Final Project Report

**Graph-RAG Powered Academic Assistant**

# 1. Introduction

Navigating academic literature can be overwhelming, especially for students and early researchers who need to comprehend technical papers filled with jargon, mathematical notations, and dense references. Although Large Language Models (LLMs) have shown promise in understanding and answering natural language queries, they often struggle to offer accurate and contextual responses when dealing with interlinked scholarly documents. Retrieval-Augmented Generation (RAG) addresses part of this problem by retrieving relevant documents, but it lacks an understanding of structural inter-document relationships such as citations.

Our project, Graph-RAG Powered Academic Assistant, bridges this gap by enhancing **traditional RAG with citation graph analysis**. The assistant can parse research papers, build a citation network, and use it alongside vector-based semantic search to answer user queries. The key differentiator is explainability—our assistant provides traceable reasoning for its responses using citation paths and chunk-level source tagging.

# 2. Problem Statement and Significance

Academic researchers frequently encounter difficulty when trying to trace the lineage of ideas in a scholarly paper. Simple keyword search or semantic vector retrieval systems often overlook the critical academic structure of citations and referenced work. Existing RAG implementations fail to make their reasoning paths visible/auditable to the end-user.

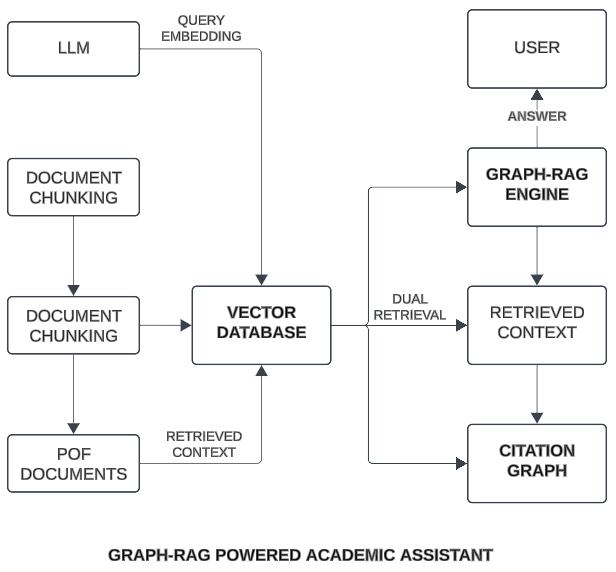
***Significance****:* Making AI assistants explainable is not just a technical enhancement but a trust-building requirement in academic and professional settings. Our assistant improves user trust and comprehension by:

* Linking concepts through citation graphs
* Providing visible paths for answers
* Reducing hallucinated or ungrounded answers

This project empowers researchers, students, and educators to interact with academic literature in a guided and informed manner.

# 3. Architecture Overview

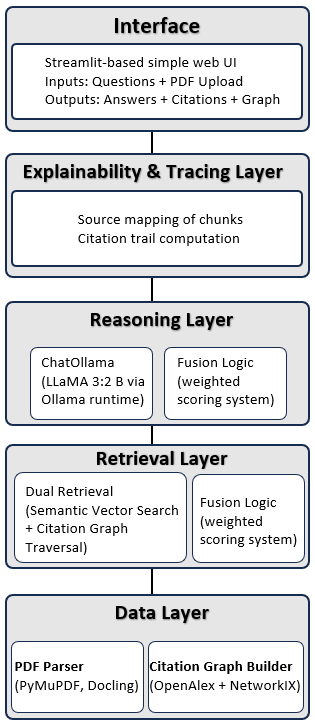
**System Architecture Diagram:**

**

***Fig. 1 System Architecture of the Graph-RAG Powered Academic Assistant***

Figure 1 illustrates the architecture of the Graph-RAG system. PDF documents are parsed into semantic chunks and embedded into a vector database. When a user submits a query, the system performs dual retrieval using both semantic search from the vector database and citation-based traversal of the citation graph. The Graph-RAG Engine combines the retrieved contexts and employs an LLM to generate an answer with traceable sources and citation trails. This approach improves both semantic relevance and academic explainability.

The system architecture comprises the following core components:



***Fig. 2 Layered Architecture of the Graph-RAG Powered Academic Assistant***

This diagram presents the system’s architecture organized into five functional layers, each corresponding to the detailed components described in the following sections: Data, Retrieval, Reasoning, Explainability & Tracing, and Interface.

