Big Data

Assignment 2 Report

Author: Nazgul Salikhova (n.salikhova@innopolis.university)

Group: B22-AAI-02

Methodology

The assignment focuses on building a basic information retrieval pipeline using a distributed architecture. We leverage **Apache Spark** for data processing, **Hadoop HDFS** for storage, and **Apache Cassandra** for storing the indexed data and scores. The goal is to preprocess a document collection, extract relevant statistics, and compute BM25 scores for efficient document retrieval. The project is not done, because only 1st stage is ready - Indexing

Data Preparation

The documents are loaded into **HDFS**, where Spark jobs access them for tokenization, normalization, and stop-word removal. Each document is assigned a unique doc_id and its tokens are extracted for indexing.

1) Indexing

Using Spark, we compute the following:

- Vocabulary statistics: document frequency and total occurrences of each term.
- **Document statistics**: length, title, and average term frequency per document.
- **Inverted index**: mapping terms to the documents they appear in along with term frequency and word positions.
- **BM25 scores**: relevance scores for each (term, document) pair using the BM25 algorithm.

Cassandra Table Description

To store the information obtained during document indexing, a **keyspace** named search_index was created in Cassandra. It uses the NetworkTopologyStrategy replication strategy with one replica in datacenter1.

1. vocabulary

Stores statistics about individual terms across the document collection:

- o term the text value of the term (primary key).
- o document_frequency the number of documents that contain the term.
- total_occurrences the total number of times the term appears in the entire collection.

2. document_stats

Contains metadata for each document:

- o doc_id the unique identifier of the document (primary key).
- o title the title of the document.
- o doc length the total number of words in the document.
- avg_term_frequency the average frequency of terms in the document (total term frequency divided by the number of unique terms).

3. inverted index

Represents a classic inverted index:

- o term the term (part of the composite primary key).
- doc_id the ID of the document where the term appears (second part of the composite key).
- term_frequency the number of times the term appears in the document.
- o positions a list of word positions in the document where the term occurs.

4. bm25_scores

Stores precomputed BM25 relevance scores for each term-document pair:

- o term the term text.
- o doc id the document ID.
- score the BM25 score representing the relevance of the document to the term

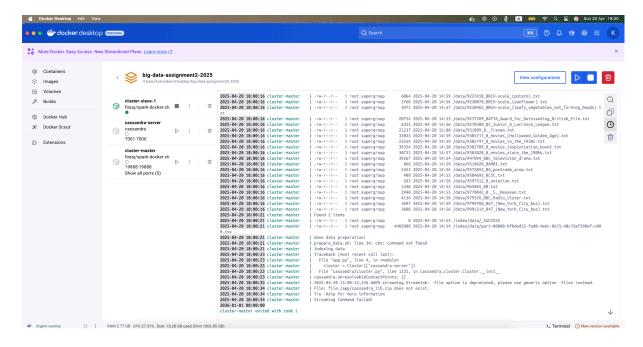
These tables provide an efficient structure for implementing full-text search, term-based lookups using an inverted index, and document ranking based on relevance metrics.

How to Run

- 1. Clone the repository
- 2. Start the system using Docker Compose:
 - docker-compose up -d

Demonstration

Data Preparation:



Data Indexing:

