Part 1: Retrieval-Augmented Generation (RAG) Model for QA Bot on P&L Data

Problem Statement:

Develop a Retrieval-Augmented Generation (RAG) model for a Question Answering (QA) bot that can process financial terms and insights from a Profit & Loss (P&L) table extracted from PDF documents.
The QA bot should retrieve relevant information related to income, expenses, profit margins, and other key financial metrics from the provided P&L table and generate accurate and coherent responses.

Model Architecture and Approach

The pipeline consists of three core components:

1. Data Extraction and Preprocessing

Input: Financial data as a pdf file

Steps:

- Extract relevant pages that contain Profit & Loss table from financial PDFThis is done by scanning the PDF using pdfplumber library to identify the relevant page(s) containing "Statement of Profit and Loss," "Revenue," and "Expenses" word.
  Since the contents page also has these words, the page with P&L table will be the second relevant page containing the above said words.
- Parse structured financial tables using `Camelot`. The identified page number is used to extract the P & L table.
- Cleans and formats the extracted data into a structured `pandas` DataFrame. This is done by removing unnecessary rows/columns, splitting multi-line values into separate columns and standardizing column names.

Output: A structured pandas DataFrame Profit & Loss table data.

2. Embedding-Based Information Retrieval(Vector Search with ChromaDB)

Model Used**: all-MiniLM-L6-v2** (SentenceTransformer)
Vector database used: **Chromadb**

Steps:

- Embedding Generation: Convert extracted financial data into dense vector embeddings using selected Sentence Transformers.
- Vector Storage in ChromaDB:Store the generated embeddings in ChromaDB, a vector database for efficient similarity search alongside their corresponding row metadata.
- Query Processing and Retrieval Mechanism: Convert the user query into an embedding using the same model. Use k-nearest neighbor (KNN) search to retrieve the most relevant financial data rows based on query similarity.

Output: Retrieved financial data segments

3. Generative AI-Based Answering

LLM Model Used: **DeepSeek Coder 1.3B Instruct** (LLM)

Use the LLM model to generate answers based on retrieved financial context. The model used is deepseek-coder-1.3b-instruct, a transformer-based causal model trained for reasoning over structured data.

Steps:

- Context Preparation: The retrieved financial data rows are formatted into a structured text context. The context and query are combined to form a prompt which will be input to the LLM model.
- Response Generation: The prompt is tokenized using AutoTokenizer and autocast is used for optimizing computation on MPS/CUDA. The LLM model generates responses using the tokenized prompt input by setting temperature and max length of tokens which controls response length.

Output: Answer to the query and the retrieved financial data segments.

End-to-End Flow

1.Extract P&L page → 2. Parse financial table → 3. Embed & store in ChromaDB →

4.Encode query → 5. Retrieve relevant rows → 6. Generate LLM response.

This architecture ensures efficient financial data querying with structured retrieval and generative AI reasoning.

Challenges and solutions

1. Extracting P&L table from the input pdf document.'

The PL table usually contains the words “Statement of Profit and Loss" , "Revenue" and "Expenses”. This logic is used to extract the relevant pages. Another point is that the “Contents” page which appears in the beginning of the document will also be having these terms. Hence the page with the PL table will be the second one having all the said words. This logic is used to extract the correct page with the PL table.

2. Preprocessing the table. P&L table format may vary across documents.

Unwanted rows and columns are removed.. The columns are also edited to make a coherent structure.

3. Choosing the vectorbase.

Different open source vector base approaches were tried such as FAISS and Chromadb. ChromaDB is selected as it is designed for AI and Retrieval-Augmented Generation (RAG)applications. It is optimized for fast similarity search and can store, index, and retrieve embeddings efficiently.

4. Choosing the llm model.

Initially the TAPAS model, which stands for Tabular Pretrained Language Model, which is a deep learning model designed for handling tabular data (data presented in tables, often seen in spreadsheets or databases). But deepseek models are better open-source alternatives with higher efficiency.

5. Augmenting the context.

According to the llm ie deepsake model, the context was created by combining query with retrieved financial data converted into string format.

6. Slow inference. Inference was taking more than 30 seconds.

Used multithreading especially for multiple queries. Also GPU/MPS if available was used.