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

Initial Report

DS-5110 Essential of Data Science

Northeastern University, Boston

Team Members: Dhairya Bhatt, Vidhi Patel,Diya Kaswa, Akanksh Daggupati

GitHub Repository:
`https://github.com/vidhi3839/RAG-Retrieval-Augmented-Generation-`

1. Introduction

The project **"RAG-AI: Knowledge-Grounded Q&A System using Retrieval-Augmented Generation"** aims to build an intelligent assistant capable of generating accurate, context-aware, and knowledge-grounded answers from custom documents. The system integrates a vector-based retrieval mechanism with a Large Language Model (LLM) to achieve domain-specific understanding.

Unlike conventional LLMs, which rely solely on internal parameters, the RAG approach retrieves relevant information from an external knowledge base before generating answers. This ensures responses are verifiable and aligned with factual content from user-provided PDFs, articles, and notes.

2. Project Scope and Objectives

2.1 Objectives

- Implement a Retrieval-Augmented Generation pipeline combining FAISS vector search.
- Generate embeddings using `sentence-transformers (all-MiniLM-L6-v2)`.
- Build an interactive Streamlit interface for user queries and grounded responses.
- Log queries, retrievals, and responses in a SQL-based metadata layer for reproducibility.
- Evaluate the system using metrics such as retrieval accuracy, response relevance, and latency.

2.2 Scope

Document ingestion, text chunking, embedding generation, FAISS-based retrieval, LLM integration, SQL logging, and a basic Streamlit interface.

3. Key Deliverables and Milestones

Phase	Timeline	Deliverable
Week 1	Project setup & data preparation	Document ingestion, chunking, and embedding generation.
Week 2	Retrieval system	FAISS index creation, top-k retrieval testing.
Week 3	Generation & evaluation	LLM response generation + retrieval accuracy testing.
Week 4	Interface & report	Streamlit UI, SQL reports, visual dashboards, final documentation.

4. Team Roles and Capabilities

Member	Skills	Responsibilities
Dhairya Bhatt	Python, NLP, Data Analysis, SQL analytics	Data preprocessing, embedding generation, report writing, LLM integration.
Vidhi Patel	Python, Data analysis and API Design, LLM integration, Testing	LLM integration, prompt design, testing, report writing.
Diya Kaswa	Python, API integration, Web/UI Development	Streamlit frontend, user interface design, report writing.
Akanksh Daggupati	Python, Web development, Database Design, Visualization, SQL	Metadata logging, SQL report generation, visualization, report writing.

5. Data Access and Compliance

All documents used in this project will be publicly available, copyright-free resources such as research papers, open educational materials, or institution-provided notes. No proprietary or paid datasets will be accessed. Each data source will be cited appropriately in the final report.

6. Database Design and SQL Layer

To enhance data traceability and reproducibility, a lightweight **SQL metadata layer** will be integrated. It will store:

- User queries and timestamps
- Retrieved document identifiers and similarity scores
- Generated model responses
- Response latency and confidence scores

Additionally, **8 SQL analytical queries** will be implemented to demonstrate database proficiency. Example queries include:

1. Query logs by document type or topic.
2. Most frequently retrieved files or passages.
3. Average response latency by query length.
4. Query success rate (retrieval + response match).
5. Embedding similarity distribution.
6. User interaction frequency by session.
7. Accuracy or confidence trends over time.
8. System error or fallback frequency.

7. Evaluation Metrics

The system will be evaluated using:

- **Retrieval Accuracy:** Percentage of relevant chunks retrieved.
- **Response Relevance:** Human-rated alignment between generated answers and source text.
- **Latency:** Average response time from query to final answer.

These metrics will be visualized in Streamlit dashboards, demonstrating performance trends and reliability.

8. Tools and Technologies

- **Programming Language:** Python 3.10+
- **Libraries:** Sentence-Transformers, FAISS, LangChain, Streamlit, SQL
- **Version Control:** Git and GitHub
- **Documentation:** Overleaf (LaTeX) and Excel Progress Tracker

9. Progress Tracking

Project progress will be monitored weekly using an Excel tracker recording milestones, task completion percentage, and blockers.

- This Overleaf PDF report (3 pages)
- Excel progress tracker
- GitHub repository with code, documentation, and SQL reports

10. Conclusion

This project will deliver a fully functional Retrieval-Augmented Generation (RAG) pipeline capable of producing knowledge-grounded answers from uploaded documents. The integration of FAISS, LLM, and a SQL analytics layer ensures both technical sophistication and real-world applicability. The focus on reproducibility, authentic data use, and performance evaluation aligns with the project's educational and research objectives.