RAG-AI: Knowledge-Grounded Q&A System using Retrieval-Augmented Generation

Project Update Report

Dhairya Bhatt, Vidhi Patel, Diya Kaswa, Akanksh Daggupati
DS-5110 Essential of Data Science
Northeastern University, Boston

November 11, 2025

1. Dataset Description

Dataset Link: The project will utilize publicly available, copyright-free research papers, open educational materials, and institution-provided notes. Specific dataset sources will be finalized and documented in the GitHub repository.

1.1 Description

The dataset consists of various document formats (primarily PDFs) containing technical and educational content across multiple domains. These documents will be processed through a text extraction pipeline, chunked into semantically meaningful segments, and converted into vector embeddings using the sentence-transformers model (all-MiniLM-L6-v2). The dataset structure includes raw documents, processed text chunks, generated embeddings, and associated metadata stored in a SQL database layer that tracks document identifiers, chunk boundaries, timestamps, and retrieval statistics.

1.2 Why This Dataset

This particular set of open source educational and research materials is perfect for showing off a Retrieval-Augmented Generation system because it has a wide range of factual content that can be checked against source documents. These materials, on the other hand, guarantee complete reproducibility and follow academic integrity rules. The technical nature of the content makes it possible to thoroughly test the accuracy of retrieval and the relevance of responses. The different types of documents also test the system's strength across different text structures and domains.

2. Tools and Methodologies

The RAG-AI system employs a comprehensive technology stack designed to integrate retrieval and generation capabilities efficiently. The core retrieval mechanism uses **FAISS (Facebook AI Similarity Search)**, a library optimized for dense vector similarity search, enabling fast and scalable retrieval of relevant document chunks from potentially large knowledge bases. For embedding generation, **Sentence-Transformers** with the all-MiniLM-L6-v2 model converts text chunks into 384-dimensional dense vectors, balancing computational efficiency with semantic representation quality. **LangChain** serves as the orchestration framework, providing pre-built components for document loading, text splitting, prompt templating, and LLM integration, significantly reducing development complexity. The user interface is built with **Streamlit**, chosen for its rapid prototyping capabilities and native Python integration, allowing seamless connection between frontend interactions and backend processing.

For the generation component, the system integrates a Large Language Model through API calls, with the specific model selection (likely GPT-based or open-source alternatives like Llama) to be determined based on performance and accessibility. A **SQLite database** provides the metadata layer for logging queries, retrievals, responses, and performance metrics, supporting both operational analytics and project evaluation requirements. **Python 3.10+** serves as the primary programming language, leveraging its rich ecosystem for NLP and data science. Version control through **Git/GitHub** ensures collaborative development and code reproducibility, while **LaTeX (Overleaf)** maintains professional documentation standards.

This toolset was selected over alternatives for several compelling reasons. FAISS outperforms traditional database vector search in speed and memory efficiency for high-dimensional embeddings. Sentence-Transformers provides superior semantic understanding compared to older word-embedding approaches like Word2Vec. LangChain abstracts away boilerplate code that would otherwise require custom implementation when working with multiple LLMs and retrieval systems. Streamlit offers faster development cycles than traditional web frameworks like Flask or Django while maintaining sufficient functionality for a research prototype. The lightweight SQLite database is perfectly suited for the project's analytical needs without the overhead of enterprise database systems.

3. Preliminary Timeline

3.1 Week 1 (Current Week - Project Setup & Data Preparation) - Completed

- Finalize and document all data sources with proper citations
- Implement PDF text extraction pipeline using PyPDF2 or pdfplumber
- Develop text chunking strategy with configurable parameters (chunk size, overlap)
- Generate embeddings for all document chunks using Sentence-Transformers
- Create initial SQLite database schema for metadata storage
- **Deliverable:** Processed dataset with embeddings, initial database structure

