Legal Search Embedding Pipeline - Milvus

This document outlines the entire process followed to integrate legal search embeddings using Milvus, Sentence Transformers, and FastAPI in the Open WebUI project. It covers file placements, model configuration, embedding generation, debugging issues, and testing the query functionality.

Server Implemented on:
(ec2-13-200-168-201.ap-south-1.compute.amazonaws.com)

(Site: http://13.200.168.201:8080/, http://13.200.168.201:5173/)

1) Placing the Files & Setting Up the Project Structure:

❖ Received Files from a colleague:

- Central_acts_DB.py
- SC_judgments_DB.py
- docker-compose.yml
- milvus.yaml
- query.py
- volumes.zip
- Central_acts.csv
- SC_metadata_judgments_content.csv

❖ Added Files:

- test.py
- extract_100_rows.py

➔ These provided files were placed in /home/ubuntu/open-webui-0.4.8/vector_db/embeddings with two respective folders “code” & “data”

➔ code/ Directory (for all scripts and configurations) there are Central_acts_DB.py (Script to create embeddings and store Central Acts in Milvus.), SC_judgments_DB.py (Script to create embeddings and store Supreme Court Judgments in Milvus.), docker-compose.yml (Configuration to run Milvus as a containerized service.), milvus.yaml (Configuration settings for Milvus.), query.py (Script to query the stored embeddings and test retrieval.), test.py (Script to validate whether the Milvus collections were correctly created and accessible.), volumes.zip (Contained data for Milvus volumes but was not extracted due to recurring rocksmq errors.) and venv.

➔ **data/ Directory (for dataset files) there are Central_acts.csv (Processed dataset containing legal statutes with first 100 rows), Central_actsss.csv (Original full dataset before processing.), SC_metadata_judgments_content.csv (Processed dataset containing Supreme Court case metadata with first 100 rows), SC_metadata_judgments_contentt.csv (Original full dataset before processing.) and extract_100_rows.py (A script to extract first 100 rows from the original datasets for testing.)**

- **I placed all received files into their respective directories. Initially, I tried extracting the volumes.zip folder a couple of times and ran sudo docker compose up -d before the extraction process but when running python3 test.py it showed no collections are present in milvus db as Milvus showed persistent rocksmq errors related to missing storage files & a corrupted file even though the volumes.zip folder size was 31GB. Instead, I opted to recreate the Milvus collections manually by running embedding scripts.**

- **To ensure testing was efficient, I ran extract_100_rows.py to create Central_acts.csv and SC_metadata_judgments_content.csv with only 100 rows each. This allowed us to test embedding generation on a smaller dataset before running it on a full dataset.**

- **Before running any script, I checked if collections already existed in Milvus. I manually dropped any existing collections (acts_db, final_db) using:**

- **from pymilvus import connections, utility**
  connections.connect(host="127.0.0.1", port="19530")
  if utility.has_collection("acts_db"):
  utility.drop_collection("acts_db")
  if utility.has_collection("final_db"):
  utility.drop_collection("final_db")
  print("Collections cleaned up! Now you can re-run your script.")

- **This step ensured no old embeddings caused conflicts when inserting new ones.**

- **We executed python3 Central_acts_DB.py to generate embeddings for acts_db and then executed python3 SC_judgments_DB.py for final_db after that I encountered dimension mismatch errors, which were due to different embedding models:**

- **Central_acts_DB.py used MiniLM (384 dimensions) & SC_judgments_DB.py used mixedbread-ai/mxbai-embed-large-v1 (1536 dimensions). To resolve this, I standardized both scripts to use mixedbread (mxbai-embed-large-v1, 1024 dimensions).**

- After embeddings were created, I ran test.py to validate Milvus collections and ensure stored vectors were correctly indexed and then ran query.py to perform test searches against the embedded legal texts.

- With embeddings successfully stored in Milvus, we registered the legal search API route in open_webui/main.py using:

- from open_webui.apps.legalsearch.main import router as legalsearch_router
- app.include_router(legalsearch_router, prefix="/api/legalsearch", tags=["Legal Search"])

- Updated the frontend API logic in index.ts and also Integrated the search input field & UI logic in +page.svelte to allow users to submit queries and fetch relevant legal documents. Initially, when submitting queries, we encountered a 405 Method Not Allowed error, which was fixed by ensuring correct API route registration.

- After correctly aligning the embeddings, re-running the scripts, and properly registering API routes, the legal search feature started working smoothly.

- The docker-compose.yml file is used to set up Milvus and its required services, such as etcd (for metadata storage) and Minio (for object storage) as our server does not have a GPU, so we had to disable GPU-related settings and used the CPU-only version of Milvus by specifying milvusdb/milvus:v2.5.4 commented out the NVIDIA GPU configurations under deploy:

- How our data is stored in mivlus standalone When I ran the embedding scripts, the following vectors were generated from legal documents using model = SentenceTransformer("mixedbread-ai/mxbai-embed-large-v1", device="cpu") Milvus API was used to insert these vectors into acts_db (For Central Acts Data.) & final_db (For Supreme Court Judgments.) The vector data is stored inside Milvus Standalone (milvus-standalone) under /var/lib/milvus while metadata is managed by etcd, and minio which is not actively used in our current setup.

