**Documentation: Retrieval-Augmented Generation (RAG) Model for QA Bot on P&L Data**

**Overview**

This document provides a comprehensive explanation of the Retrieval-Augmented Generation (RAG) model pipeline. It includes details on the model architecture, data extraction and preprocessing methods, the approach to generating responses, challenges faced during development, and the solutions implemented.

**Model Architecture**

The RAG pipeline combines two key components:

1. **Retriever**:
   * **Purpose**: Extracts the most relevant pieces of information from a document repository.
   * **Implementation**: Utilizes FAISS (Facebook AI Similarity Search) as the vector database to store embeddings and perform similarity searches. Google Generative AI embeddings are employed to convert text chunks into vector representations.
2. **Generator**:
   * **Purpose**: Generates detailed and context-aware responses based on the retrieved documents.
   * **Implementation**: Uses the Gemini-Pro model from Google Generative AI for natural language generation. A custom prompt template ensures the generated responses remain accurate and are rooted in the provided context.

**Approach to Data Extraction and Preprocessing**

**1. Data Extraction**

* **Input Format**: PDF documents.
* **Tool Used**: PyPDF2 library for text extraction.
* **Process**:
  + Each page of the PDF is processed to extract raw text.
  + The text is aggregated into a single string for further processing.

**2. Text Preprocessing**

* **Chunking**:
  + **Tool Used**: RecursiveCharacterTextSplitter from LangChain.
  + **Parameters**: Chunk size = 10,000 characters, Chunk overlap = 1,000 characters.
  + **Purpose**: Breaks long documents into smaller, manageable segments to ensure efficient embedding and retrieval.
* **Vector Embedding**:
  + **Tool Used**: GoogleGenerativeAIEmbeddings.
  + **Process**: Each chunk is converted into a high-dimensional vector representation and stored in the FAISS vector store.

**Generative Response Creation**

1. **Similarity Search**:
   * The vector store is queried using the user’s question to retrieve the most relevant document chunks.
   * FAISS ensures efficient and accurate retrieval based on vector similarity.
2. **Response Generation**:
   * The retrieved documents and user’s question are passed to the generative AI model.
   * **Prompt Template**: Custom-designed to ensure responses are accurate and grounded in the provided context. The model is explicitly instructed not to fabricate information when the answer is unavailable in the context.
3. **Output**:
   * A detailed, contextually appropriate response is generated.
   * If the context does not contain an answer, the model responds with “Answer is not available in the context.”

**Challenges and Solutions**

**1. Challenge: Extracting Clean Text from PDFs**

* **Problem**: PDF files often contain complex layouts, including headers, footers, and tables, which can disrupt text extraction.
* **Solution**: Implemented preprocessing steps to remove unnecessary characters and validate extracted text integrity. Tested multiple PDF parsing libraries for optimal results.

**2. Challenge: Managing Large Document Sizes**

* **Problem**: Some documents exceeded the token limits of the generative model.
* **Solution**: Introduced chunking with overlap to ensure no critical information was lost while maintaining coherence across document boundaries.

**3. Challenge: Ensuring Response Accuracy**

* **Problem**: The generative model sometimes produced incorrect or hallucinated answers.
* **Solution**:
  + Customized the prompt template to emphasize accuracy and discourage fabrication.
  + Used similarity search to ensure only relevant context was fed into the model.

**4. Challenge: Performance Bottlenecks in Retrieval**

* **Problem**: The similarity search process could be slow with large vector stores.
* **Solution**: Optimized FAISS indexing by experimenting with index types and implementing hierarchical clustering for faster queries.

**Conclusion**

The RAG pipeline effectively combines retrieval and generation to deliver accurate and contextually relevant answers. By leveraging FAISS for retrieval and Google Generative AI for response generation, the model achieves a robust and scalable architecture. The solutions implemented to address common challenges ensure reliability and high performance, making the pipeline suitable for various real-world applications.

**Example Query 1:**

**Query**: "What is the total revenue for the year ending March 31, 2024?"

**Response**: "The total revenue for the year ending March 31, 2024, is ₹158,381 crore, as reported under the revenue section."

**Example Query 2:**

**Query**: "What is the profit after tax for the year ending March 31, 2024?"

**Response**: "The profit after tax for the year ending March 31, 2024, is ₹26,248 crore, as listed under the profit and loss statement."

**Example Query 3:**

**Query**: "How much cash and cash equivalents does the company hold as of March 31, 2024?"

**Response**: "The company holds ₹14,786 crore in cash and cash equivalents as of March 31, 2024, as noted in the balance sheet."

**Example Query 4:**

**Query**: "What are the total assets as of March 31, 2024?"

**Response**: "The total assets as of March 31, 2024, amount to ₹137,814 crore, as reported in the condensed consolidated balance sheet."

**Example Query 5:**

**Query**: "What is the equity attributable to shareholders as of March 31, 2024?"

**Response**: "The equity attributable to shareholders as of March 31, 2024, is ₹88,116 crore, as shown in the equity section of the balance sheet."

**Example Query 6:**

**Query**: "What is the basic earnings per share (EPS) for the year ending March 31, 2024?"

**Response**: "The basic earnings per share (EPS) for the year ending March 31, 2024, is ₹63.39."

**Example Query 7:**

**Query**: "What is the depreciation and amortization expense for the year ending March 31, 2024?" **Response**: "The depreciation and amortization expense for the year ending March 31, 2024, is ₹4,678 crore, as reported under expenses."

**Example Query 8:**

**Query**: "What is the revenue from operations for the three months ending March 31, 2024?"

**Response**: "The revenue from operations for the three months ending March 31, 2024, is ₹37,923 crore, as detailed in the statement of profit and loss."