AGENTIC AI LAB

(CSCR-3214)

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

Submitted by:

Mayank Kumar

(2023542006) CSF(G-2)

Submitted to: Mr. Ayush Kumar

Assistant Professor

Department of Computer Science and Engineering

School of Computing Science & Engineering

Sharda University, Greater Noida

Indian Sign Language Knowledge Assistant (Advanced RAG)

1. Project Overview

Large Language Models often struggle with localized linguistic nuances, frequently misrepresenting **Indian Sign Language (ISL)** as a mere hand-representation of spoken languages like Hindi or English. This project addresses these gaps by implementing a **Retrieval-Augmented Generation (RAG)** system.

The system utilizes a curated knowledge base of 35 factual anchors regarding ISL grammar, history, and its status as an independent, natural language. By anchoring the generator in these official texts, the system eliminates hallucinations and provides factually grounded answers to complex questions about the Deaf community's primary language.

2. Tools & Libraries Used

The project is built using a modern AI stack optimized for local execution in a Google Colab environment:

- **LangChain / LangChain-Classic:** Orchestration framework for connecting document loaders, vector stores, and LLM chains.
- **HuggingFace-Transformers / HuggingFacePipeline:** Used to load and run the embedding and generation models locally on CPU/GPU.
- **Sentence-Transformers (all-MiniLM-L6-v2):** A lightweight model that maps sentences to a 384-dimensional dense vector space for high-accuracy semantic similarity.
- **FAISS (Facebook AI Similarity Search):** A high-performance library for efficient similarity search and clustering of dense vectors.
- **PyPDF:** Used for parsing and extracting text from digital PDF knowledge sources.
- **Google FLAN-T5-Large:** The sequence-to-sequence generator model used to synthesize answers based on retrieved context.

3. Instructions to Run the Notebook

To execute the project, follow these steps within your Jupyter or Colab environment:

1. **Environment Setup:** Ensure you have a GPU runtime enabled (optional but recommended).
2. **Installation:** Run the initial cells to install the required dependencies:

Bash

```
pip install langchain langchain-community langchain-huggingface faiss-cpu sentence-transformers pypdf
```

3. **Data Loading:** Upload your PDF source (e.g., SignLang_RAG.pdf). The PyPDFLoader will automatically parse the document into pages and metadata.
4. **Chunking:** The system uses a RecursiveCharacterTextSplitter with a chunk_size of 400 characters and a chunk_overlap of 50 to maintain context across splits.
5. **Vectorization:** Execute the embedding cell to generate vectors and store them in the local FAISS index.
6. **Querying:** Use the qa_chain.invoke(query) function to ask questions. The system will retrieve the top 3 relevant sources and print the fact-grounded answer.

4. Future Improvements

While the current implementation provides a robust baseline, the following enhancements are planned:

- **Semantic Chunking:** Replace fixed-character splitting with semantic splitters that analyze the embedding distance between sentences to ensure chunks are contextually complete.
- **Neural Reranking:** Implement a two-stage retrieval process. Use FAISS for the initial top-10 candidates, followed by a **Cross-Encoder Reranker** to prioritize the most relevant text before passing it to the LLM.
- **Metadata Filtering:** Leverage document metadata (like page numbers or chapters) to allow users to narrow their search to specific sections of the knowledge base.
- **Interactive UI:** Integrate **Gradio** to provide a user-friendly chat interface, allowing non-technical users to interact with the ISL assistant directly via a web link.
- **Hybrid Search:** Combine semantic vector search with traditional keyword-based search (BM25) to improve accuracy for specific terminology or ISL-specific proper nouns.