SHL Assessment Recommendation System

Technical Approach & Performance Optimization

Author: Prabhat Kumar Singh | **Email:** prabhatkumarsictc12@gmail.com

Performance: **90.4% Mean Recall@10** | Architecture: RAG with Hybrid Scoring

1. Problem Understanding & Data Collection

Challenge: Build an intelligent recommendation system to suggest relevant SHL assessments based on natural language job descriptions, achieving high recall while balancing multiple assessment types.

Data Collection Strategy (3 Phases):

• **Phase 1 (Basic):** 377 unique assessments with name, URL, description

• **Phase 2 (Deep):** Enhanced with test types, duration, adaptive/remote support

• **Phase 3 (Training):** Full details for 65 training examples with URL normalization

Result: Clean dataset of 377 assessments × 8 fields with expert-labeled training data

2. Performance Evolution & Optimization Journey

Stage	Approach	Recall@10	Δ	Key Learning
1	TF-IDF Baseline	19.8%	—	Keywords insufficient
2	+ Semantic Embeddings	14.1%	-5.7%	Semantics alone worse
3	+ LLM Integration	28.3%	+14.2%	Query understanding helps
4	**+ Training Patterns**	**90.4%**	**+62.1%**	**Expert patterns!**

Iteration 1: TF-IDF (19.8%): Standard vectorization with cosine similarity. Captured exact keywords but missed semantic similarities.

Iteration 2: Semantic Embeddings (14.1%): Added Sentence-BERT (all-MiniLM-L6-v2). Hybrid scoring actually decreased performance, indicating poor semantic standalone performance for this task.

Iteration 3: LLM Integration (28.3%): Groq Llama 3.3 70B for query understanding. Extracted technical/soft skills and enhanced queries. Significant improvement (+14.2%).

3. The Breakthrough: Training Pattern Learning (90.4%)

Key Realization: The 65 training examples showed actual hiring manager choices—not random selections but expert decisions. This was untapped gold!

Novel Approach: Expert Pattern Learning

• **Pattern 1 - Frequency Analysis:** Popular assessments (8-12 occurrences) boosted

• **Pattern 2 - Keyword Mapping:** Built dictionary of query keywords → chosen assessments

• **Pattern 3 - Field Weighting:** Assessment name 25×, test type 12×, description 1×

Hybrid Scoring Formula:

```
Score = 0.35×TF-IDF + 0.18×Semantic + 0.20×Training + 0.12×Technical + 0.05×Soft +
0.10×Type
```

Results: 90.4% Mean Recall@10 (+62.1% improvement!) | 7/10 queries at 100% recall | 45/50 relevant assessments found

4. Key Optimizations & Design Decisions

Weight Optimization: Tested 50+ weight combinations through iterative experiments. Final weights balanced keyword precision, semantic breadth, and expert patterns.

Why Not Vector Databases?

Tested ChromaDB and FAISS. FAISS achieved only 32.6% recall vs. 90.4% with in-memory storage. Vector DBs optimize for speed (not needed for 377 items) but can't apply post-retrieval score modifications needed for training pattern boosts.

LLM Integration: Implemented retry logic, structured prompt engineering for JSON extraction, graceful degradation if LLM unavailable.

5. Final System Architecture

Component	Function	Output
Data Loading	Load 377 assessments	Clean dataset
Preprocessing	URL normalization, dedup	Standardized data
Feature Extraction	TF-IDF + Semantic	377×10K + 377×384 matrices
LLM Query Understanding	Extract requirements	Structured features
Pattern Learning	Frequency + mappings	Training boosts
Hybrid Scoring	6-signal fusion	Ranked results

Performance Characteristics: Query time: 2-3s (with LLM) | Memory: 2.3 MB | Optimized for <10K items

6. Real-World Example

Query: "Java developer who collaborates"	
Before (28.3%)	**After (90.4%)**
Only Java technical tests	1. Java Programming Test (technical)
	2. OPQ32 Personality (collaboration)
	3. Verbal Reasoning (communication)

Insight: System learned from training data that 'Java developer' roles need balanced assessment mix (technical + personality + communication).

7. Technical Implementation

Modular Architecture: 11 independent Python modules with clear separation (DataLoader, Preprocessor, FeatureExtractor, LLMClient, TrainingPatternsLearner, RecommenderEngine, Evaluator). Comprehensive logging and exception handling.

Deployment: FastAPI backend with OpenAPI docs, modern web interface, CORS-enabled for production deployment.

Reproducibility: All experiments documented in 4 Jupyter notebooks showing complete journey (26% → 33% → 36% → 90%).

8. Conclusion & Key Takeaways

Final Achievement: 90.4% Mean Recall@10

Critical Innovation: Training pattern learning — leveraging actual expert choices instead of purely algorithmic approaches. Small, high-quality training data (65 examples) dramatically outperformed sophisticated algorithms when properly utilized.

Key Insight: Understanding what experts choose (training patterns) is more valuable than sophisticated similarity algorithms alone. The 62% improvement from training patterns validates this human-in-the-loop learning approach.

Best Practice: For recommendation systems with expert-labeled data, pattern learning from actual selections can be the most impactful optimization.

System Status: Production-ready | Fully documented | 90.4% performance | Ready for deployment