**System Architecture and Components Documentation**

The repository contains two code files. Code1 contains RAG pipeline , whereas code2 is a simple code for text analysis. Code2 also require to replace your Hugging Face API Key in order to execute it.

Project summary is provided with respect to Code1.

**1. Project Summary**

The system is a **Retrieval-Augmented Generation (RAG) Pipeline** for processing, analyzing, and querying information from PDF documents. It integrates multiple technologies to handle diverse types of PDFs, including both digital and scanned files, and builds a robust information retrieval system leveraging OpenAI's language models.

**Key Features:**

* Extracts text from digital PDFs using PyMuPDF.
* Processes scanned PDFs using OCR (Tesseract).
* Preprocesses extracted text for efficient chunking.
* Constructs a FAISS-based vector store for information retrieval.
* Implements OpenAI's GPT model for query answering with relevant source documents.

This pipeline is tailored for use cases involving large-scale document analysis, making it suitable for research, legal, or financial document processing.

**2. System Architecture**

**2.1 Text Extraction**

* **Component**:
  + **Digital PDFs**: PyMuPDF (fitz) is used for fast and accurate text extraction from structured PDFs.
  + **Scanned PDFs**: Tesseract OCR processes image-based PDFs, supporting multiple languages (eng, hin, ben, chi\_sim).
* **Challenges**: Differentiating between digital and scanned PDFs automatically.

**2.2 Text Preprocessing**

* **Component**: A preprocessing function splits raw extracted text into manageable chunks of 512 tokens for optimal embedding and model performance.
* **Challenges**: Handling noisy text or incomplete extraction from scanned documents.

**2.3 Vector Store**

* **Component**: FAISS (Facebook AI Similarity Search) creates an efficient vector store for storing and retrieving embeddings of document chunks.
* **Key Functions**:
  + Builds and saves the vector store locally.
  + Reloads vector stores to optimize repeated queries.
* **Challenges**: Managing vector store size for large datasets.

**2.4 LLM Integration**

* **Component**: OpenAI GPT (text-davinci-003) performs natural language understanding and retrieval-augmented generation.
* **Key Functions**:
  + Answers user queries using retrieved embeddings.
  + Provides source documents for transparency.

**2.5 Query Answering**

* **Component**: The RetrievalQA chain integrates retrieval from FAISS with OpenAI GPT to generate concise, context-aware responses.

**3. Challenges Faced**

1. **Poppler Dependency**:
   * Issue: Required for converting PDF pages to images for OCR processing.
   * Solution: Added system PATH configuration for Poppler binaries.
2. **Tesseract OCR Accuracy**:
   * Issue: Low accuracy with poorly scanned or low-resolution documents.
   * Solution: Adjusted preprocessing steps and language models (eng+hin+ben+chi) to improve OCR output.
3. **Handling Large Documents**:
   * Issue: Memory overhead during embedding generation for extensive text.
   * Solution: Optimized chunk size and implemented FAISS-based vector store for efficiency.
4. **API Key Management**:
   * Issue: Proper environment variable setup for OpenAI API key.

**4. Future Improvement Areas**

1. **PDF Type Detection**:
   * Automate differentiation between digital and scanned PDFs.
2. **Text Cleaning**:
   * Enhance preprocessing for better noise removal and error handling in OCR outputs.
3. **Embedding Performance**:
   * Switch to more lightweight embeddings (e.g., Hugging Face models) for cost and speed optimization.
4. **Scalability**:
   * Explore distributed FAISS or cloud-based solutions for handling extremely large datasets.
5. **Interactive Query Interface**:
   * Develop a GUI or API endpoint for broader accessibility.

**5. System Flow Diagram**

The process is divided into distinct phases:

1. **Input**: Directory containing PDFs.
2. **Processing**: Text extraction (digital or scanned) and preprocessing.
3. **Vectorization**: Text embedding using OpenAIEmbeddings.
4. **Retrieval**: FAISS retrieves relevant document chunks.
5. **Generation**: OpenAI GPT generates responses based on the query and retrieved chunks.