SHL Assessment Recommender – System Overview

Data Collection & Representation

- **Crawling**:
  I scraped SHL's product catalog using **BeautifulSoup** to extract assessment metadata such as title, description, tags, job level, remote/adaptive support, and duration.
  - I also crawled individual assessment detail pages to capture richer content like **job levels** and **test traits**.

- **Representation**
  - After crawling, I enriched each assessment with **semantic tags** using the **Gemini LLM**. Tags include hard skills (e.g., Python), soft skills (e.g., communication), and test traits (e.g., scenario-based).
- **Storage**:
  - All enriched assessments were stored in a CSV, and vector embeddings (using **all-MiniLM-L6-v2**) were stored in **Pinecone**, with full metadata attached for reranking.

Retrieval & Reranking Pipeline

- **Query Input**:
  - Users can enter a **natural language query**, paste a **job description**, or provide a **JD URL**.
  - A preprocessing module intelligently classifies input and uses **Gemini** to extract the core search intent from JDs.
- **Vector Retrieval**:
  - Top 50-60 matches are fetched using **semantic vector similarity** from Pinecone.
- **LLM-based Reranking**:
  - A custom prompt is sent to **Gemini** to rerank the top candidates based on semantic alignment with:
    - Technical + soft skills
    - Duration constraints
    - Job levels and **test types (mapped)**

Evaluation & Tracing

To measure the effectiveness of our suggestion system, we developed a systematic evaluation process and incrementally iterated through several retrieval methodologies to enhance relevance and accuracy.

Evaluation Metrics Used:

- **Recall@10**: Fraction of relevant assessments retrieved in the top 8/10.
- **MAP@10 (Mean Average Precision)**: Measures both precision and position of relevant results.

Retrieval Techniques Tried

1. Semantic-only Retrieval and then Reranking (Final Approach and BEST Approach)

- Direct vector search via pincone's semantic search function..
- Gemini reranking used to interpret constraints (e.g., time limit, skills, job level).

Best observed performance:

Query: "I am hiring for Java developers who can also collaborate effectively with my business teams. Looking for an assessment(s) that can be completed in 40 minutes."

Recall@10: 0.5000 | MAP@10: 0.4375

2. Hybrid Retrieval (Indexing Retrieval using BM25 + Vector Fusion)

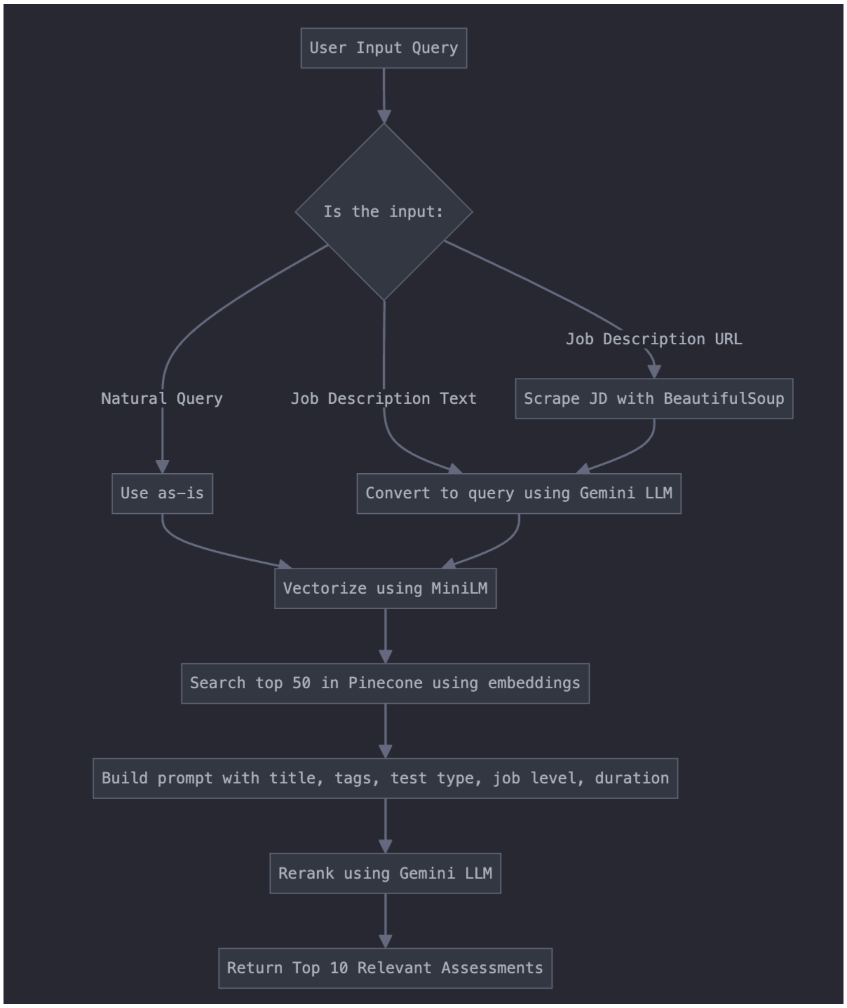
- Combined sparse keyword search (BM25) with dense vector retrieval.
- Fusion reranking weighted scores from both methods.
- **Outcome**: Moderate improvement in some queries but inconsistent performance due to noisy tokenization on short descriptions.

3. Cross-Encoder Reranking with Pinecone Inference

- Used `bge-reranker-v2-m3` to rerank top 60 vector candidates before Gemini reranking.
- Used this approach to reduce the context window for gemini prompt , but unfortunately this approach failed.
- **Observation**: reranked outputs did not consistently improve final Recall/MAP scores.

Do see all my trials in notebooks/trial.ipynb folder

FLOW DIAGRAM:



Testing

1. **API Endpoint**

- **Method:** `POST`
- **URL:** `/recommend`
- **Input:** JSON payload containing a `query` or job description

Steps to run:

1.Run this in terminal: uvicorn api:app --reload

2.Then Go to this api url: http://127.0.0.1:8000/docs#/default/recommend_assessments_recommend_post

(A user-friendly UI for interacting with your API)

2. **Gradio Frontend(HuggingFace Spaces)**
    - URL: https://huggingface.co/spaces/hshivhare/SHL-AI