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

Retrieval-Augmented Generation (RAG) model can be used as a Question Answering (QA) bot that can process financial terms and insights from a Profit & Loss (P&L) table extracted from PDF documents.

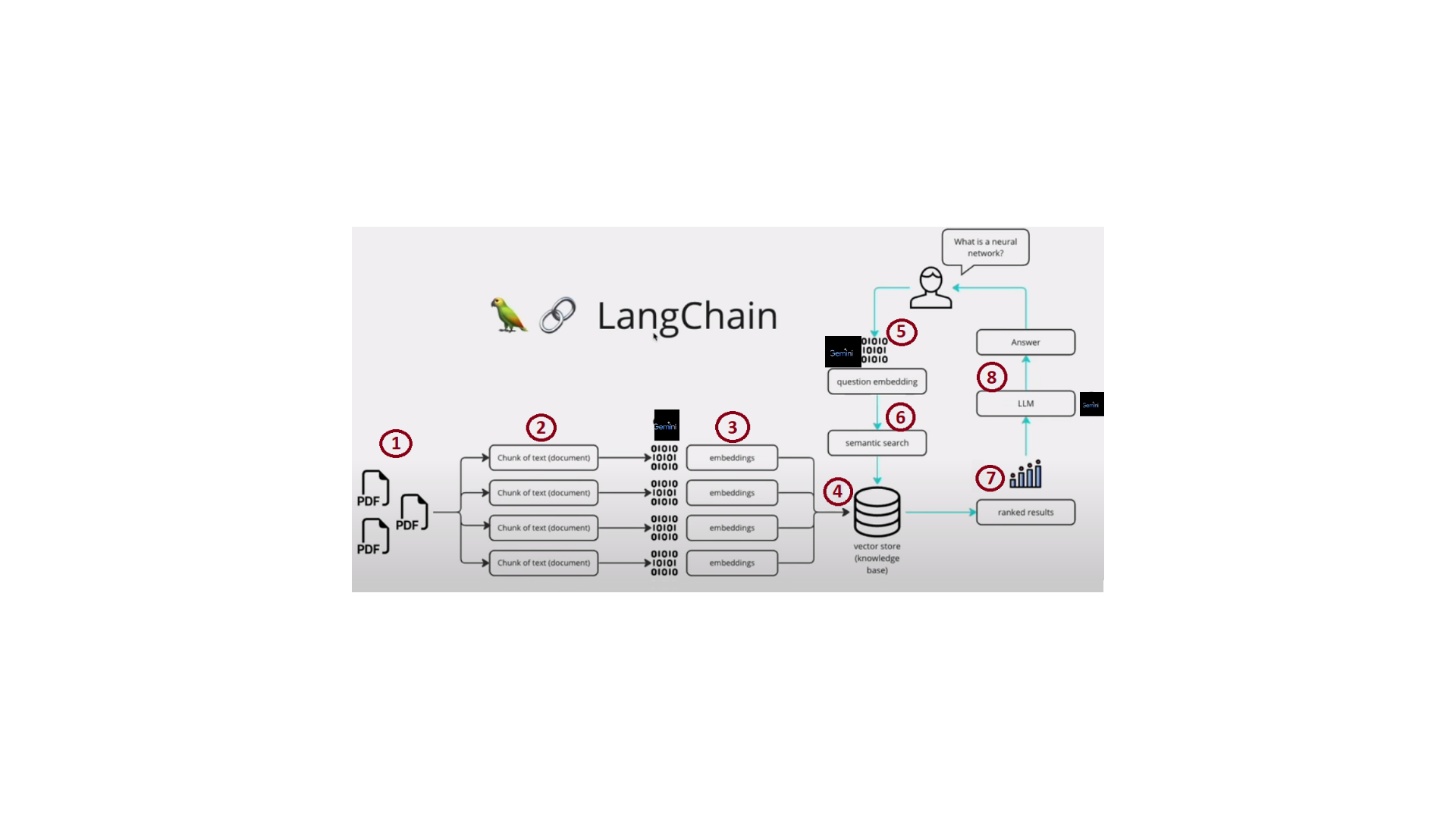
This QA bot will retrieve relevant information related to income, expenses, profit margins, and other key financial metrics from the provided P&L table and generate accurate and coherent responses.

You will be able to Upload PDF documents according to your will and ask the QA bot questions related to it. Once you have uploaded the document, RAG model will save the file and then the “langchain\_text\_splitters” will load and chunk its content for the Embedding model to create embeddings of it and store it in a vector database from where it will be easy to fetch the user queries.

**1.RAG Model Architecture:**

A blue and white logo

Description automatically generatedA blue and white logo

Description automatically generated

1. Large Language Model: The **ChatVertexAI** model (gemini-1.5-pro-001) is used as the core generative model. This model processes user queries and generates contextual responses based on the retrieved information.
2. Embeddings: **VertexAIEmbeddings** (text-embedding-005) are used to create vector embeddings of the documents. These embeddings help represent text in a numerical form, enabling similarity-based retrieval.
3. Vector Database: **InMemoryVectorStore** stores the embeddings. This allows for fast similarity search, enabling retrieval of the most relevant document chunks to the query.

**2. Data Extraction and Preprocessing:**

1. File Uploading: Files are uploaded via the Gradio interface and saved in a defined directory. The uploaded files are loaded, and their text content is chunked into manageable sizes (1,000 characters with 200 overlaps).
2. Document Loading and Splitting: Files (i.e. PDFs) are processed using DirectoryLoader and chunked into smaller, overlapping segments with RecursiveCharacterTextSplitter. This ensures proper indexing and retrieval of content.
3. Embedding Creation: The document chunks are converted into embeddings using the VertexAIEmbeddings model.
4. Indexing: The embeddings are added to the InMemoryVectorStore to enable similarity-based search.
5. Query-Driven Retrieval: When a user submits a question, the vector store retrieves the most relevant chunks based on the query’s embedding.

**3. Generating Responses:**

1. Contextual Prompting: A pre-defined prompt (from LangChain’s hub) is used to structure the LLM's input.
2. Generative Model: The **ChatVertexAI** (gemini-1.5-pro-001) model generates responses by leveraging the provided context and user query.
3. Response: The final answer is returned to the user via the Gradio interface.

**User Guide:**

1. A screenshot of a computer

   Description automatically generatedUploading Documents: Navigate to the "Upload Files" Section from the Gradio Interface. Upload your file by clicking on “Click to Upload” and then click on “Upload and Process” button.

A screenshot of a computer

Description automatically generatedOnce you have uploaded the document wait for the RAG model to process it. As soon as the Model has completed the backend processes it will give you the output as “File saved Successfully”.

1. Question-Answering: Once you have completed the process of uploading the documents, Navigate to the Ask questions tab. Now you can ask the RAG model questions related to the document you have uploaded.

A screen shot of a computer

Description automatically generatedHere is set of examples of user queries related to the Sample document provided by Sample Set.

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Challenges Encountered:

1. As we know OpenAI’s ChatGPT is one of the best Large Language Model(LLM) out there, but they have pay to use service for API Integration. Therefore, I had to switch to Google Cloud’s Vertex AI. Google Cloud provides free credits for a certain amount of time. Vertex AI might not be as good as OpenAI but it gets our work done just enough.
2. While using the Vertex AI’s (text-embedding-004) Embedding model, I faced a few issues of not getting proper results. So had to switch to (text-embedding-005) Embedding model.
3. Encountered a few problems while creating an interface for the RAG model using Gradio. I took references from StackOverflow, ChatGPT, Gradio documentation, etc. and got the idea on how to solve those problems.