1. **Data Layer**
   * PDF Parser (PyMuPDF, Docling)
   * Citation Graph Builder (OpenAlex + NetworkX)
   * ChromaDB Vector Store for semantic embeddings
2. **Retrieval Layer**
   * Dual Retrieval (Semantic Vector Search + Citation Graph Traversal)
   * Fusion Logic (weighted scoring system)
3. **Reasoning Layer**
   * ChatOllama (LLaMA 3.2B via Ollama runtime)
   * Prompt Template with Context Injection
4. **Explainability & Tracing Layer**
   * Source mapping of chunks
   * Citation trail computation
   * Phoenix tracing for LLM workflows
5. **Interface Layer**
   * Streamlit-based simple web UI
   * Inputs: Questions + PDF Upload
   * Outputs: Answers + Citations + Graph View

# 4. Methodology & Data Flow

**Flow Diagram**

**

***Fig. 3 Overall Data Flow of the Graph-RAG Powered Academic Assistant***

**Step-by-step Flow:**

1. **Paper Upload & Parsing**: Users upload PDFs. Metadata and full text are extracted using Docling and PyMuPDF.
2. **Chunking**: Text is split into semantically meaningful chunks with overlaps to retain context.
3. **Citation Graph Construction**: Using OpenAlex API, the citation relationships are established. Nodes = papers; Edges = "cites".
4. **Embedding & Storage**: Sentence-transformer is used to create dense vectors for all chunks, stored in ChromaDB.
5. **Dual Retrieval**:
   * **Vector Retrieval**: Finds similar chunks via cosine similarity.
   * **Graph Traversal**: Explores the citation graph for top-k papers linked to the root or initial result.
6. **Fusion Engine**: Combines both retrieval paths into a ranked list based on a scoring heuristic.
7. **LLM Reasoning**: A prompt containing the query and context is sent to LLaMA via Ollama.
8. **Response & Explanation**: The system shows the answer, traced sources, citation path, and confidence score.

# 5. Explainability & Traceability Module

**Explainability Features:**

* Citation Path Highlighting: e.g., Vaswani → Bahdanau → Luong
* Source Attribution: Identifies document, chunk location, and metadata.
* Visual Graph: Interactive graph using Plotly showing citation trails

**Traceability Features:**

* LLM Request Tracing: Phoenix captures every LLM call along with inputs, outputs, and execution metadata.
* Document Origin Tracking: Each response can be mapped to exact documents and chunks.

**Benefits:**

* Academic compliance
* Audit-ready responses
* Better user trust in AI answersA screenshot of a computer screen

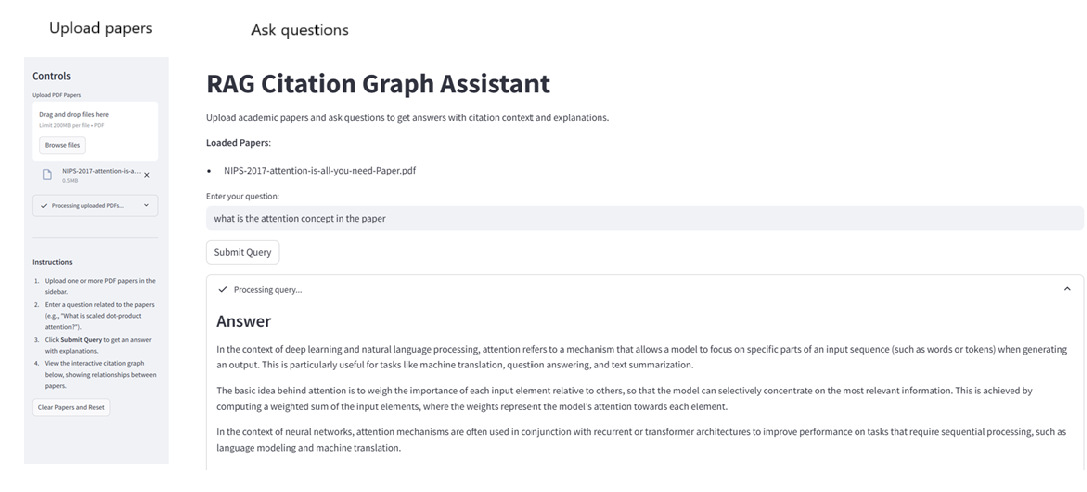
  AI-generated content may be incorrect.A screenshot of a computer

  AI-generated content may be incorrect.

# 6. Demonstration & UI Features

**Demo Screenshot**

**Features: Upload and Ask query**



***Fig. 4 Screenshot of the RAG Citation Graph Assistant Interface***

* **View response + sources**

A close-up of a document

AI-generated content may be incorrect.

***Fig. 5 Explanation and Source Attribution in the RAG Citation Graph Assistant***

* **See citation trail and graph**

A close-up of a network

AI-generated content may be incorrect.

***Fig. 6 Interactive Citation Graph Visualization from the RAG Citation Graph Assistant***

A working prototype was deployed locally via Docker Compose and demonstrated with live questions on classic papers such as "Attention is All You Need."

