A Study of Embeddings, LLMs, and RAG Methods

Information Retrieval – Project Milestone 1

Patrascu Adrian Modiga Miriam Toderian Vitalii

Faculty of Automatic Control and Computer Science
Politehnica University of Bucharest

November 2025

Project Overview

Goal: Benchmark RAG retrieval methods for Information Retrieval

Stage 1 – Implemented:

- Production RAG application
- pgvector vector store
- Dense embeddings baseline
- NeMo Guardrails integration

Stage 1.5 – Research:

- ColBERT prototype (RAGatouille)
- Not production-integrated

Stage 2/3 – Planned:

- Full ColBERT integration
- FAISS comparison
- Sparse embeddings (BM25, SPLADE)
- Benchmark evaluation
- Performance analysis

Approach: Baseline → Prototype → Integration → Benchmarks

Current Implementation – Stage 1

System Architecture:

- **Frontend**: React 19 + Vite + TailwindCSS
- **Backend**: FastAPI + LangChain
- **LLM Serving**: vLLM (Llama-3.2-3B)
- **Vector Stores**: pgvector / Qdrant
- **Embeddings**: all-MiniLM-L6-v2 (384d)

Key Features:

- RAG toggle (enable/disable)
- PDF document upload
- Streaming chat interface
- Multiple chunking strategies
- NeMo Guardrails agents
- Dataset registry

Deployment: Docker Compose with GPU/CPU profiles

ColBERT – Late Interaction Retrieval [Stage 1.5]

Key Concept: Token-level matching

How it works:

1. Encode query & document with BERT
2. Store per-token embeddings
3. Compute MaxSim for relevance

Advantages:

- High accuracy
- Pre-computed doc embeddings
- Fine-grained semantic matching

MaxSim Score

$$S_{q,d} = \sum_i \max_j E_{q_i} \cdot E_{d_j}^T$$

Paper: Khattab & Zaharia, SIGIR 2020

Prototype Status: Standalone module using RAGatouille, not yet integrated into production

ColBERT Prototype Implementation [Stage 1.5]

Implementation Details:

- **Library**: RAGatouille (Python wrapper)
- **Model**: colbert-ir/colbertv2.0
- **Features**: Indexing + Late interaction search
- **Max doc length**: 512 characters
- **File**: `colbert_retriever.py`

Current Status:

- ✓ Standalone retriever module
- ✓ Test scripts working
- × Not integrated into FastAPI
- × Not accessible via UI
- × No benchmarks vs current system

Next Steps: API integration, benchmarking, production deployment

FAISS – Similarity Search at Scale [Stage 2]

Key Concept: Approximate nearest neighbor

Index Types:

- **Flat**: Exact search
- **IVF**: Inverted file
- **HNSW**: Graph-based
- **PQ**: Compressed

Advantages:

- Very fast queries
- GPU acceleration
- Billion-scale support
- Memory efficient

Paper: Johnson et al., 2017

Current: Using pgvector (PostgreSQL) & Qdrant for vector storage

Embedders & LLMs

Current Implementation:

- **Embedder**: all-MiniLM-L6-v2
  - 384 dimensions
  - $\sim$50ms CPU, $\sim$10ms GPU
- **LLM**: Llama-3.2-3B-Instruct
  - Served via vLLM
  - Streaming responses
- Multi-embedder architecture ready

Planned Additions [Stage 2/3]:

- **Sparse**: BM25, SPLADE
- **Dense**: bge-base, e5-base
- **LLMs**: Mistral, Phi-3
- **Evaluation Datasets**:
  - Natural Questions (NQ)
  - MS MARCO

Technology Stack – Current Implementation

Component	Tech
Backend	FastAPI + uvicorn
Frontend	React 19 + Vite
Styling	TailwindCSS
LLM Serving	vLLM
RAG	LangChain
Vectors (Prod)	pgvector
ColBERT (Proto)	RAGatouille
Guardrails	NeMo 0.13.0
Database	PostgreSQL 16

