LiteRAG: A Lightweight RAG Framework

1. Introduction

LiteRAG is a lightweight information retrieval and retrieval-augmented generation (RAG) framework designed for edge nodes and low-memory environments.

Initially developed as a search engine supporting TF-IDF ranking and classical information retrieval (IR), LiteRAG's long-term goal is to evolve into a resource-optimized RAG system capable of running efficiently on constrained hardware such as Raspberry Pi, IoT gateways, or single-board computers.

2. Current Capabilities

A. Parsing

- Supports parsing of large text corpora such as Wikipedia XML dumps.
- Extracts document ID, title, and body content.
- Batches parsed output into compressed .json.gz files for downstream processing.
- Includes optional checkpointing and progress logging.

B. Tokenization

- Implements diacritics removal, lowercasing, and punctuation normalization.
- Removes English stop words from a configurable list.
- Integrates a Porter Stemmer for term normalization.
- Outputs tokenized data as structured JSON documents with per-document token lists.

C. Indexing (SPIMI)

- Builds Single-Pass In-Memory Index (SPIMI) blocks.

- Each block stores:
    - Term → { docID → frequency }
- Supports periodic flushing of blocks to compressed disk batches (.json.gz) once memory threshold is exceeded.
- Includes tiered merge using a min-heap k-way merge strategy for scalable index consolidation.
- Index compression is done using Delta encoding, this has lead to an approx 50% reduction in space needed to store the index.
- Token to offset indexing allowing for immediate seek for file offset in index based on token we are searching for.
- The entire token offset file is loaded into a hashmap on engine initialization, this can be optimized to use lesser space later on maybe with an FST.
- The inverted index is not compressed for now, we can look into block based indexing for scalability and optimizations.

---

3. Architecture Overview

Insert a well written read/write path arch here. For clarity.

4. Roadmap / Next Steps (Phase 1)

- Make the data queryable and implement IR and TF-IDF.
    - Implement doc stats.
- Implement a basic UI for the query engine.
- Look into how to pivot to RAG.
- Add multithreading, better architecture, right now its very simple.
- Add more complex flushing mechanisms while indexing.
- ~~Compress the generated indexes, currently they're taking up a lot of space~~
- Come up with the query processing core.

5. Technical Goals

7. Vision

LiteRAG aims to redefine RAG accessibility bringing retrieval-augmented systems to low-cost, low-power devices without relying on large cloud models or heavy vector databases.