Document-Based Question Answering System

Overview

This pipeline processes user-uploaded PDF documents and allows them to ask questions based on the document content. The system retrieves relevant information from the documents and generates answers using CohereAPI and Pinecone's vector search capabilities.

Architecture Components

- **Document Processing:** This step involves loading PDF documents, splitting them into smaller chunks for easier retrieval, and preparing them for embedding generation.
- **Embeddings:** Using Cohere pre-trained embedding model, document chunks are transformed into vector embeddings.
- **Vector Search:** Pinecone’s vector database indexes these embeddings, allowing similarity-based search to retrieve relevant document segments based on the user's query.
- **Generative Response:** Using Cohere language model, the system generates a response by processing the retrieved relevant document segments.

Step-by-Step Pipeline

1. Initialization

Before processing any documents or handling queries, the system needs to initialize the services involved:

- **Cohere Client Initialization:** Initializes the connection to the Cohere API for embedding generation and LLM.
- **Pinecone Initialization:** Initializes Pinecone’s vector database, which stores and retrieves document embeddings for similarity searches.

This is done using environment variables stored in a .env file to keep API keys secure.

2. Document Upload and Preprocessing

The user uploads PDF documents. The system:

- **Extracts Text:** Extracts the text from the uploaded PDF using a PDF loader.
- **Splits the Documents:** Using a text-splitting mechanism (RecursiveCharacterTextSplitter), the system breaks the document text into smaller chunks. Each chunk can overlap slightly with the next to maintain context for retrieving coherent information.

Why Split Documents? Documents are split into manageable chunks (typically 800 characters with a 20-character overlap) because it improves the quality of the embeddings and retrieval, ensuring that the model can extract relevant answers even from large documents.

3. Embedding Generation

Once documents are split into smaller chunks:

- **Cohere Embeddings Model:** Each chunk is converted into a vector embedding using Cohere's embed-english-v2.0 model. These embeddings numerically represent the meaning of each chunk, enabling similarity-based search later.

4. Indexing and Storing in Pinecone

- **Index Creation:** If an index named "qa-bot-cohere" doesn't already exist in Pinecone, it will be created.
- **Embedding Storage:** The document embeddings are stored in Pinecone's index with metadata, enabling fast retrieval when a query is made.

5. Query Handling and Document Retrieval

When the user submits a question, the system:

- **Searches for Similar Documents:** Using Pinecone's vector search, the system retrieves the most relevant document chunks by comparing the query embedding (generated by Cohere) with the stored document

embeddings. The search is cosine-similarity-based, which ensures that the most semantically similar document chunks are returned.

- **Number of Results (k):** The system retrieves the top k results, where k is configurable (in this case, 2).

6. Generative Response Creation

After retrieving the relevant document segments:

- **Cohere LLM Response:** Using the retrieved document segments as context, the Cohere language model generates an answer to the user's query. This answer is based on the content from the document chunks and is returned as a natural language response.

7. User Interaction (Streamlit Integration)

The user interacts with the system through a simple interface built using Streamlit:

- **Document Upload:** Users can upload multiple PDF files through a file uploader.
- **Query Input:** Users type their questions based on the document content.
- **Answer Display:** The system displays the generated answer on the interface.

Streamlining the Pipeline

This document-based QA pipeline is efficient because:

- It splits documents into manageable chunks, improving the quality of both embedding generation and retrieval.
- It uses Pinecone's high-performance vector database for fast and scalable retrieval.
- It utilizes Cohere LLM to generate meaningful, context-aware answers based on document content.

Conclusion

- This system allows for easy document-based question answering using cutting-edge machine learning techniques for document embedding, retrieval, and generation. The pipeline is highly adaptable and scalable, capable of handling multiple documents and large queries efficiently.