SHL Recommendation System - Technical Report

1. Approach & Architecture

Data Collection

To ensure comprehensive coverage, we implemented a custom web scraper targeting the SHL Product Catalog. The scraper iterates through paginated listings (`?start=X`) to capture all "Individual Test Solutions", collecting: - Assessment Name - Direct URL - Description (initially from metadata, enriched with page content) - Inferred Type (Technical, Behavioral, Cognitive, etc.)

Challenges: The site uses strict anti-bot measures and server-side timeouts. **Solution**: We implemented robust retry logic, session management, and polite delays.

Embedding Strategy

We utilized the `sentence-transformers/all-MiniLM-L6-v2` model for generating vector embeddings. This model was chosen for its optimal balance between speed and semantic accuracy. - **Preprocessing**: We combined the Assessment Name, Description, and our inferred "Type" into a single text block. This ensures that the vector representation captures both the semantic meaning of the description and the categorical nature of the test. - **Indexing**: Embeddings are normalized and stored in a **FAISS** (Facebook AI Similarity Search) index using Inner Product (equivalent to Cosine Similarity for normalized vectors) for sub-millisecond retrieval.

Recommendation Engine (2-Stage RAG)

We implemented a state-of-the-art **Retrieve & Re-rank** pipeline: 1. **Retrieval (High Recall)**: The user's query is encoded into a vector and matched against the FAISS index to retrieve the top 50 candidates. This casts a wide net to capture all potentially relevant assessments. 2. **Re-ranking (High Precision)**: We employ a **Cross-Encoder** (`cross-encoder/ms-marco-MiniLM-L-6-v2`) to independently score each of the 50 candidate pairs (Query, Document). This model is far more accurate than simple cosine similarity as it captures deep semantic interactions between the query and the assessment text. 3. **Result**: The re-ranked top 10 results are presented to the user, ensuring that the most semantically precise matches appear first. 4. **Intent Balancing**: A final heuristic layer ensures diversity (Technical vs Behavioral) if the Cross-Encoder scores are close or if the query implies a mixed role (e.g., "Engineering Manager").

---

2. Evaluation & Tuning

We used the provided `Train-Set` (10 labeled queries) to tune the system. The primary metric was **Recall@10** (Checking if the ground truth `Assessment_url` appears in the top 10 recommendations).

Performance Logic

- **Baseline**: Pure semantic search often missed specific "Test" nuances, favoring generic descriptions.
- **Improvement**: Adding the "Type" field to the embedding corpus significantly improved the separation between Cognitive and Technical tests.
- **Result**: The system achieves a high precision on the training set, successfully retrieving the correct URLs for queries like "Java Developer" and "Sales Manager".

Balancing Results

We observed that pure similarity search would sometimes flood the results with 10 variations of "Java Test". The implemented balancing logic successfully diversifies the output to include related soft-skill assessments when the query implies it, aligning with the "Individual Test Solutions" philosophy.

---

3. Future Improvements

- **Dynamic Scraping**: Integrate a headless browser (like Playwright) for more resilient data collection if the static site structure changes.
- **LLM Reranking**: Use a lightweight LLM to re-rank the top 20 results by relevance, explaining *why* a specific test fits the generic JD.
- **Feedback Loop**: Implement User Feedback storage to fine-tune the embeddings over time.

---

4. Conclusion

The developed system successfully meets the requirement of providing relevant, diverse, and accurate assessment recommendations (~380+ items indexed). The architecture is modular, allowing for easy updates to the model or data source.