Vector Database Setup Documentation

Overview

This document provides detailed insights into how the vector database is configured and how embeddings from the resumes are stored and utilized. The system processes resume in different formats (CSV and PDF) and uses machine learning embeddings to facilitate efficient querying and retrieval based on user questions.

Step 1: Resume Data Extraction

Resumes are extracted from a CSV file and PDFs stored in a specified directory. The CSV file contains basic data, including an ID and Category, while the actual text content may either be stored directly in the CSV or extracted from the corresponding PDF.

Expected Input:

- CSV file containing ID, Resume_str, and Category.
- PDF folder with PDF files organized by category.

Step 2: Preprocessing Text Data

The extracted text is pre-processed by removing special characters and converting text to lowercase. This step ensures that the embeddings are generated from clean, normalized data.

Step 3: Embedding Generation

The pre-processed resume texts are embedded using the HuggingFaceEmbeddings model (all-mpnet-base-v2), which converts textual data into high-dimensional vectors.

Expected Output: An embedding matrix of shape (number_of_documents, embedding_dimension).

Step 4: Indexing with FAISS

The FAISS (Facebook AI Similarity Search) library is used to create an efficient vector index for similarity searches.

- The code initializes a FAISS index for L2 (Euclidean) distance.
- It then moves the index to the GPU for faster computation and adds the embeddings.

Index Type:

- IndexFlatL2: Uses L2 distance for similarity search.
- gpu_index: Utilizes GPU for accelerated searches.

Expected Output: A FAISS index ready to handle nearest neighbor searches for incoming user queries.

Step 5: Query Handling and Retrieval

The system processes user queries by:

1. Embedding the user’s question.
2. Searching for similar resumes using FAISS.
3. Using BM25 (Okapi) to perform a secondary relevance ranking.

Search Output:

- Top 5 results from FAISS and BM25 combined.
- Ranked and filtered to create a unique set of relevant results.

Step 6: Response Generation

The InferenceClient from HuggingFace generates the final response based on the top-ranked resume excerpts using a large language model.

Dependencies

- **Libraries Used**: PyPDF2, pandas, FAISS, langchain, HuggingFaceEmbeddings, BM25Okapi, huggingface_hub, re.

Expected Environment Variables

- HUGGINGFACEHUB_API_TOKEN: API token for accessing HuggingFace's inference services.

This configuration allows efficient storage, retrieval, and ranking of resume embeddings to handle complex user queries using state-of-the-art machine learning models and similarity search techniques.