AI-Powered Jsearch Hybrid

Objective: To deliver a dynamic, context-aware search experience for product Discoverability

Core Technologies:

- **LangChain**: Manages the retrieval and generation workflow.
- **Embedding Model:** sentence-transformers/all-MiniLM-L6-v2
- **Cross Encoder** : cross-encoder/ms-marco-MiniLM-L-12-v2
- **LLM** : Ollama 3.1 8B
- **ElasticSearch** : Lexical search using BM25 Encoder under the hood
- **FAISS**: Provides fast similarity-based search through embeddings.
- **MLflow**: Tracks experiments, model performance, and versioning.
- **FastAPI**: Used to develop the API for handling search queries.

Approach: The system leverages **Retrieval-Augmented Generation (RAG)**, which combines **hybrid retrieval from a knowledge base (FAISS) and ELasticSearch using Ensemble Mechanism** with **language model generation (LLM)** for contextual retrieval and Responses .

PRODUCT SEARCH FLOW HYBRID

This prototype implements a two-stage **hybrid search pipeline**, combining **semantic similarity** (via dense vectors) and **lexical matching** (via BM25) with LLM-powered result summarization to deliver intelligent product discovery.

Module 1: Data Ingestion & Indexing

•Processes product data and metadata.

•Builds:

- **FAISS index** for semantic search.
- **Elasticsearch (BM25)** index for keyword search.

Module 2: Interactive Query Pipeline

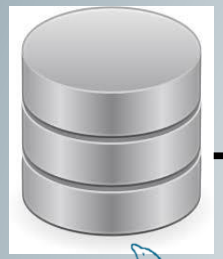
•**Hybrid Retrieval**: Combines FAISS + BM25 results via EnsembleRetriever.

•**Re-ranking**: Refines top-k results using a Cross-Encoder model.

•**LLM Summarization**: Top 25 re-ranked products summarized by **ChatOllama** into a natural-language response.

Enterprise Knowledge Base

1



Local Store

Preprocessed Data

Docs

mlflow *Model Registry*

Fine-Tuned Hugging Face Embedding model

Embedded Docs

Knowledge Base

Elastic Client

FAISS META

Langchain

Prompt-tuned LLm

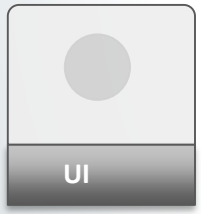
Prompt+ Query Enhanceed context

TOP K Products

Product Summary and info

User Query & Search generation

2



User query

Enterprise App

Module 1 – Data Ingestion & Indexing

Goal

Prepare product data for hybrid search using both **semantic** and **lexical** indexing.

Steps Involved

1. **Data Loading**
   - Read product catalog from CSV.
   - Clean, normalize, and enrich with metadata (e.g., name, supplier, location, category).
2. **Document Creation**
   - Create descriptive product chunks using LangChain Document.
   - Add structured metadata for downstream filtering and explanation.
3. **Dense Indexing (Semantic Search)**
   - Generate embeddings using SentenceTransformer.
   - Store vectors in **FAISS** for fast semantic similarity search.
4. **Sparse Indexing (Lexical Search)**
   - Filter and format product descriptions.
   - Index documents into **Elasticsearch (BM25)** for keyword-based retrieval.

Output

Two parallel search backends:

- **FAISS Vector Store**
- **BM25 Elasticsearch Index**

Module 2 – Interactive Query Pipeline

Goal

Enable users to perform hybrid product search with ranked results and LLM-generated summaries

Pipeline Flow

1.Hybrid Retrieval

- Query passed to both **FAISS** and **BM25**.
- Results combined using EnsembleRetriever (weighted scoring).

2.Re-ranking

- Cross-Encoder scores relevance between query and each result.
- Top-k results are reordered for better accuracy.

3.Metadata Compilation

- Metadata from top results compiled into a structured format (ensemble_metadata).

4.LLM Summarization

- Top 25 results sent to **ChatOllama** with a custom prompt.
- LLM generates a concise, human-readable product summary in llm_text.

Output

A ranked list of products + a **natural-language summary** tailored to the user's query.

Fine-Tuning Sentence Bert

What is Fine-Tuning?

Fine-tuning is the process of training a **pre-trained deep learning model** on a **custom dataset**. In our case, we are fine-tuning a **Sentence-BERT (SBERT)** model to improve its performance in **semantic similarity and text retrieval tasks**. The model is trained using **contrastive learning**, which improves its ability to differentiate between **similar and dissimilar** text pairs.

The training process involves:

- **Generating positive and negative pairs** from a dataset.
- **Training the model** with CoSENT Loss to enhance embedding quality.
- **Logging and tracking experiments** using MLflow.
- **Registering the trained model** for future deployment.

Fine-Tuning Sentence Bert

What is Contrastive Learning?

Contrastive learning is a technique that improves embeddings by:

- **Maximizing similarity** between semantically related text pairs (**positive pairs**).
- **Minimizing similarity** between unrelated text pairs (**negative pairs**).

It is widely used in **metric learning, sentence embeddings, and information retrieval**.

How It Works

- Instead of optimizing absolute distances, **it optimizes ranking**.
- **Always ensures** that positive pairs have **higher similarity** than negative pairs.
- Uses **log-sigmoid ranking loss**, avoiding the **margin tuning issue** of Contrastive Loss.

Fine-Tuning Sentence Bert

CoSENT Loss

CoSENT (**Contrastive Semantic Embedding Loss**) is an improved loss function designed for **Sentence-BERT** models. Unlike traditional contrastive loss, **CoSENT ensures proper ranking between positive and negative pairs within a batch**.

How CoSENT Loss Works

1. **Compute cosine similarity** between embeddings.
2. **Sort pairs dynamically**, ensuring positive pairs have higher similarity scores than negative pairs.
3. **Apply log-sigmoid loss** to enforce ranking constraints.

Sentence Pair	Similarity (After Training)
(Engine Oil, Car Lubricant)	0.96 (Highly similar)
(Laptop, Computer)	0.89 (Still highly ranked)
(Engine Oil, Olive Oil)	0.05 (Lower similarity, enforced dynamically)
(Laptop, Banana)	0.02 (Enforced lower than negative threshold)

Advantages of CoSENT Loss

- **No need for margin tuning** (Unlike Contrastive/Triplet loss).
- **Efficient batch-based training** (computes loss across multiple pairs).
- **Works well with real-world noisy data**.
- **Scales better to large datasets** compared to traditional contrastive loss.

Fine-Tuning Sentence Bert

Step	Description
Data Preparation	Convert product descriptions into a structured text format.
Positive Pair Generation	Group similar products to form training examples for similarity learning.
Negative Pair Generation	Select unrelated products to form training examples for dissimilarity learning.
Fine-Tuning	Train the SentenceTransformer model using CoSENT Loss to adjust embedding distances.
Experiment Tracking	Log model performance, loss values, and parameters using MLflow.
Model Registration	Store the fine-tuned model in the MLflow Model Registry/Local store for deployment.

Conclusion and Next Steps

Innovative Approach: Our product search pipeline seamlessly integrates FAISS & ElasticSearch Client for rapid retrieval with LLM-based summarization, enabling efficient and dynamic product discovery.

Commitment to Feedback: Ongoing user feedback will drive our iterative enhancements, ensuring the product evolves to meet market needs effectively.

Next Steps:
Data Collection: Gather the necessary data for the Pilot Product.
MVP Development: Create the Minimum Viable Product (MVP) and launch the pilot.