Phase 2: Naive RAG Implementation Report

Overview

This implementation establishes a functional Retrieval-Augmented Generation pipeline using the RAG Mini Wikipedia dataset. The system employs sentence-transformers for semantic embeddings and FAISS for efficient vector similarity search, providing the foundation for question-answering tasks in subsequent phases.

Technical Implementation

1. **Embedding Generation**

The system uses the all-MiniLM-L6-v2 model from sentence-transformers, producing 384-dimensional embeddings. All 3,200 Wikipedia passages were encoded in batches of 32, with L2 normalization applied to enable cosine similarity computation. The embedding process completed in approximately 2-3 minutes on CPU.

2. **Vector Database**

FAISS IndexFlatIP provides exact nearest-neighbor search through inner product computation on normalized vectors. This configuration ensures accurate retrieval for our corpus size without requiring approximate search methods. The complete index, containing all passage embeddings, was serialized to disk (5MB) for reuse across sessions, eliminating redundant computation in future phases.

3. **Retrieval Performance**

Initial testing with sample queries demonstrates effective semantic matching. Queries about well-known facts (e.g., "What is the capital of France?") consistently retrieve relevant passages with similarity scores above 0.70, indicating strong embedding quality.

4. Code Organization and Error Handling

The implementation follows object-oriented design through the NaiveRAG class, encapsulating all pipeline operations with comprehensive error handling. Each major component includes validation logic: dataset loading verifies required columns exist, embedding generation checks for empty outputs, and index construction validates vector counts match expectations. All operations are logged at appropriate levels (INFO

for major steps, WARNING for parameter adjustments, ERROR for failures) to both console and file, enabling debugging and performance monitoring.

Configuration management through YAML externalization allows parameter adjustment without code modification. Key configurable elements include embedding batch size, retrieval top-k values, similarity thresholds, device selection, and logging verbosity.