Retrieval-Augmented Generation (RAG) Pipeline Documentation

1. Introduction

Retrieval-Augmented Generation (RAG) is a modern AI architecture that combines information retrieval with natural language generation. Instead of relying solely on a model's internal knowledge, RAG retrieves relevant information from external documents and uses that context to generate accurate and grounded outputs. In this project, we built a complete end-to-end RAG pipeline using SEC financial filings. The pipeline includes document loading, preprocessing, embedding generation, vector indexing using FAISS, retrieval, structured information extraction, Spark integration, and visualization.

2. Problem Statement

Traditional language models suffer from hallucination and lack access to real-time or domain-specific data. Financial documents such as SEC filings contain critical business information that must be retrieved accurately. The goal of this project was to build a scalable system capable of: • Processing large financial documents • Converting text into semantic embeddings • Storing embeddings efficiently • Retrieving relevant information based on queries • Extracting structured insights such as revenue, income, and industry

3. Architecture Overview

The pipeline consists of the following components: Document Layer: Loads SEC filings dataset Preprocessing Layer: Combines sections into a single text document Splits into smaller chunks Embedding Layer: Uses sentence-transformers to convert text into vector embeddings Vector Database Layer: Stores embeddings in FAISS index Retrieval Layer: Finds most similar chunks based on query Extraction Layer: Extracts structured information Processing Layer: Uses Apache Spark for scalable distributed processing Visualization Layer: Displays similarity heatmaps and dashboards

4. Implementation Steps

Step 1: Document Loading Loaded SEC dataset using HuggingFace datasets library. Step 2: Text Preprocessing Combined all sections and prepared clean text. Step 3: Chunking Split documents into smaller chunks for better retrieval accuracy. Step 4: Embedding Generation Used sentence-transformers to convert text chunks into vectors. Step 5: Vector Storage Stored vectors in FAISS for fast similarity search. Step 6: Retrieval Implemented similarity-based retrieval using cosine similarity. Step 7: Extraction Extracted

key financial variables such as revenue and industry. Step 8: Spark Integration Converted results into Spark DataFrame for scalability. Step 9: Visualization Created dashboards and similarity heatmaps.

5. Your Understanding and Technical Knowledge

Through this project, you demonstrated understanding of: • Vector embeddings and semantic similarity • Vector databases such as FAISS • Retrieval-based architectures • Distributed computing using Apache Spark • Data preprocessing and transformation • Information extraction from unstructured data • Visualization and analysis This shows strong capability in AI Engineering, Data Engineering, and MLOps.

6. Benefits of RAG Pipeline

Key benefits include: • Eliminates hallucination • Improves factual accuracy • Scales to large datasets • Enables real-time knowledge retrieval • Works with domain-specific data This makes RAG ideal for financial analysis, legal systems, enterprise search, and AI assistants.

7. Why RAG is Important

RAG is critical because: • LLMs cannot memorize all information • Enterprises need private knowledge integration • Improves reliability of AI systems • Enables production-grade AI systems Most modern AI systems such as ChatGPT Enterprise use RAG architecture.

8. Production Use Cases

Real-world applications include: • Financial analysis systems • Enterprise document search • Legal research assistants • Customer support AI • Healthcare knowledge retrieval

9. Conclusion

This project demonstrates a complete production-grade RAG pipeline. It integrates document processing, embedding, vector databases, distributed processing, and visualization. This architecture represents modern AI system design and is used widely in industry.