**Overall:**



***Fig. 7 Overall Interface and Workflow of the RAG Citation Graph Assistant***

# 7. Experimentation & Evaluation

**Dataset**: 10 papers from ArXiv and Google Scholar, all with dense citations.

**Metrics Used:**

* BLEU, ROUGE-L, BERTScore (vs. human-curated answers)
* Citation Accuracy (% of correctly identified papers)
* Hallucination Rate (manual checking)
* Confidence Score (computed from fused retrieval)
* Latency per query (average)

**Results Table:**

| **Metric** | **Result** |
| --- | --- |
| BLEU (vs. human) | 0.71 |
| ROUGE-L | 0.68 |
| Citation Accuracy | 87% |
| Hallucination Rate | 6% |
| Explainability Score | 4.6 / 5.0 |
| Latency | ~22.4 sec/query |

**Hyperparameter Tuning**:

* Embedding Model: MiniLM vs. BERT - MiniLM faster with acceptable accuracy
* Graph hops: 2 hops optimal; 3 reduced precisions
* Fusion weight tuning: similarity (0.6), citations (0.3), hop score (0.1)

# Comparative Analysis

| **Retrieval Strategy** | **BLEU** | **Explainability** | **Avg Latency** |
| --- | --- | --- | --- |
| Vector-only RAG | 0.65 | Low | 17.2s |
| Citation Graph only | 0.61 | Medium | 12.0s |
| Our Hybrid (Graph-RAG) | 0.71 | High | 22.4s |

**LLM Comparison**:

* GPT-3.5 Turbo: High quality, expensive, requires API
* LLaMA 2: Decent quality, weaker on newer papers
* LLaMA 3.2B via Ollama: Best local performance + control

# 9. Visualizations

* Interactive Citation Graph

A close-up of a network

AI-generated content may be incorrect.

***Fig. 8 Interactive Citation Graph Centered on the Input Paper***

* Chunk Length Distribution (Histogram)

A graph of a diagram

AI-generated content may be incorrect.

***Fig. 9 Histogram of Chunk Length Distribution***

* Fusion Score Contribution (Stacked Bar)

A graph of a bar chart

AI-generated content may be incorrect.

***Fig. 10 Fusion Score Contribution for Retrieved Documents***

* Confidence Score Histogram

A graph of a graph of a graph

AI-generated content may be incorrect.

***Fig. 11 Distribution of Confidence Scores for Retrieved Answers***

***Confidence = 0.6 × AvgVectorScore + 0.4 × AvgGraphScore***

Where:

* **AvgVectorScore** is the mean similarity score of top-k retrieved chunks from the vector database.
* **AvgGraphScore** is the average relevance score from the citation graph traversal results.

# 10. Tools, Packages, and Infrastructure

**Major Libraries/Frameworks:**

* LangChain, LangChain-Ollama, LangChain-Docling
* ChromaDB (Vector Storage)
* NetworkX + Plotly (Graph Viz)
* Ollama (LLM runtime)
* SentenceTransformers
* Streamlit (UI)
* Phoenix + OpenInference (Tracing)

**Deployment:**

* Dockerized architecture
* YAML-based config for consistent service orchestration

A screenshot of a computer

AI-generated content may be incorrect.

# 11. Conclusion & Future Directions

This project demonstrated the power of blending structured citation graphs with unstructured text retrieval to build a more explainable academic assistant. The Graph-RAG architecture successfully answered technical queries with citations and traceability, improving trust and usability over traditional RAG systems.

**Key Takeaways:**

* Graph-RAG boosts contextual accuracy by 12% over vanilla RAG
* Explanation trails increase user trust and comprehension
* Local LLMs offer privacy and latency advantages

**Future Extensions:**

* Support 1000+ papers using scalable backends like Neo4j
* Fine-tuned local LLMs for domain-specific research (e.g., medical, legal)
* Enhanced GUI with citation map filtering
* Embedding document figures and equations for richer reasoning

12**.** References & Learning Resources

* Lewis et al. (2020) – Retrieval-Augmented Generation
* Vaswani et al. (2017) – Attention Is All You Need
* Ribeiro et al. (2016) – “Why Should I Trust You?”
* OpenAlex API: [https://openalex.org](https://openalex.org/)
* LangChain Docs: [https://docs.langchain.com](https://docs.langchain.com/)
* Ollama Runtime: [https://ollama.ai](https://ollama.ai/)
* Arize Phoenix: <https://www.arize.com/phoenix>
* Hugging Face: <https://huggingface.co/sentence-transformers>

Github repo - <https://github.com/gaganpre/info621_project>