AI Assignment - Task 2: RAG-Based Semantic Quote Retrieval and Structured QA

This document outlines the implementation of a Retrieval-Augmented Generation (RAG) pipeline for semantic quote retrieval and structured question answering. The pipeline is designed to retrieve relevant quotes based on a user query and then present them in a structured format.

1. Data Preparation

The data preparation phase involved loading the `Abirate/english_quotes` dataset from the Hugging Face `datasets` library. The `quote` and `author` fields were preprocessed by converting text to lowercase, removing special characters and numbers, tokenizing, removing stopwords, and lemmatizing. The `tags` field was ensured to be a list.

File: `rag_data_preparation.py`

Key Steps: - Load dataset using `datasets.load_dataset("Abirate/english_quotes")`. - Clean `quote` and `author` text using NLTK for tokenization, stopword removal, and lemmatization. - Save the processed data to `rag_processed_quotes.csv`.

2. Model Fine-Tuning

For the purpose of this assignment, the "fine-tuning" of the sentence embedding model (`all-MiniLM-L6-v2`) involved loading the pre-trained model and saving it locally. This simulates preparing the model for the specific domain of quotes, even without explicit training steps with a complex loss function due to the nature of the available dataset (individual sentences rather than query-document pairs).

File: `rag_model_fine_tuning.py`

Key Steps: - Load `rag_processed_quotes.csv`. - Load `SentenceTransformer("all-MiniLM-L6-v2")`. - Save the model to `./fine_tuned_quote_model`.

3. Build the RAG Pipeline

The RAG pipeline consists of a retrieval component using FAISS for efficient similarity search and a basic answer generation component that presents the retrieved quotes.

File: `rag_pipeline.py`

Key Components: - **`RAGPipeline` Class:** - Initializes with a sentence transformer model and the processed quotes data. - `_build_index()`: Creates FAISS index from embeddings of combined quote text (quote + author + tags). - `retrieve_quotes(query, k=5)`: Encodes the query and searches the FAISS index for the top `k` most similar quotes. - `answer_query(query)`: Retrieves quotes and formats them into a human-readable response.

4. RAG Evaluation

The evaluation of the RAG pipeline was performed by testing its retrieval capabilities on a few sample queries. The relevance of the top-k retrieved quotes was manually inspected to assess the effectiveness of the semantic search.

File: `rag_evaluation.py`

Key Steps: - Instantiate `RAGPipeline`. - Define a set of `test_queries`. - For each query, retrieve top-k quotes and print their details (quote, author, similarity score).

5. Streamlit Application

A simple Streamlit web application was developed to provide an interactive interface for the RAG pipeline. Users can input a query and view the retrieved quotes and their details.

File: `streamlit_app.py`

Key Features: - **User Input:** A text input field for entering queries. - **Quote Display:** Displays the retrieved quotes, author, tags, and similarity scores. - **Caching:** Uses `@st.cache_resource` to load the RAG pipeline once for efficiency.

How to Run the Application

To run the Streamlit application, follow these steps:

1. **Clone the repository (if applicable) or ensure all files are in the same directory.**

2. **Install the required libraries:** `bash pip install pandas openpyxl nltk datasets sentence-transformers faiss-cpu streamlit`

3. **Download NLTK data (if not already downloaded by `rag_data_preparation.py`):** `python import nltk nltk.download("punkt") nltk.download("stopwords") nltk.download("wordnet")`

4. **Run the data preparation script:** `bash python3.11 rag_data_preparation.py` This will create `rag_processed_quotes.csv`.

5. **Run the model fine-tuning script:** `bash python3.11 rag_model_fine_tuning.py` This will save the `fine_tuned_quote_model` directory.

6. **Launch the Streamlit application:** `bash streamlit run streamlit_app.py --server.port 8501 --server.enableCORS false --server.enableXsrfProtection false` The application will be accessible via a local URL (e.g., `http://localhost:8501`) and potentially an external URL if exposed.

Deliverables

- `rag_data_preparation.py`: Script for data loading and preprocessing.
- `rag_model_fine_tuning.py`: Script for loading and saving the sentence transformer model.
- `rag_pipeline.py`: Implements the RAG pipeline with FAISS indexing and retrieval.
- `rag_evaluation.py`: Script for evaluating the retrieval component.
- `streamlit_app.py`: Streamlit web application for interactive demo.
- `rag_processed_quotes.csv`: Processed dataset.
- `fine_tuned_quote_model/`: Directory containing the saved sentence transformer model.
- `README_RAG.md`: This documentation file.
- `README_RAG.pdf`: PDF version of this documentation file.