

Building a Production-Ready End-to-End Recommender System AI Research

This document outlines the experimental and evaluation aspects of solving the **information retrieval (IR) problem** for recommending assessments. The entire system architecture, including detailed design approaches, deployment considerations, and scalability, is discussed in the repository's README file (https://github.com/AGAMPANDEYY/llm_recommendation_engine?tab=readme-ov-file).

This summary focuses on key experiments, design choices, and measured impacts to demonstrate research-oriented AI engineering and system design.

1. Problem Overview and Design Philosophy

The task involves building a recommender stack for assessments, addressing challenges in catalog ingestion, query understanding, hybrid retrieval, reranking, and post-processing. Prioritizing **recall@10** and **MRR@10** on a small dataset (train/val: 52/13 queries), the system uses a modular pipeline:

scrape → enrich → embed → index → rewrite → retrieve → fuse → rerank → constrain.

Experiments emphasize **CPU-friendly** components for production reproducibility on resource-constrained environments (e.g., HF Spaces), avoiding external dependencies. Metrics are logged in runs/*/metrics.json for traceability.

- Hybrid signals (lexical + semantic) for robust IR.
- Adaptive query handling via deterministic/LLM rewrites.
- Finetuning for precision gains without GPU reliance.
- Ablations on candidate depth, fusion weights, and components to optimize quality-latency trade-off.

2. Data and Catalog Build

- **Source & Enrichment:** Crawled **389** unique assessments (up from 377 via parser fixes in `crawler/parser_detail.py`). Enriched with roles, languages, remote/adaptive flags, durations, and test types. Used `catalog_docs_rich.jsonl` for concatenated fields `(name + description + doc_text + metadata)` to boost recall on long queries.
- **Design Choices:** Retained rich fields for multi-signal retrieval and filters. Relaxed parsing rules to ingest variant URLs pages without strict metadata, improving coverage.
- **Artifacts:** **FAISS** index (`data/faiss_index/index_bge.faiss`), embedding ID map (`data/embeddings_bge/assessment_ids.json`), role vocab(`data/catalog_role_vocab.json`).
- **Impact:** Rich catalog improved **semantic/lexical recall**; experiments showed baseline (`catalog_docs.jsonl`) underperformed on metadata-heavy queries.

3. Indexing and Embeddings

- **Embeddings:** BAAI/bge-small-en-v1.5 for CPU efficiency and strong semantic quality.
- **Indexing:** Flat FAISS (not HNSW) prioritized for exact recall on small corpus (~389 items); latency acceptable.
- **Experiments:** Tested flat vs. HNSW (`data/faiss_index/index_hnsw.faiss`) – flat chosen for recall priority. Rich embeddings (`index_rich.faiss`) enhanced long-tail coverage.
- **Rationale:** Avoided external DBs (e.g., Pinecone) for offline reproducibility and cost.

4. Retrieval and Fusion

- **Components:** BM25 (lexical, in-repo impl) + dense (BGE + FAISS cosine) fused via **weighted RRF** (k=60, topn=200 candidates).
- **Experiments and Baselines:**
 - BM25-only: recall@10 = 0.0769 (val).
 - Vector-only: recall@10 = 0.1538.
 - Hybrid RRF (no rerank): recall@10 = 0.2308.
 - Candidate ablations: Recall plateaued at 200; lower hurt recall, higher increased latency marginally.

- Adaptive weights: Biased toward BM25 for skill-heavy queries, vector for verbose ones; outperformed equal weights.
- Hybrid fusion improved exact-term and semantic matching, with RRF enabling query-adaptive biasing.

5. Reranking

- **Model:** Finetuned cross-encoder/ms-marco-MiniLM-L-6-v2 (models/reranker_crossenc/v0.1.0) on pairwise data (binary CE loss, 1 epoch, lr=1e-5).
- **Process:** Rescores top-200 fused candidates to top-10.
- **Experiments:** Off-the-shelf MiniLM vs. finetuned: *Finetuning lifted recall@10 from 0.23 (no rerank) to 0.38–0.46; MRR@10 to ~0.19.*
- **Rationale:** Compact model for CPU latency; pairwise training stabilized on small data.

6. Query Rewriting and Planning

- **Deterministic:** Heuristics for intent (TECH/BEHAVIORAL), skills (must/soft/negated via vocab), roles, constraints.
- **LLM Variants:** Qwen2.5-1.5B-Instruct (best quality, JSON schema); fallbacks: NuExtract, FLAN-T5-small.
- **Experiments:** LLM rewrite + hybrid + rerank achieved best *recall@10=0.4615*. Qwen > FLAN (underperformed on structure); deterministic as safe floor.
- **Impact:** Injected structure improved retrieval on ambiguous queries; fallback ensured robustness.

7. Constraints and Post-Processing

- **Filters:** Duration (<= target), remote/adaptive (inferred via string matches if missing).
- **Guarantees:** At least one result; sanitized numerics/debug payloads.
- **Code:** tools/constraints_tool.py for modular application.

8. Evaluation Summary

Run ID	Pipeline	Val recall@10	Val recall@5	Val MRR@10
20251217_081916_hybrid_rf_rerank_rewrite	Hybrid RRF + rerank + rewrite	0.4615	0.1538	0.1250
20251217_075251_hybrid_rf_rerank_rewrite	Hybrid RRF + rerank + rewrite	0.3846	0.2308	0.1865
20251217_000844_hybrid_rf_rerank_bge	Hybrid RRF + rerank (BGE)	0.3846	0.2308	0.1865
20251216_152326_hybrid_rf_rerank	Hybrid RRF + rerank	0.3846	0.2308	0.1865
20251216_151000_hybrid_rf	Hybrid RRF (no rerank)	0.2308	0.0769	0.0000
20251216_144825_vector	Vector only	0.1538	0.0769	0.0000
20251216_140500_bm25	BM25 only	0.0769	0.0000	0.0000

- **Insights:** Stacked improvements (rewrite + hybrid + rerank) drove monotonic gains. Best config: rewrite for structure, RRF for fusion, rerank for precision.

10. Future Improvements

- Agentic extensions: ReAct-style planner for iterative refinement.
- KG augmentation: Role-skill graph for expansion/fusion.
- Learning: DPO/RLHF on feedback; company adapters (LoRA).
- Adaptation: Real-time vocab/weight updates; nDCG/latency profiling.