Laboratory

file on

AGENTIC AI

School of Engineering and Technology

Department of Computer Science and Engineering

Subject code – **CSCR3215**

SUBMITTED BY:
Aishwarya Shelke
2023000836

SUBMITTED TO:
Mr. Ayush Kumar Singh

Sharda University

Greater Noida, Uttar Pradesh

Lab 2 - Implementation of Multimodal Retrieval-Augmented Generation (RAG) Architecture

Aim

To design and implement a Multimodal Retrieval-Augmented Generation (RAG) system that retrieves relevant information from multiple data modalities and generates accurate, context-aware responses using a language model.

Objective

1. To understand the concept of Retrieval-Augmented Generation (RAG).
2. To integrate multiple data modalities such as text and images.
3. To retrieve relevant information using embeddings.
4. To generate responses using a large language model.
5. To improve answer accuracy by grounding generation in retrieved data.

Theory

Retrieval-Augmented Generation (RAG) is a hybrid architecture that combines **information retrieval** with **text generation**. Instead of generating responses solely based on pre-trained knowledge, RAG retrieves relevant external data and uses it as additional context for generation.

A **Multimodal RAG** system extends this concept by supporting multiple data types such as:

- Text
- Images
- Documents
- Metadata

The system first converts inputs into embeddings, retrieves the most relevant data from a vector database, and then passes the retrieved context to a language model to generate meaningful and accurate responses.

This architecture reduces hallucinations and improves factual correctness.

Software Requirements

- Python 3.x
- Jupyter Notebook / Google Colab
- PyTorch
- Hugging Face Transformers
- FAISS / Vector Database
- PIL / OpenCV (for image processing)

Hardware Requirements

- System with GPU support (recommended)
- Minimum 8 GB RAM

Dataset Description

The dataset may include:

- Text documents
- Images
- Associated metadata

Each data item is converted into embeddings and stored in a vector database for retrieval.

Working

Step 1: Import Required Libraries

Libraries for deep learning, embeddings, vector search, and data handling are imported.

Step 2: Load Embedding Models

Separate embedding models are used for:

- Text data
- Image data

These models convert inputs into high-dimensional numerical vectors.

Step 3: Data Preprocessing

- Text is cleaned and tokenized.
- Images are resized and normalized.
- Each data item is converted into embeddings.

This ensures consistent and efficient retrieval.

Step 4: Store Embeddings in Vector Database

All embeddings are stored in a vector database such as FAISS, enabling fast similarity search.

Step 5: User Query Processing

- User input is converted into an embedding.

- Similar embeddings are retrieved from the vector database.

This step identifies the most relevant content.

Step 6: Context Construction

Retrieved text and image information is combined into a single context block to be passed to the language model.

Step 7: Response Generation

The language model receives:

- User query
- Retrieved contextual information

It then generates a response grounded in retrieved data.

Step 8: Output Display

The final response is displayed to the user, ensuring it is accurate and context-aware.

Result

The Multimodal RAG system successfully retrieves relevant information from multiple data sources and generates accurate, context-based responses. The system performs better than standalone language models by reducing hallucinations.

Conclusion

Multimodal RAG architecture enhances the reliability and accuracy of generated responses by combining retrieval and generation. Integrating multiple modalities further improves contextual understanding and applicability across diverse domains.