Documentation Report for Chatbot Training Code Using LLM

This documentation provides a detailed overview of the Python code used to train a chatbot utilizing Large Language Models (LLMs) through various Python libraries including langchain, unstructured, and FAISS. The process involves extracting and categorizing data from PDF documents, summarizing texts and tables, and embedding these summaries into a vector database for efficient retrieval.

Overview

The chatbot is designed to handle financial information queries, particularly in a banking context. It processes PDF documents, extracts textual and tabular data, summarizes the contents, and stores these summaries in a FAISS vector database for efficient retrieval during query answering.

Dependencies

- uuid: Generates unique identifiers for each document.
- unstructured: Used for extracting elements from PDF files.
- langchain: Integrates LLM capabilities including embeddings, retrievers, and chat models.
- FAISS: Manages the vector storage for efficient retrieval.

Workflow

1. **PDF Partitioning and Data Extraction:**
    - **The partition_pdf function from the unstructured library is used to extract and partition elements from a PDF file based on titles, with options to handle images and table structures.**
2. **Data Categorization:**
    - **Extracted elements are categorized into texts and tables. This separation is crucial for applying appropriate processing techniques to different types of data.**
3. **Data Summarization:**
    - **Texts and tables are summarized using a LLM, specifically designed to optimize the summaries for retrieval purposes. This involves constructing a chatbot prompt to generate concise summaries.**
4. **Vector Database Creation:**
    - **Summarized texts and tables are converted into Document objects, which are then used to create a FAISS vector store. This database enables fast retrieval of documents based on their semantic content.**
5. **Query Handling and Contextual Response Generation:**
    - **A LLMChain is used to handle queries. It fetches relevant documents from the vector database and constructs a context for the LLM to generate informed and accurate responses.**

Detailed Code Breakdown

Data Extraction and Categorization

- **partition_pdf function is utilized to extract data from the specified PDF file. It is configured to ignore images, infer table structures, and manage text chunking based on character limits.**
- **Extracted data is categorized by type (text or table) to facilitate separate processing.**

Summarization Process

- **Summaries are generated using a chain of operations including a customized prompt template and a LLM (gpt-4). Summarization is performed only if explicitly requested, allowing for flexibility in processing.**

Vector Database Setup

- **Each summarized piece of text or table is encapsulated into a Document object, with each document receiving a unique identifier.**
- **These documents are then embedded into a FAISS vector store using OpenAIEmbeddings for the embeddings. The vector store is saved locally for subsequent retrieval.**

Query Answering

- **The final segment of the code loads the FAISS database and sets up a LLMChain for answering queries based on the embedded summaries.**
- **The answer function retrieves relevant documents based on the query, constructs a context, and uses the LLMChain to generate a response.**

Conclusion

This code effectively leverages advanced NLP techniques and vector space modeling to create a chatbot that can intelligently handle queries by retrieving and synthesizing information from a structured database of summarized documents. This approach ensures efficient and accurate responses to user queries, making it highly suitable for applications requiring quick access to detailed and complex information, such as in financial analysis and banking.