Document-based Q&A System with Retrieval-Augmented Generation (RAG)

This report outlines the approach, challenges, and potential improvements for developing a document-based Q&A system using Retrieval-Augmented Generation (RAG). This system leverages Azure OpenAI’s `gpt-4o` model for generating responses and FAISS for efficient vector storage and retrieval.

# Approach

**1. Data Extraction and Preprocessing:**

- The system allows users to upload PDF documents, from which text is extracted using `PyPDF2`. Given that PDFs can contain a mix of structured and unstructured content, it was essential to ensure accurate extraction of text data.

- After extraction, the text is divided into smaller chunks, optimizing it for embedding generation and retrieval. This chunking allows the system to focus on smaller content sections, increasing the relevance of retrieved information.

**2. Embedding Generation:**

- Azure OpenAI’s `text-embedding-ada-002` model is used to generate embeddings, converting each text chunk into a high-dimensional vector. These embeddings capture the semantic meaning of the text, making it possible to retrieve the most relevant sections based on a user’s question.

**3. Embedding Storage and Retrieval with FAISS:**

- FAISS (Facebook AI Similarity Search) was chosen for vector storage and retrieval. This library provides a simple yet efficient way to perform similarity searches, which is essential for finding relevant text chunks based on a question embedding.

- To ensure easy retrieval of the original text content, each FAISS vector ID is mapped to its corresponding text chunk in a dictionary.

**4. Answer Generation:**

- Once FAISS retrieves the most relevant chunks, these chunks are passed to Azure OpenAI’s `gpt-4o` model. This model uses the retrieved context to generate a response that is both relevant and contextually accurate.

# Challenges Faced

**1. ChromaDB Integration Issues:**

- Initially, ChromaDB was selected for storing and retrieving embeddings, but integration issues arose, including errors related to tenant connections. These issues necessitated a shift to FAISS, which offered a more straightforward setup with fewer dependencies.

**2. Text Chunking:**

- Determining the optimal chunk size posed a challenge. Using smaller chunks improved retrieval accuracy by isolating specific details, but could lose broader context. Conversely, larger chunks maintained context but reduced retrieval precision. Different chunk sizes were tested to find a balance between detail and context.

**3. Latency in Embedding Generation and Retrieval:**

- Embedding generation with Azure OpenAI’s `text-embedding-ada-002` model introduced latency, especially for longer documents with many chunks. This aspect required optimization to ensure quick response times for user queries.

**4. Incomplete Retrieval Context for Complex Questions:**

- Some questions, especially those with broader or multifaceted requirements, required more than a few top retrievals to cover the full context. Limiting retrieval to three results sometimes led to incomplete answers, highlighting the need for improved retrieval methods for complex questions.

# Potential Improvements

**1. Dynamic Chunking Based on Document Structure:**

- Implementing a chunking approach based on document structure (e.g., sections, paragraphs, headings) rather than fixed text length could help capture context more accurately, enhancing retrieval relevance.

**2. Batch Processing for Embedding Generation:**

- By processing text chunks in batches, the system could reduce latency in embedding generation. This change would make the system more efficient, especially for documents with extensive content.

**3. Hybrid Retrieval Strategy:**

- A hybrid retrieval strategy that combines FAISS with keyword-based filtering could improve initial candidate selection before performing similarity searches. This approach may help narrow down relevant chunks quickly, particularly for larger document sets.

**4. Dynamic Expansion of Retrieval Context:**

- Expanding the retrieval set dynamically for more complex questions would help capture more comprehensive context. This approach could involve recursive querying until a confidence threshold is achieved, ensuring that multi-faceted questions receive well-rounded answers.

**5. Persistent FAISS Indexing:**

- Currently, the FAISS index is in-memory, which means the data is lost between sessions. Introducing persistent FAISS indexing would allow the system to handle larger document sets and maintain document embeddings across multiple sessions.

**6. Model Fine-tuning:**

- Fine-tuning the Azure OpenAI model based on domain-specific question-and-answer pairs could improve response relevance and accuracy, especially if the system is tailored to specific fields such as legal or medical.