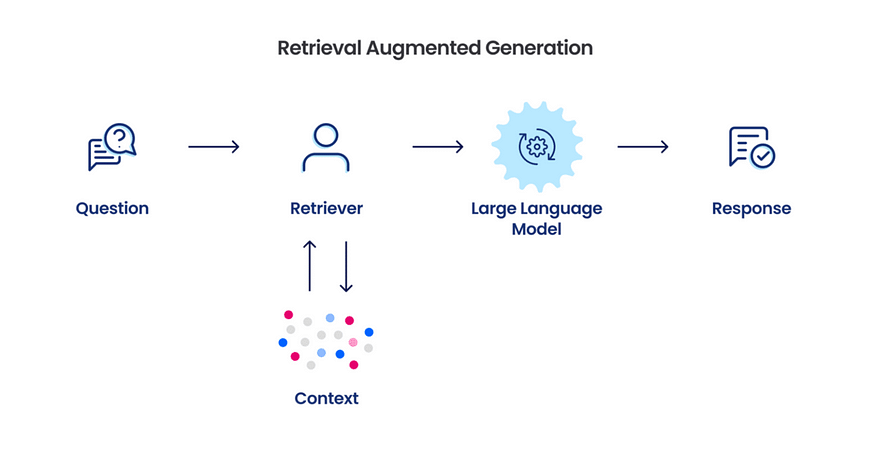
**CSV Miner Using Basic RAG**

**Developed by Rishi Modgil**

RAG combines classic language models with a novel twist by incorporating information retrieval right into the creation process.

RAG stands for **retrieval-augmented generation.** Responses from traditional language models are produced using information that have already been learnt during the training stage. But because of the inherent limitations of the data they were trained on, these models frequently produce answers that may be shallow or lack specialized knowledge and start hallucinating when asked something about the topic they don’t know about. By importing external data as required throughout the creation process, RAG overcomes this issue of hallucination. In order to inform and guide the creation of the response, the RAG system first extracts information from a dataset or knowledge base when a question is submitted.



This project uses RAG to interact with CSV data. It allows users to ask natural language questions about the contents of a CSV file, and the system generates SQL queries to extract the relevant information, with integrated Google Generative AI and combining document loaders, embeddings, and vector stores for efficient data retrieval.

**Approach for the assignment**

1. **Dataset Selection**:
   * Chose a publicly available retail dataset (Kaggle’s Sales dataset) which contained multiple columns such as product details, sales, and customer information, order number etc.
2. **Embedding Generation**:
   * Used a pre-trained model (GoogleGenerativeAIEmbeddings) to generate vector representations for relevant data columns.
   * Stored embeddings in an efficient format (using FAISS).
3. **Query Parsing**:
   * Integrated an LLM (gemini-1.5-pro) to interpret queries.
4. **RAG Pipeline Implementation**:
   * Used the query to perform similarity matching with the dataset embeddings.
   * Created a meaningful response by combining the obtained data with reasoning produced by LLM.
5. **Output Generation**:
   * Designed the system to return a concise and human-readable answer to the query.

**Assumptions**

1. **Dataset Quality**:
   * Assumed the dataset is sufficiently clean and contains the required columns.
2. **LLM Capability**:
   * Assumed the chosen LLM can interpret natural language queries effectively.
   * Assumed embeddings accurately capture the semantic relationships in the data.
3. **Query Complexity**:
   * Focused on straightforward queries related to sales, products, or customers and avoiding the overly complicated queries.

**Learnings from this assignment**

1. **RAG Workflow**:
   * Learned to integrate a Retrieval-Augmented Generation pipeline effectively by combining embeddings with LLM reasoning.
2. **Embedding Management**:
   * Understood the importance of efficient storage and retrieval of embeddings (FAISS).
3. **Query Interpretation**:
   * Gained insights into parsing natural language queries and mapping them to structured csv data.
4. **LLM Integration**:
   * Realized the potential of LLMs to enhance data retrieval and provide responses.

This assignment provided practical experience in building intelligent systems that combine machine learning and natural language processing for data mining tasks.

**How to make this project run on your machine?**

Step 1: Install [Jupyter Notebook](https://jupyter.org/install)

Step 2: Get an API key of Google Gemini from [here](https://aistudio.google.com/app/apikey)

Step 3: Install the dataset provided with this project

Step 4: install the libraries listed in the requirements.txt file which is present in the folder

Step 5: Run the ragCSV.ipynb file in Jupyter Notebook

Step 6: Inside the ragCSV file, enter the input in the line:

answer = rag\_chain.invoke({"input": "*enter the desired input here*"})

*Disclaimer: To run any separate csv file with this model, one needs to modify the system prompt slightly to match the dataset.*

If any issues are faced regarding the project, please contact me at [rishimodgil120@gmail.com](mailto:rishimodgil120@gmail.com)