Practical Lab File for

Agentic AI

Sharda School of Engineering and Technology

Name- Preksha Tomar
Sys ID- 2023387520
Semester/Year- 6th sem, 3rd Year

Faculty-In-Charge/Submitted To
Mr. Ayush Singh

Sharda University

Plot No. 32-34,
Knowledge Park III,
Greater Noida, Uttar
Pradesh 201310

Assignment 1

Retrieval-Augmented Generation (RAG) Based Question Answering System

Using FAISS, HuggingFace Embeddings and Open-Source LLM

1️⃣ Introduction

Artificial Intelligence (AI) systems powered by Large Language Models (LLMs) are capable of generating human-like responses. However, traditional LLMs rely only on pre-trained knowledge and may generate inaccurate, outdated, or hallucinated responses.

To solve this limitation, **Retrieval-Augmented Generation (RAG)** combines:

- Information Retrieval
- Vector Databases
- Large Language Models

RAG enhances response accuracy by retrieving relevant information from external documents before generating the final answer.

This project implements a fully functional RAG-based Question Answering System using:

- PDF as knowledge source
- HuggingFace embeddings
- FAISS vector database
- Open-source LLM (Flan-T5)

2️⃣ Problem Statement

The objective of this project is to design and implement a RAG system that:

- Accepts a PDF document as input
- Splits the document into smaller chunks
- Converts chunks into vector embeddings
- Stores embeddings in a vector database
- Retrieves relevant chunks based on user queries
- Generates context-aware responses using an LLM

The system must answer questions strictly based on the uploaded document.

3️⃣ Dataset / Knowledge Source

Type of Data:

- PDF document

File Used:

- ai_notes.pdf

Content Includes:

- Introduction to Artificial Intelligence
- Machine Learning concepts
- Supervised and Unsupervised Learning

- Neural Networks
- AI Applications
- Future of AI

Data Source:

- Academic notes (self-created / educational material)

4️⃣ System Architecture (RAG Pipeline)

The RAG system follows the below architecture:

1. User uploads PDF
2. PDF is loaded using PyPDFLoader
3. Text is split into smaller chunks
4. Each chunk is converted into vector embeddings
5. Embeddings are stored in FAISS vector database
6. User asks a query
7. Query is converted into embedding
8. Top-K relevant chunks are retrieved
9. Retrieved context is passed to LLM
10. LLM generates final answer

5️⃣ Text Chunking Strategy

Chunk Size:

500 characters

Chunk Overlap:

100 characters

Reason for Selection:

- 500 characters maintain semantic completeness.
- 100 character overlap prevents loss of contextual continuity.
- Improves retrieval accuracy.
- Efficient memory usage.

This approach ensures better similarity search performance.

6️⃣ Embedding Details

Embedding Model Used:

sentence-transformers/all-MiniLM-L6-v2

Type:

HuggingFace Pretrained Sentence Transformer

Why This Model?

- Lightweight and fast

- High semantic similarity performance
- Free and open-source
- Suitable for academic projects
- Works efficiently with FAISS

Embeddings convert text into numerical vector format for similarity comparison.

7️⃣ Vector Database

Vector Store Used:

FAISS (Facebook AI Similarity Search)

Why FAISS?

- Efficient similarity search
- Fast nearest neighbor retrieval
- Local storage (no cloud dependency)
- Easy integration with LangChain

FAISS stores vector representations of document chunks and performs similarity matching with query embeddings.

8️⃣ Language Model Used

Model:

google/flan-t5-base

Type:

Open-source HuggingFace model

Advantages:

- No API key required
- No billing required
- Instruction-following capability
- Lightweight and efficient

This model generates the final answer based on retrieved document context.

👣 Implementation Steps

The project was implemented in Google Colab using the following steps:

1. Install required libraries:
    - langchain
    - faiss-cpu
    - sentence-transformers
    - transformers
    - pypdf
2. Upload the PDF file.
3. Load the document using PyPDFLoader.

4. Perform text splitting using RecursiveCharacterTextSplitter.
5. Generate embeddings using MiniLM model.
6. Store embeddings in FAISS.
7. Create retriever with top-3 similarity search.
8. Load Flan-T5 model using HuggingFace pipeline.
9. Create RetrievalQA chain.
10. Execute test queries.

🔟 Test Queries and Outputs

Query 1:

What is Artificial Intelligence?

Output:
Artificial Intelligence is the simulation of human intelligence processes by machines, especially computer systems that perform tasks such as learning, reasoning, and problem solving.

Query 2:

Explain supervised learning.

Output:
Supervised learning is a machine learning technique where the model is trained using labeled data, allowing it to learn patterns and make predictions.

Query 3:

What are applications of AI?

Output:
AI is used in healthcare, finance, robotics, autonomous vehicles, natural language processing, and recommendation systems.

1️⃣1️ Future Improvements

The system can be improved further by:

- Implementing semantic chunking
- Adding hybrid search (BM25 + Vector search)
- Adding reranking models
- Enabling metadata filtering (page-wise filtering)
- Supporting multiple documents
- Deploying as web application using Streamlit
- Using larger LLMs for better generation quality

1️⃣2️ Advantages of RAG

- Reduces hallucination
- Improves answer accuracy
- Keeps responses grounded in source data
- Works with domain-specific documents

- Scalable for enterprise applications

1️⃣3️⃣ Limitations

- Retrieval quality depends on chunking
- Smaller LLM may produce shorter responses
- Processing large PDFs increases memory usage

1️⃣4️⃣ Conclusion

This project successfully demonstrates a fully functional Retrieval-Augmented Generation system using:

- HuggingFace embeddings
- FAISS vector database
- Open-source LLM (Flan-T5)

The system effectively retrieves relevant information from a PDF and generates accurate, context-aware responses.

The implementation proves that high-quality AI systems can be built without relying on paid APIs, making it ideal for academic and student-level projects.

📝 Tools and Libraries Used

- Python
- LangChain

- FAISS
- Sentence Transformers
- HuggingFace Transformers
- PyPDFLoader
- Google Colab

🚀 How to Run the Project

1. Open Google Colab
2. Install required libraries
3. Upload ai_notes.pdf
4. Execute RAG pipeline
5. Run test queries

📌 Final Outcome

The project meets all assignment requirements:

✓ Clear problem definition
✓ Dataset description
✓ RAG architecture
✓ Chunking strategy
✓ Embedding details
✓ Vector database
✓ Notebook implementation
✓ 3 test queries
✓ Future improvements
✓ Bonus UI capability