3.2 Next steps: (Embedding data and Storing in Vector Table)

- Build FAISS index from generated embeddings
- Implement top-k retrieval functionality with configurable similarity thresholds
- Develop retrieval evaluation framework with test queries
- Integrate SQL logging for retrieval operations (query, results, scores, timestamps)
- Begin preliminary retrieval accuracy testing with sample queries
- **Deliverable:** Functional retrieval system with logging, initial performance metrics

4. Team Member Contributions

4.1 Dhairya Bhatt

Dhairya took to building the project's technical base, setting up the Python 3.10+ environment and managing the configuration of key libraries. They focused on ensuring compatibility across systems and smooth integration of dependencies like pdfplumber, PyMuPDF, and pandas. Their experimentation with structured and unstructured PDFs allowed the team to refine extraction methods early on. They also advised incorporating numpy for efficient data manipulation, which made subsequent text and table handling far more streamlined.

4.2 Vidhi Patel

Vidhi worked extensively on layout-sensitive extraction, ensuring that the project could handle PDFs containing a mixture of text, tables, and images. They leveraged PyMuPDF's layout analysis to align extracted text correctly and maintain spatial coherence. Their fine-tuning of extraction logic helped minimize data loss in visually complex documents. This member also played a key role in validating output consistency by testing multiple file formats and suggesting logging mechanisms for better debugging and reproducibility.

4.3 Diya Kaswa

Diya focused on enhancing accessibility and workflow clarity. They designed preliminary data viewing mechanisms that helped the team verify extracted text, images, and tables quickly. They also worked closely with others to establish consistent formatting standards and folder structures for the extracted outputs. Their coordination ensured that data from all sources—text, image, and tabular—could be processed uniformly, making it easier to prepare for later embedding and search stages without major rework.

4.4 Akanksh Daggupati

Akanksh took responsibility for developing a systematic extraction flow that tied all components together. They coordinated between the modules built using pdfplumber and PyMuPDF, verifying that both tools complemented each other across different file types. They also proposed storing extraction results and metadata in sqlite3, a lightweight and easily accessible database, ensuring that every processed document could be retraced and verified. Their contribution strengthened the project's reliability and made future analysis more transparent.

4.5 Collaboration Approach

The team has been meeting bi-weekly for coordination sessions where progress is reviewed, blockers are discussed, and upcoming tasks are allocated. A shared Slack channel facilitates daily communication and quick problem-solving. GitHub pull requests with peer reviews ensure code quality and knowledge sharing across the team. As the project progresses into more specialized phases (retrieval, generation, interface), team members will increasingly support each other across their primary responsibility areas, with everyone contributing to testing and the final report writing.

5. Progress and Next Steps

5.1 Progress to Date

The team has successfully completed the project planning phase and established the foundational infrastructure. Key accomplishments include finalizing the project scope and objectives, identifying appropriate open-source datasets that meet academic integrity requirements, setting up the development environment with version control and documentation systems, defining the four-week timeline with specific deliverables, and producing this initial project report. The team has also completed preliminary research on all major technical components (FAISS, Sentence-Transformers, LangChain, Streamlit) and understands the implementation requirements for each phase.

5.2 Ongoing Challenges

Several challenges have been identified requiring careful attention. Selecting the optimal LLM involves balancing performance, cost, and accessibility, while determining appropriate chunk sizes and overlap parameters will significantly impact retrieval accuracy through experimentation. Additionally, designing evaluation metrics for subjective measures like response relevance and ensuring the SQL logging layer captures sufficient detail without performance bottlenecks need thoughtful development.

5.3 Next Steps (Immediate - Week 2)

In the coming week, the team will focus on completing data preparation and beginning retrieval system development. Specific tasks include finalizing the list of documents to be used with full citations, implementing the PDF text extraction and chunking pipeline with configurable parameters, generating embeddings for all document chunks using the all-MiniLM-L6-v2 model, creating the SQLite database with the designed schema, conducting initial tests to validate that embeddings capture semantic meaning effectively, and beginning FAISS index construction. The team will collaboratively work on data preprocessing, database implementation, and preparations for LLM integration and UI development.