**Chatbot**

**1. Introduction**

This project focuses on developing a **Retrieval-Augmented Generation (RAG)** pipeline for summarizing content and answering questions from large English PDF datasets. The pipeline handles both scanned and digitally-created documents, leveraging a combination of Optical Character Recognition (OCR), embeddings, and large language models (LLMs) to provide accurate and relevant answers while maintaining scalability.

**2. System Architecture and Components**

**2.1 Text Extraction**

* **Objective**: Extract text from both scanned and digitally-created PDFs.
* **Methods**:
  + **OCR**: Use **Tesseract** to handle text extraction from scanned documents.
  + **Digital PDF Parsing**: Use **PyMuPDF** (Fitz) for efficient text extraction from digitally-created PDFs.

**2.2 Optimized Text Chunking**

* **Objective**: Chunk text efficiently to support better retrieval and embedding generation.
* **Methods**:
  + **Recursive Character Text Splitter**: Used to divide documents into chunks of 300-500 tokens. This ensures optimal chunk sizes for embedding while maintaining document context.

**2.3 Hybrid Search (Keyword + Semantic Search)**

* **Objective**: Implement a combination of keyword-based and semantic search for accurate information retrieval.
* **Methods**:
  + **Semantic Search**: Utilize **FAISS** for fast vector search based on embeddings created by **all-MiniLM-L6-v2**, a lightweight model from the **Sentence-Transformers** library.
  + **Keyword-based Search**: Add a metadata filtering mechanism to handle language, source type, and other attributes to improve the search results.

**2.4 LLM and Embedding Models**

* **Objective**: Select compact models that ensure scalability and efficiency while maintaining fluency and relevance in summarization and answers.
* **Models**:
  + **Embedding Model**:
    - **all-MiniLM-L6-v2**: A compact sentence-transformer model that generates high-quality embeddings with a small memory footprint (~33MB), ideal for efficient semantic search.
  + **LLM**:
    - **Llama2-7B**: A relatively small model (7 billion parameters) designed for generating fluent text, used for summarization and question answering. It balances performance and model size effectively.
  + **Pipeline**: **HuggingFace’s text generation pipeline** to integrate Llama2 and manage the generative response functionality.

**2.5 Query Decomposition**

* **Objective**: Break down complex user queries into smaller sub-queries to extract more relevant and granular information.
* **Methods**:
  + Use **LangChain’s structured query decomposition** to incrementally retrieve and process content by splitting complex queries into simpler sub-tasks for more accurate results.

**2.6 Reranking for Improved Relevance**

* **Objective**: Ensure the returned search results are both relevant and diverse.
* **Methods**:
  + Implement **Maximal Marginal Relevance (MMR)** to prioritize the most relevant and non-redundant chunks, ensuring that users receive diverse yet accurate results.

**2.7 Vector Database and Metadata Filtering**

* **Objective**: Efficiently retrieve relevant data by leveraging high-performance vector databases and metadata-based filtering.
* **Methods**:
  + **FAISS** is employed for handling large-scale vector data to manage semantic search.
  + **Metadata Filtering**: Enables filtering based on attributes like language or document type, ensuring more precise retrieval.

**3. Evaluation Criteria**

**3.1 Query Relevance**

* The system evaluates the search results based on how well they align with the user’s query and intent, ensuring the extracted information is accurate and relevant.

**3.2 Retrieval Test**

* Tests are performed to validate whether the returned chunks match the context of the query, ensuring the responses make use of relevant and coherent sections of the document.

**3.3 Latency**

* The pipeline is evaluated on its response times, with a focus on optimizing for low-latency performance to provide a faster user experience (aiming for 4-5 seconds per query).

**3.4 Fluency**

* The answers generated by the LLM are assessed for fluency, coherence, and readability, ensuring that users receive clear and well-organized responses.

**3.5 Size of Models**

* The pipeline aims to use smaller models (2B parameters or less) wherever possible to ensure that the system remains scalable while maintaining strong performance.

**4. Performance and Evaluation**

**4.1 Performance Metrics**

* **Latency**: Measures the system’s response time for each query, optimized for 2-3 seconds.
* **Scalability**: Tests are conducted with increasing dataset sizes (up to 1TB) to ensure stability and performance at scale.

**4.2 System Capabilities**

* **Embeddings Creation**: Efficient embedding generation for large PDF files using compact models like **all-MiniLM-L6-v2**.
* **Query Decomposition**: Demonstrated capability in breaking down complex queries into smaller sub-queries for more accurate retrieval.
* **Chat Memory**: The system maintains conversation memory, allowing better contextual responses over multiple interactions.
* **Metadata Filtering**: Integrates metadata-based filtering, improving precision and relevance in search results.

By incorporating lightweight yet high-performing models, the pipeline is optimized for scalability, ensuring it handles large amounts of data efficiently while maintaining relevance and performance across multiple languages and document types.