A Study of Embeddings, LLMs, and RAG Methods

Information Retrieval – Project Stage 1

Patrascu Adrian[1] Modiga Miriam[1]
Toderian Vitalii[1]

[1]Faculty of Automatic Control and Computer Science
Politehnica University of Bucharest

November 2024

1 Project Description

This project aims to benchmark and compare two prominent Retrieval-Augmented Generation (RAG) methods: **ColBERT** and **FAISS**. RAG systems enhance Large Language Model responses by retrieving relevant documents from a knowledge base before generating answers.

1.1 Objectives

- Run 2-3 Information Retrieval benchmarks
- Compare 3-5 sparse vs dense embedding methods
- Evaluate 3-5 open-source LLMs for generation
- Implement conversational tests using LangGraph
- Analyze trade-offs between accuracy, speed, and memory usage

1.2 Scope

We will evaluate:

- **Retrieval Methods**: ColBERT (late interaction) vs FAISS (approximate nearest neighbor)
- **Embedders**: BM25, SPLADE (sparse); MiniLM, BGE, E5 (dense)
- **LLMs**: Llama 2, Mistral 7B, Phi-2, Zephyr 7B
- **Benchmarks**: MS MARCO, Natural Questions

2 State-of-the-Art

2.1 ColBERT – Late Interaction Retrieval

ColBERT [1] introduces a "late interaction" architecture that independently encodes queries and documents using BERT, then computes relevance via MaxSim operation:

$$S_{q,d} = \sum_i \max_j E_{q_i} \cdot E_{d_j}^T \tag{1}$$

Key advantages: Pre-computed document embeddings, token-level matching, scalable to large collections.

Implementations:

- Official: https://github.com/stanford-futuredata/ColBERT
- RAGatouille: https://github.com/AnswerDotAI/RAGatouille (used in this project)

2.2 FAISS – Similarity Search at Scale

FAISS [2] is Facebook's library for efficient similarity search in high-dimensional spaces. It supports multiple index types:

- **Flat**: Exact search (baseline)
- **IVF**: Inverted file index for faster search
- **HNSW**: Graph-based approximate search
- **PQ**: Product quantization for compression

Implementation: https://github.com/facebookresearch/faiss

2.3 Related Work

- **DPR** [3]: Dense Passage Retrieval for open-domain QA
- **SPLADE** [4]: Sparse lexical and expansion model
- **BEIR** [5]: Benchmark for zero-shot IR evaluation
- **LangChain/LangGraph**: Frameworks for building LLM applications

3 Technologies

3.1 Core Stack

Component	Technology
Language	Python 3.10+
Deep Learning	PyTorch 2.0+
LLM Framework	Transformers, vLLM
RAG Pipeline	LangChain, LangGraph
ColBERT	RAGatouille (colbert-ir/colbertv2.0)
Vector Search	FAISS (CPU/GPU), PGVector
Embeddings	Sentence-Transformers
Evaluation	datasets, ranx

3.2 Models

Embedders:

- Sparse: BM25, SPLADE
- Dense: all-MiniLM-L6-v2, bge-base-en, e5-base-v2

LLMs: Llama 2 7B, Mistral 7B, Phi-2, Zephyr 7B

4 System Architecture

4.1 Pipeline Components

1. **Ingestion**: Load documents, chunk text, generate embeddings
2. **Indexing**: Build ColBERT index or FAISS index
3. **Retrieval**: Search for relevant passages given query
4. **Generation**: Use LLM to generate answer from context
5. **Evaluation**: Compute metrics (MRR, Recall, NDCG)

