{SAMPLE SET}

Gen AI Engineer / Machine Learning Engineer Assignment

Assignment - Part 1

Documentation: Retrieval-Augmented Generation (RAG) Model for QA Bot

1. Overview

The primary goal of Part 1 is to process financial data from P&L data and prepare it for retrieval and generation tasks. The process involves extracting structured data, generating embeddings, and building a FAISS index for efficient retrieval. The extracted data enables the QA bot to answer financial queries interactively.

2. Model Architecture

a. Components

1. Text Extraction:
- Used PyPDF2 to extract text from the financial statement PDF.
- Preprocessed the raw text to identify key metrics and values.

2. Structured Data Creation:
- Regular expressions (re module) were applied to extract financial metrics such as Revenue, Profit, Cost, etc., along with their respective amounts.
- Structured data was saved in a CSV file for downstream processing.

3. Embedding Generation:
- Leveraged the SentenceTransformer model (all-MiniLM-L6-v2) to generate vector embeddings for the financial metrics and values.
- Stored embeddings in a NumPy file for efficient use.

4. FAISS Index:
- Created a FAISS (Facebook AI Similarity Search) index for efficient nearest-neighbor search on the generated embeddings.
- The index enables quick retrieval of relevant financial data for user queries.

3. Approach to Data Extraction and Preprocessing

a. Data Extraction

Tool: PyPDF2

- Extracted text from each page of the PDF file. Saved it as a plain text file for further processing.

b. Preprocessing

1. Identified relevant sections in the extracted text using regular expressions.
2. Extracted financial metrics and their values (e.g., Profit: 2021, Revenue: 10,000).
3. Filtered the extracted data to remove irrelevant entries and noise, ensuring only meaningful metrics are retained.

c. Embedding Generation

- Combined financial metrics and values (e.g., 'Profit: 2021') as text.
- Generated embeddings using a pre-trained SentenceTransformer model.
- Stored embeddings for later use in the retrieval system.

d. FAISS Index Construction

- Initialized a FAISS index (IndexFlatL2) with the embedding dimensions.
- Added embeddings to the index to support similarity-based retrieval for user queries.

4. Challenges and Solutions

1. Handling Large PDF Files:
- Challenge: Processing large PDFs caused memory issues.
- Solution: Extracted text in smaller chunks and optimized the text processing pipeline.

2. Irregular Text Extraction:
- Challenge: Financial data in PDFs often has inconsistent formatting.
- Solution: Developed robust regular expressions to identify metrics and values reliably.

3. Indexing Large Embeddings:
- Challenge: Memory limitations during FAISS index creation.
- Solution: Reduced embedding size using a lightweight model (all-MiniLM-L6-v2) without compromising accuracy.

4. Data Cleaning:
- Challenge: Extracted text contained irrelevant information and noise.
- Solution: Applied thresholds (e.g., filtering values below 100) to retain only meaningful financial entries.

5. Outputs

1. Structured Data:
- Cleaned and organized data stored in CSV format.
  Example entries:

Metric	Amount
Profit	2021
Revenue	10000
Depreciation	200

2. Embeddings:
- High-dimensional vectors representing financial metrics and their values.
- File saved: embeddings.npy

3. FAISS Index:
- Efficient index for querying financial data based on similarity.
- File saved: faiss_index

6. Example Queries and Outputs

Below are some example queries and their corresponding outputs from the Financial QA Bot:

- **Query**: What is the profit?
- **Output**: "Profit for the period is INR 7,975 crore."
- **Query**: What is the total revenue?
- **Output**: "Total revenue from operations is INR 37,923 crore."
- **Query**: What is the cost of sales?
- **Output**: "Cost of sales amounts to INR 26,748 crore."
- **Query**: What is the operating profit?
- **Output**: "Operating profit stands at INR 7,621 crore."

7. Deployment

The application is available here

8. Conclusion

The development of the data processing pipeline in Part 1 establishes a solid foundation for creating a robust financial QA bot. By integrating advanced tools like **PyPDF2** for text extraction, **SentenceTransformer** for embedding generation, and **FAISS** for efficient indexing, the system ensures seamless handling of financial data from unstructured PDFs. This pipeline not only enables accurate data extraction but also prepares the dataset for sophisticated retrieval and generative tasks, ensuring that the QA bot provides reliable and contextually accurate answers. The structured approach and modular design make this system scalable, adaptable, and ready for deployment in real-world financial analysis scenarios.