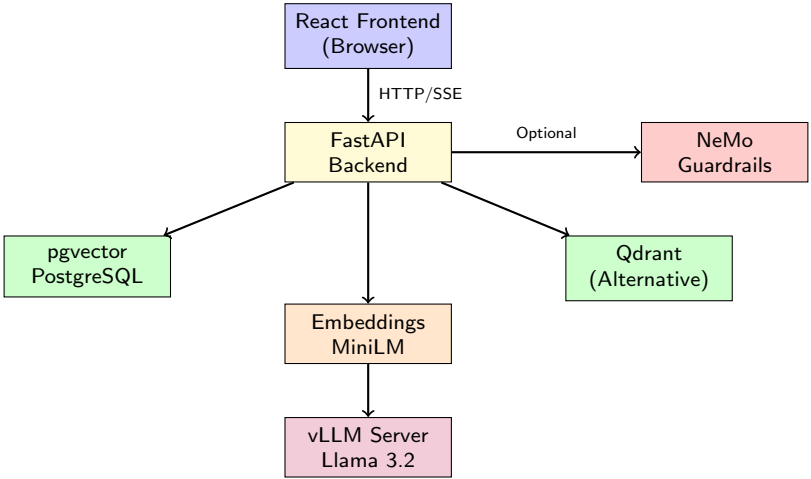
Key Libraries:

- sentence-transformers
- PyPDF (document processing)
- Server-Sent Events (SSE)
- RAGatouille (ColBERT prototype)

Deployment:

- Docker Compose
- GPU & CPU profiles
- Package manager: uv

System Architecture – Current Implementation



Flow: User query → Embed → Vector search → RAG context → LLM → Stream response

Key Features & RAG Implementation

User-Facing Features:

- Chat interface with streaming
- RAG toggle (enable/disable)
- PDF document upload
- Multiple chat sessions
- Dark/light theme support
- NeMo Guardrails agents

RAG Components:

- **Chunking**:
  - Recursive (default)
  - Fixed-size
  - Semantic
- **Chunk size**: 1000 chars
- **Overlap**: 200 chars (20%)
- **Retrieval**: Top-k=3, cosine similarity
- Dataset registry
- Vector store migration tools

Evaluation Datasets & Metrics [Stage 3]

Evaluation Datasets:

- **Natural Questions (NQ)**
  - Google queries + Wikipedia
  - Factoid QA (PDF use case)
- **MS MARCO**
  - Bing queries + 8.8M passages
  - Large-scale ranking

Current Testing:

- Qualitative evaluation
- Manual testing

Retrieval Metrics:

- **MRR@10**: Position of first relevant result
- **Recall@20**: % of relevant docs found
- **nDCG@10**: Ranking quality (best on top)

Generation Metrics:

- **BERTScore**: Semantic similarity with gold answer
- **Answer Relevance**: Does answer address query?

Efficiency:

- Query latency, memory, throughput

Challenges – Encountered & Expected

Stage 1 Challenges (Solved):

- Docker orchestration complexity
- vLLM GPU memory optimization
- PostgreSQL pgvector setup
- Frontend-backend SSE streaming
- NeMo Guardrails integration

Future Challenges (Stage 2/3):

- ColBERT index size management
- Fair comparison methodology
- Benchmark dataset preparation
- Hyperparameter tuning
- Evaluation automation

Aspect	**Current (pgvector)**	**Planned (ColBERT)**
Accuracy	Baseline	Higher (expected)
Speed	Fast	Slower
Index Size	Moderate	Larger

Project Timeline & Questions

Project Stages:

1. **Stage 1 – COMPLETED (Nov 2025)**:
   - Production RAG baseline application
   - pgvector integration
   - NeMo Guardrails implementation
   - Full-stack deployment (React + FastAPI + vLLM)
2. **Stage 1.5 – IN PROGRESS**:
   - ColBERT prototype (RAGatouille)
   - Standalone module, requires integration
3. **Stage 2 – Planned**: Fully integrate ColBERT, HippoRAG and FAISS
4. **Stage 3 – Planned**: Benchmark evaluation (MS MARCO, NQ)
5. **Stage 4 – Final**: Comparative analysis & report

Questions?

Thank you for your attention!