docling Architecture." [Online]. Available: https://docling-project.github.io/docling/concepts/architecture/ [Accessed: Jan. 18, 2026]

[3] X. Zhang et al., "Knowledge-Graph-Based Generation with Semantic Aggregation and Hierarchical Retrieval," arXiv:2508.10391, 2025. [Online]. Available: https://github.com/KnowledgeXLab/LeanRAG

[4] Y. Li et al., "LinearRAG: Linear Graph Retrieval-Augmented Generation on Large-scale Corpora," arXiv:2510.10114, 2025. [Online]. Available: https://github.com/DEEP-PolyU/LinearRAG

[5] M. Belova, J. Xiao, S. Tuli, and N. K. Jha, "GraphMERT: Efficient and Scalable Distillation of Reliable Knowledge Graphs from Unstructured Data," arXiv:2510.09580, 2025. [Online]. Available: https://github.com/creativeautomaton/graphMERT-python

[6] Vectara, "Open RAG Benchmark (1000 PDFs, 3000 Queries): A Multimodal PDF Dataset for Comprehensive RAG Evaluation." [Online]. Available: https://github.com/vectara/open-rag-bench

Appendix

A. A - E

Chunk/Chunking: Division of a document into smaller text units (chunks) for processing in RAG systems. The chunking strategy significantly influences retrieval quality.

Contain Accuracy: Metric that measures whether retrieved passages contain the information necessary to answer a question.

Cosine Similarity: Measure of similarity between two vectors, frequently used in vector-based RAG for comparing embeddings.

DVC (Data Version Control): Tool for versioning data and models, used in IMARA for reproducibility and traceability.

Embedding: Dense vector representation of text that captures semantic meaning. Used in RAG for similarity search.

Entity: A concrete object or concept in a knowledge graph (e.g., person, organization, method, dataset).

B. F - L

Faithfulness: Metric that measures the fidelity of a generated answer to source documents.

GraphMERT: Neurosymbolic approach for building reliable knowledge graphs through multi-stage extraction, consolidation, and verification.

GraphRAG: Extension of RAG with graph-based knowledge representation and retrieval.

Knowledge Graph: Structured representation of knowledge as a graph of entities and relations.

LeanRAG: Graph-based RAG approach with semantic aggregation and hierarchical retrieval.

LinearRAG: Graph-based RAG approach with linear ordering of entities, particularly suitable for temporally or causally structured texts.

LLM (Large Language Model): Large language model (e.g., GPT-4, Claude) used in RAG for answer generation.

LLM Accuracy: Metric that uses an LLM to evaluate the quality of generated answers.

C. M - R

MLflow: Platform for experiment tracking and model management, used in IMARA for logging parameters, metrics, and artifacts.

Multi-Hop Reasoning: Reasoning that requires multiple steps or traverses multiple documents.

Naive RAG: Simple, purely vector-based RAG approach that treats knowledge as isolated facts (chunks), strongly depends on chunking strategy, and often suffers from contextual fragmentation.

Neurosymbolic AI: Combination of neural networks (for generalization and learning) and symbolic AI (for abstraction, logic, and graph structures), as implemented in the GraphMERT approach.

OpenRAGBench: Reference dataset (benchmark) with approximately 1000 scientific PDFs (arXiv) and associated question-answer pairs, used in the project to ensure measurability and comparability of results.

OpenRAG-Eval: Evaluation framework that compares different RAG and GraphRAG systems based on uniform metrics (e.g., LLM Accuracy, Contain Accuracy, Faithfulness, Runtime) and is used in the project for orchestrating benchmarks.

D. S - V

Semantic Aggregation: Feature of LeanRAG where entities are grouped into semantically coherent summaries (clusters) and represented as aggregated nodes with explicit relations to improve graph navigation and hierarchical retrieval.

Synthetic Data Generation (SDG): Approach for generating artificial test data for system evaluation, often using LLMs as a ”judge” to measure, e.g., answer quality or robustness.

TF-IDF (Term Frequency–Inverse Document Frequency): Statistical measure for evaluating the importance of a term in a document relative to a document collection. A common formula is TF-IDF = $\log(1 + \text{tf}) \times \log(\text{N} / \text{df})$, where tf is the term frequency, N is the total number of documents, and df is the number of documents containing the term. In the LinearRAG approach, TF-IDF is used, among other things, to weight passage-entity edges to quantify the semantic relevance of entities for specific text sections.

Triple: The basic data unit of a knowledge graph, consisting of subject, predicate (relation), and object (e.g., ”Müller” $\rightarrow$ ”has profession” $\rightarrow$ ”mason”).

Unsloth: Framework for resource-efficient fine-tuning of LLMs, used in the project to perform model adaptations with lower hardware requirements.

Vector Similarity Search: Standard search method of classical RAG systems, where text sections are represented as vectors in embedding space and compared based on their distance (e.g., cosine or Euclidean distance). It enables semantic search but often does not consider explicit relations between entities.