Reproducing “Experimental Analysis of Dense vs. Sparse Retrieval”

Gabriele Righi

University of Pisa

February 11, 2026

Introduction & Research Goals

Context: The "Operational" Gap

- Modern IR has shifted from Sparse (BM25) to Dense (Embeddings).
- Dense retrieval is effective but resource-intensive.

The 3 Research Questions (Lin et al., 2024)

This project reproduces the analysis to answer three key operational questions:

1. **RQ1:** Is the complexity of HNSW always necessary, or is **Flat index** sufficient for typical workloads?
2. **RQ2:** Can we use **INT8 Quantization** to save RAM without hurting retrieval quality (nDCG)?
3. **RQ3:** How does **Sparse Retrieval** (BM25/SPLADE) compare to Dense methods in a modern GPU environment?

Experimental Setup

- **Benchmark**: BEIR (Heterogeneous Evaluation).
- **Datasets Analyzed**: `scifact`, `nfcorpus`, `cqadupstack` (Android, Gaming, GIS).
- **Hardware**: NVIDIA T4 GPU (16GB VRAM).
  - *Constraints simulated a realistic production environment.*

Metrics

nDCG@10 Retrieval Quality (Ranking accuracy).

Recall@10 Retrieval Coverage (Relevant items found).

QPS Queries Per Second (Throughput/Speed).

Challenge I: The Missing Link in Pyserini

Original Plan: Use `Pyserini` (Lucene wrapper) for *both* Sparse (BM25) and Dense (HNSW) indexing, matching the paper exactly.

The Obstacle:

- Pyserini supports HNSW via Java, but the **Python API for adding vectors to HNSW indexes is limited/missing**.
- Impossible to feed custom BGE embeddings directly into Pyserini's HNSW from Python easily.

The Engineering Pivot: Hybrid Pipeline

- **Sparse Path:** Retained Pyserini for BM25 (Industry Standard).
- **Dense Path:** Migrated to **Faiss** (Facebook AI Similarity Search).
- *Why?* Faiss offers native **GPU support** and flexible python bindings for custom embeddings, enabling a fairer comparison for RQ1 (Flat vs HNSW).

Challenge II: The SPLADE Underperformance Mystery

The Anomaly While BM25 and Dense Retrieval matched the paper's baselines immediately, the standard SPLADE implementation (via Pyserini) failed.

Method	Implementation	nDCG@10	Status
BM25	Pyserini (Lucene)	0.323	Success
BGE Dense	Faiss	0.370	Success
SPLADE	**Pyserini Impact**	**0.230**	**Fail (-28%)**

Root Cause Analysis

- *Hypothesis A (Wrong Model)*: Switched to `selfdistil`. No change.
- *Hypothesis B (Quantization)*: Forced integer scaling ($w \times 100$). No change.
- **Conclusion**: Pyserini's "Black Box" Impact Indexing was introducing **lossy compression artifacts**, destroying the precision of sparse weights.

The Solution: Custom Matrix Engine

The Fix: Bypassed Pyserini. Built a **pure Python sparse engine** using `SciPy`.

Methodology

- **Direct Encoding**: Used HuggingFace to avoid quantization artifacts.
- **Vectorized Search**: Replaced index lookup with Matrix Multiplication.

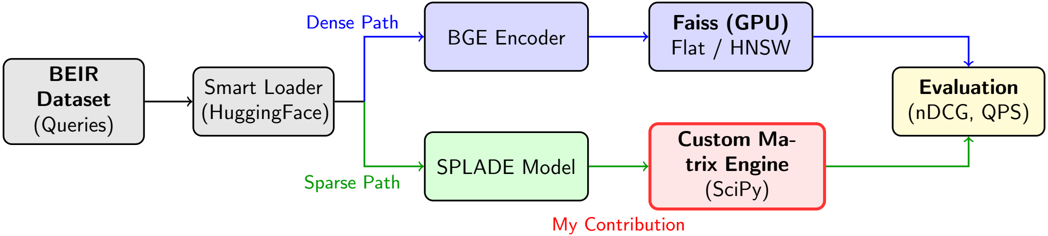
Core Logic

$$S = \mathrm{D} \times \mathrm{q}^{T}$$

Impact

- **Quality**: nDCG@10 restored to 0.357 (vs 0.230).
- **Speed**: ~296 QPS (Efficient on Python).

