SHL Assessment Recommendation System

[-By KHUSHI SHUKLA]

Approach and Optimization Document

---

1. Objective

The goal of this project was to design a recommendation system that maps job descriptions or user queries to the most relevant **SHL assessment products**. The system needed to support **an API**, a **Streamlit web interface**, and produce a **CSV output** of predictions on the provided test dataset.

---

2. Overall Approach

The solution integrates **semantic embedding models** to capture the meaning of both job descriptions and assessment content.
Key steps followed:

Step 1 – Data Preparation

- The file **products.csv** was created from the source Excel file **Gen_AI Dataset.xlsx**. The original dataset contained detailed information about SHL products, including names, descriptions, and categories. To prepare it for the recommendation model, only the **text-based columns** — mainly *Product Name* and *Description* — were extracted. The text was cleaned by removing missing values, extra spaces, and special symbols.
- The provided product catalog (products.csv) was preprocessed by combining fields like *AssessmentName*, *Description*, and *Category* into a unified text representation.

Step 2 – Model Selection

- The primary model used was **SentenceTransformer – all-MiniLM-L6-v2**, a compact yet high-performing model optimized for semantic similarity.
- Additionally, a **Hugging Face Transformer model** was integrated to experiment with custom embeddings and compare against SentenceTransformer results.
- Gemini LLM was initially considered for cloud-based embedding generation, but due to quota limits, the final version focused on **offline models**, ensuring reproducibility and cost-free usage.

Step 3 – Embedding Generation

- Both the catalog entries and user/job queries were converted into **vector embeddings** using the transformer model.

- **Cosine similarity** was used to measure semantic closeness between a query and each assessment.

Step 4 – Recommendation Logic

- For each query, similarity scores were sorted in descending order.
- The top-N recommendations were returned (N configurable in Streamlit).
- The system output included the **most relevant assessment name**, its **similarity score**, and a **ranked list** of top matches.

---

3. System Components

Component	Description
1. Streamlit App	User interface to input job descriptions, select model type, and view recommendations interactively.
2. FastAPI Backend	Serves as an API endpoint for programmatic access to the recommender, returning JSON responses.
3. Prediction Script	Generates predictions for all queries in the given test set and saves them as predictions.csv.

---

4. Optimization Strategy

During development, several experiments were conducted to improve the **relevance and stability** of recommendations:

Stage	Optimization	Outcome
Baseline	Used default SentenceTransformer embeddings with raw text.	Moderate accuracy, noisy recommendations.
Text Enrichment	Concatenated multiple catalog columns into combined_text.	Improved context representation.
Normalization	Normalized embeddings and removed stopwords.	Enhanced semantic distance accuracy.
Model Fine-Tuning	Adjusted sentence length truncation and batch size.	Faster inference with minimal accuracy drop.

Stage	Optimization	Outcome
Top-K Tuning	Experimented with top-3, top-5, and top-10 results.	Top-10 proved most consistent for HR domain relevance.

The final model achieved a **balanced trade-off** between speed, interpretability, and recommendation precision.

---

6. Key Takeaways

- **Offline embeddings** via SentenceTransformer are fast and reliable for semantic similarity tasks.
- **Hugging Face models** offer flexibility to extend to domain-specific fine-tuning if more labeled data becomes available.
- **Configurable recommendations (Top-K)** in Streamlit improve usability for HR professionals exploring multiple assessments.
- The pipeline is **modular**, enabling future integration with APIs, caching, or database indexing for large catalogs.

---

7. Final Deliverables

Deliverable	Description
API Endpoint	FastAPI service returning top-N recommendations in JSON format.
Web Interface	Streamlit app for interactive exploration.
CSV File	predictions.csv – Predictions for test queries with ranked top-10 results.
GitHub Repository	Complete implementation with code, model references, and documentation.

---

8. Future Improvements

- Fine-tune the transformer model using labeled SHL historical recommendation data.
- Implement **vector database indexing** (FAISS / Pinecone) for faster retrieval on larger datasets.
- Integrate a **feedback loop** to continuously improve relevance using user click data.

✅ Conclusion

This project demonstrates a robust and interpretable **semantic recommendation system** tailored for SHL assessments.
Through model selection, embedding optimization, and modular deployment (Streamlit + FastAPI), the system achieves a high degree of accuracy, scalability, and end-user usability.