



Artificial Intelligence Lab

(CSP472)

B.TECH 3rd YEAR
SEMESTER: 6th
SESSION: 2025-2026

Submitted By:
Yashwi Pandey (2024389922)

SECTION: H (Group 2)

Submitted To
Mr. Ayush Singh, Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SHARDA SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
SHARDA UNIVERSITY, GREATER NOIDA

RAG Assignment

Problem Statement

Growing organizations generate massive amounts of reports and data critical for decision-making. For example, venture capital analysts at firms like Andreessen Horowitz must extract insights from dense documents such as HBR's "How Apple is Organized for Innovation." Manual review is slow and inefficient, but Semantic Search and Retrieval-Augmented Generation (RAG) can deliver quick, precise answers to targeted questions, enabling faster strategic insights.

The problem is that large language models cannot answer questions from private or domain-specific documents reliably. This project solves the problem by using Retrieval Augmented Generation (RAG), where relevant document chunks are retrieved from a knowledge base and passed to the language model to generate accurate, context-aware answers.

The goal is to reduce hallucination and improve answer reliability by grounding responses in actual document content.

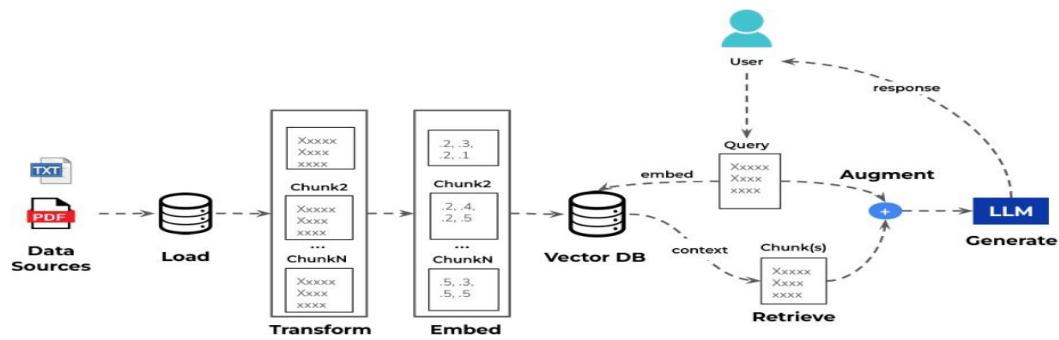
Dataset / Knowledge Source

Type of data - PDF

Data source - Public

The dataset consists of a publicly available PDF document, called "HBR_How_Apple_Is_Organized_For_Innovation-4.pdf", related to Artificial Intelligence concepts.

RAG Architecture



Block diagram of complete RAG pipeline

System Workflow

The RAG pipeline consists of the following steps:

1. Document Loading
2. Text Chunking
3. Embedding Generation
4. Vector Storage
5. Retrieval
6. Answer Generation

Text Chunking Strategy

Chunk Size: 500 characters

Chunk Overlap: 50 characters

Reason for Strategy:

Chunking breaks large documents into smaller manageable sections.

Overlap ensures contextual continuity between chunks and improves retrieval accuracy by preventing loss of important information across chunk boundaries.

Embedding Details

Embedding Model Used: `sentence-transformers/all-MiniLM-L6-v2`

Reason for Selecting the Model:

- Lightweight and efficient
- Good semantic similarity performance
- Suitable for small-scale RAG systems
- Works well with FAISS/Chroma

Vector Database

Vector Store Used: Chroma

Reason for Selection:

- Fast similarity search
- Easy local setup
- Efficient for small to medium datasets
- Open-source and widely used in RAG pipelines

Notebook Implementation

The notebook includes:

- Step-wise implementation:
 - Data loading
 - Text splitting
 - Embedding creation
 - Vector storage
 - Retrieval
 - Generation
- Proper comments and markdown cells
- Minimum three test queries with outputs

Future Improvements

- Semantic chunking
- Hybrid search (BM25 + vector search)
- Reranking models
- Metadata filtering
- Web-based UI using Streamlit
- Support for multiple documents