Implementation Pipeline: The "Dual-Engine" Architecture



Architectural Highlights:

- **Parallel Processing:** Simultaneous execution of Dense and Sparse pipelines.
- **Custom Matrix Engine**: Replaced Pyserini's "Black Box" indexer with a transparent vector-matrix multiplication engine for SPLADE.
- **GPU Acceleration**: Integrated Faiss to unlock GPU speeds for Dense Retrieval.

Operational Advice: RQ1 & RQ2

Summary: When to switch Index? When to Quantize?

Dataset	Method	Build	RAM	QPS	nDCG	Verdict / Trade-off
SciFact	Flat (FP32)	<**0.1s**	.01 GB	**4,205**	**0.738**	**Best Choice**
(5k)	Flat (INT8)	<0.1s	.01 GB	2,209	0.736	*Slower (CPU limit)*
Quora	Flat (FP32)	**1s**	.38 GB	120	0.889	*Too Slow*
(523k)	HNSW (FP32)	4m	.45 GB	**361**	0.889	**Worth the wait**
NQ	HNSW (FP32)	19m	2.4 GB	9,034	0.464	*High RAM*
(2.6M)	HNSW (**INT8**)	25m	**0.6 GB**	**9,100**	0.468	**-75% RAM (Safe)**

Final Recommendations

- **RQ1 (Scale):** Use **Flat** for small data (<100k). Use **HNSW** for large data (>200k) as QPS scales better, with negligible nDCG drop (-0.004).
- **RQ2 (Quantization):** Always use **INT8** for HNSW. It saves 75% RAM while maintaining identical nDCG quality (0.541 vs 0.540).

Quality Analysis: Sparse vs. Learned Sparse vs. Dense

Q: Does SPLADE bridge the gap? Is Dense still superior?

Dataset	BM25	SPLADE	Dense	Winner
SciFact *(Small)*	0.679	0.717	**0.738** *(Flat FP32)*	**Dense**
Quora *(Medium)*	0.789	0.842	**0.889** *(Flat FP32)*	**Dense**
NQ *(Large)*	0.235	**0.577***	**0.541** *(Flat FP32)*	**Dense**

- **Small/Medium Data:** SPLADE improves over BM25 significantly (+15-20%).
- **NQ Result:** Dense is the winner, but required ∼**4h GPU encoding** (vs BM25 minutes).
- ***Note on SPLADE:** Due to computational constraints, SPLADE was evaluated on a **100k subset**.
- **Conclusion:** Dense Retrieval provides the best quality, justifying the high setup cost.

Conclusions & Operational Advice

Answering the Research Questions:

RQ1 **HNSW vs Flat:** *Verdict:* HNSW is **not** always necessary. **Flat Index** is faster for datasets < 100k docs and simpler to maintain.

RQ2 **Quantization Safety:** *Verdict:* **Yes.** INT8 reduces RAM by 75% with negligible quality loss ($< 1\%$ nDCG drop). It should be the default.

RQ3 **Sparse vs Dense:** *Verdict:* Dense methods (BGE) generally outperform Sparse (BM25) in quality, but custom Sparse implementations (Matrix Engine) can be extremely fast on GPU.

Final Recommendation

For most production scenarios under 1M docs: **Use Dense Retrieval with Flat Index + INT8 Quantization.**

References I

- Formal, Thibault, Benjamin Piwowarski, and Stéphane Clinchant (2021). "SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking". In: *SIGIR '21*.
- Lin, Sheng-Chieh, Shiguang Yang, and Jimmy Lin (2024). *Operational Advice for Dense and Sparse Retrievers: HNSW, Flat, or Inverted Indexes?* arXiv: 2409.06464 [cs.IR]. URL: https://arxiv.org/abs/2409.06464.
- Righi, Gabriele (2026). *Dense vs Sparse Retrieval - Replication Repository*. GitHub Repository. URL: https://github.com/Gheb6/dense-vs-sparse-retrieval.
- Robertson, Stephen and Hugo Zaragoza (2009). "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends® in Information Retrieval* 3.4, pp. 333–389. DOI: 10.1561/1500000019.

Thank you!

Questions?