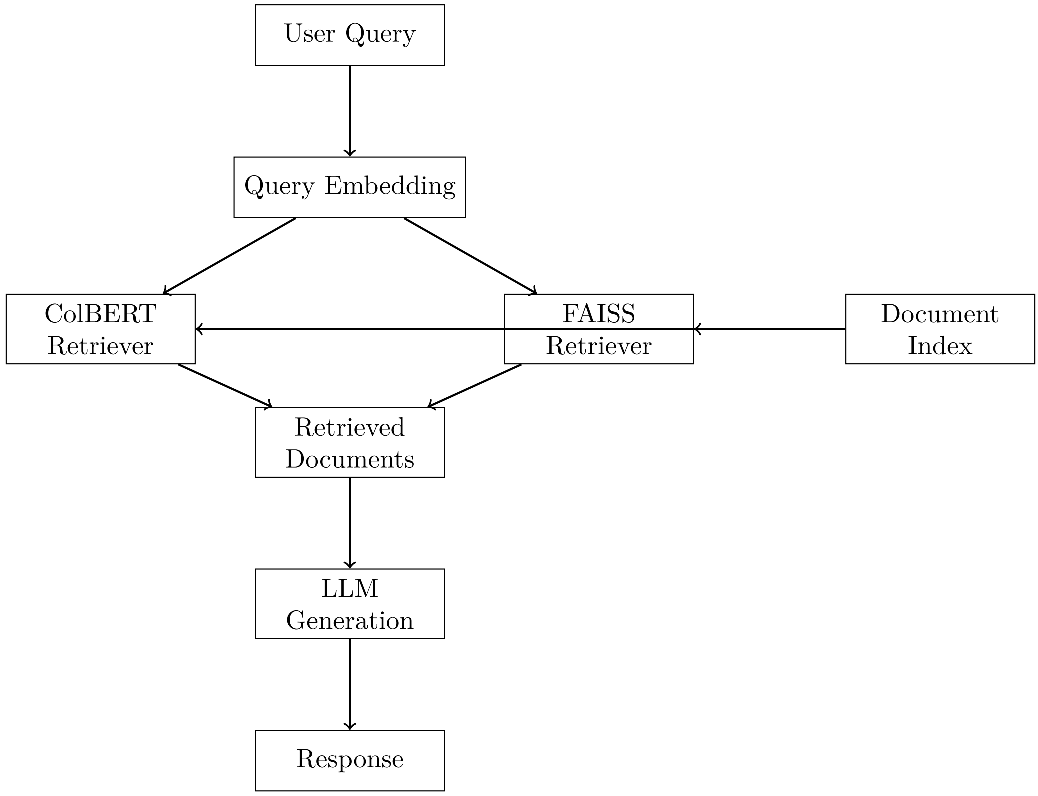

Figure 1: RAG System Architecture

4.2 LangGraph Integration

LangGraph orchestrates the conversational RAG pipeline:

- State management across conversation turns
- Conditional routing between retrievers
- History-aware query reformulation

4.3 Implementation Status

4.3.1 Current Progress

As of Milestone 1, the following components have been implemented on the `miriam/colbert-integration` branch:

- **ColBERT Retriever Module**: Complete implementation using RAGatouille library with the official ColBERTv2.0 model (`colbert-ir/colbertv2.0`)
- **Core Functionality**: Document indexing with late interaction search, returning ranked results with relevance scores
- **Test Suite**: Validation scripts demonstrating indexing and retrieval on sample datasets
- **Vector Database**: PGVector backend for FAISS integration
- **RAG Pipeline**: LangChain-based retrieval-augmented generation with vLLM

4.3.2 Technical Implementation Details

ColBERT Configuration:

- Model: `colbert-ir/colbertv2.0`
- Library: RAGatouille (Python wrapper)

- Max document length: 512 tokens (configurable)
- Index storage: Local filesystem (`.ragatouille` directory)
- LangChain integration: Full compatibility with Document objects

Code Organization:

- `backend/colbert_retriever.py`: Core retriever implementation (101 lines)
- `backend/test_colbert.py`: Integration testing
- `backend/rag_components.py`: RAG pipeline orchestration
- `backend/multi_embedder_manager.py`: Multi-model embedding support

4.3.3 Ongoing Development

The following features are currently under development:

- Integration of ColBERT retriever with main FastAPI application
- API endpoints for ColBERT document indexing and search
- Reranking functionality for hybrid retrieval approaches
- Comparative benchmarking framework between ColBERT and FAISS
- LangGraph integration for conversational RAG with multiple retrieval strategies

5 Potential Challenges

5.1 Technical Challenges

1. **Memory constraints**: ColBERT indexes can be large (storing per-token embeddings). Solution: Use compression, quantization.

2. **GPU requirements**: Running multiple LLMs requires significant VRAM. Solution: Use quantized models (4-bit), CPU offloading.

3. **Indexing time**: Building ColBERT indexes is slower than FAISS. Solution: Pre-build indexes, use batch processing.

4. **Benchmark consistency**: Ensuring fair comparison across different methods. Solution: Standardized evaluation scripts, same hardware.

5.2 Methodological Challenges

1. **Hyperparameter tuning**: Each method has different optimal settings (chunk size, top-k, temperature). Solution: Grid search on validation set.

2. **Metric selection**: Different metrics favor different systems. Solution: Report multiple metrics (MRR, Recall, latency).

3. **LLM variability**: Generation quality varies with prompts. Solution: Use consistent prompt templates.

5.3 Expected Trade-offs

6 Timeline and Deliverables

6.1 Project Stages

1. **Stage 1** (Current): Project description, architecture, technology selection
2. **Stage 2**: Implementation of retrieval pipelines

Aspect	ColBERT	FAISS
Retrieval accuracy	Higher	Lower
Query latency	Higher	Lower
Index size	Larger	Smaller
Setup complexity	More complex	Simpler

Table 1: Expected Trade-offs

3. **Stage 3**: Benchmark evaluation and results analysis
4. **Final**: Complete report with findings and recommendations

6.2 Milestone 1 Achievements

- **Architecture Design**: Complete system architecture for dual-retriever RAG system
- **Technology Stack**: Selection and validation of core technologies (RAGatouille, FAISS, vLLM, LangChain)
- **ColBERT Implementation**: Working ColBERT retriever module with indexing and search capabilities
- **Development Environment**: Dockerized setup with PostgreSQL, vLLM, and frontend
- **Test Framework**: Unit tests and integration tests for retrieval components

6.3 Next Steps for Stage 2

- Merge ColBERT integration branch into main codebase
- Implement unified API endpoints for both retrieval methods
- Develop comparative benchmarking scripts using MS MARCO dataset
- Create evaluation metrics dashboard
- Integrate multiple embedding models (BM25, SPLADE, MiniLM, BGE, E5)
- Conduct initial performance testing on target hardware

References

[1] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over bert," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 39–48, ACM, 2020.

[2] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[3] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

[4] T. Formal, B. Piwowarski, and S. Clinchant, "Splade: Sparse lexical and expansion model for first stage ranking," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2288–2292, ACM, 2021.

[5] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.