- In our embedding scripts (Central_acts_DB.py and SC_judgments_DB.py), the connection to Milvus was established using:

- connection = connections.connect(host="127.0.0.1", port=19530)

- Instead of connection = connections.connect(host="milvus-standalone", port=19530) Since I executed the embedding scripts directly on the host machine (outside the Docker network), Docker's internal service name (milvus-standalone) is not resolvable. Instead, we use 127.0.0.1, which maps to the local machine where Milvus is running. In our Docker Compose setup, I mapped Milvus' internal

port 19530 to the host machine This means we can directly access Milvus on localhost (127.0.0.1), even if it is running inside a container. If we were running the embedding script inside a Docker container, then we would need to use: connection = connections.connect(host="milvus-standalone", port=19530) This is because Docker containers use an internal network where services are identified by their container names (e.g., milvus-standalone).

- The embeddings for Central_acts_DB.py and SC_judgements_DB.py were successfully created using the mixedbread-ai/mxbai-embed-large-v1 model. In main.py, we used a similar implementation to the one found in the ronnyopenthirteen image. Now, the search functionality is working seamlessly, retrieving results that include the document ID, document text, and metadata. The metadata contains details such as the case title and judgment date for legal searches and statute details for statute searches, depending on the search_type parameter.

- During testing, we encountered an issue where case titles were missing in the retrieved results. Upon inspecting both CSV files, we found that there was no dedicated case_title column. However, in SC_metadata_judgments_content.csv, the file_name column appeared to contain case titles. For Central_acts.csv, this issue was irrelevant since it only contains legal statutes and does not require case titles. The current implementation retrieves legal documents directly from Milvus, and the LLM-based summarization using llm_client = HuggingFaceHub(repo_id="NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO", model_kwargs={"temperature": 0.2, "max_length": 10000}) is not currently in use. The code does include a commented-out section that, when enabled, takes the retrieved legal documents, summarizes them using the LLM, and then provides a detailed answer with document references.

2) Steps to Running the Legal Document Retrieval System from the Beginning:

- After receiving the necessary files, they were placed in the correct directory structure:

➔ /home/ubuntu/open-webui-0.4.8/vector_db/embeddings/

- Inside this directory:

➔ code/ – Contains scripts related to database initialization, embedding generation, and querying.

➔ data/ – Contains CSV files with legal documents and extracted case metadata.

- **Before proceeding, we checked if any containers were running using sudo docker ps -a**

- **If containers were running but needed to be reset, we stopped them by sudo docker compose down**

- **Then, to start Milvus and dependencies cd /home/ubuntu/open-webui-0.4.8/vector_db/embeddings/code and then run sudo docker compose up -d.**

- **Once started, we ensured the containers were healthy: sudo docker logs milvus-standalone, sudo docker logs milvus-etcd and sudo docker logs milvus-minio. If all logs confirmed successful initialization, we proceeded.**

- **To check if collections existed in the Milvus vector database I ran python3 test.py If collections needed to be dropped we drop them by uncommenting the part which is there in test.py Once validated, we proceed with embedding generation.**

- **To generate vector embeddings and store them in Milvus, I ran python3 Central_acts_DB.py Once completed python3 SC_judgments_DB.py**

  (Note: These scripts were executed sequentially to prevent excessive RAM usage. Please be patient, as the process may take some time.)

- **Used mixedbread-ai/mxbai-embed-large-v1 embedding model consistently across both scripts. Ensured the correct embedding dimension (1024) to prevent errors. Converted embeddings to lists before inserting into Milvus (vector.tolist()).**

- **Once embeddings were generated, we validated their presence using python3 test.py If collections were present and contained vectors, we proceeded to querying.**

- **To test retrieval, we ran python3 query.py if results returned successfully, our setup was validated.**

3) **Conclusion:**

- **The Legal Document Retrieval System was successfully set up, optimized, and integrated into the Open-WebUI project. We processed legal statutes and Supreme Court judgments into vector embeddings using Milvus as the vector database and mixedbread-ai/mxbai-embed-large-v1 as the embedding model. The entire pipeline from data preprocessing, embedding generation, and storage in Milvus to querying via FastAPI and frontend integration was structured to ensure efficient retrieval.**

- **Key challenges, such as embedding dimension mismatches, GPU-related adjustments, and metadata parsing issues, were identified and resolved. The backend API seamlessly retrieves and returns relevant documents, displaying case titles, judgment dates, and legal statute details. Currently, LLM summarization is not enabled, but the retrieval system functions independently with Milvus-based searches. The implementation is now fully functional, allowing accurate legal information retrieval based on user queries.**

- **Ibrahim Sultan**