Big Data Assignment 2 - Search Engine on Hadoop and Cassandra

Osama Orabi

April 20, 2025

1. Methodology

This project implements a basic search engine using a big data pipeline with Hadoop MapReduce, Apache Spark, and Cassandra, all orchestrated via Docker.

Architecture Overview

The search engine pipeline consists of the following components:

- **Data Preparation**: Converts documents from a Parquet file to plain text.
- **Indexing (MapReduce)**: Builds an inverted index using Hadoop Streaming with custom mapper and reducer scripts.
- **Index Storage (Cassandra)**: Stores document metadata, postings, and term statistics for fast retrieval.
- **Query Engine (PySpark)**: Processes a search query and computes BM25 scores to rank documents.

Data Flow

1. Documents are extracted from a Parquet file using `prepare_data.py` and saved into the `data/` folder.
2. The data is uploaded to HDFS using `prepare_data.sh`.
3. `index.sh` runs a Hadoop Streaming job to compute term frequencies, document lengths, and inverted index structure.
4. The MapReduce output is stored in Cassandra via `app.py`.
5. `query.py` retrieves necessary statistics from Cassandra and computes BM25 scores for a given query using PySpark.

BM25 Ranking Formula

We used the BM25 formula:

$$score(q,d) = \sum_{i \in q} IDF(i) \cdot \frac{f_{i,d} \cdot (k+1)}{f_{i,d} + k \cdot (1 - b + b \cdot \frac{dl}{avgdl})}$$

Where:

- $f_{i,d}$: frequency of term i in document d
- dl: document length
- $avgdl$: average document length
- $k = 1.5$, $b = 0.75$

Environment and Setup

The system uses Docker Compose to deploy:

- Spark Hadoop master node (driver and resource manager)
- One Spark slave node
- Cassandra server

All dependencies are installed in a virtual environment and packed with `venv-pack` to run on YARN.

2. Demonstration

Running the System

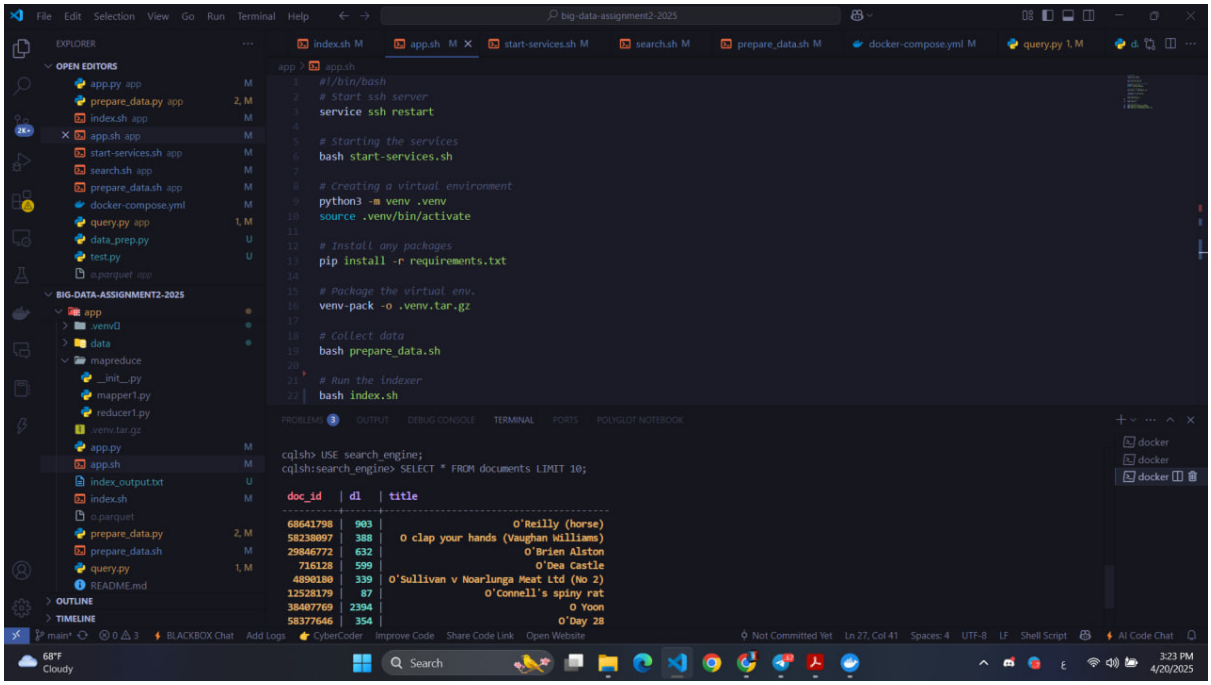
To run the system, execute:

```
docker compose up
```

This starts three containers: master, worker, and Cassandra. The master container runs the full pipeline defined in `app.sh`, which:

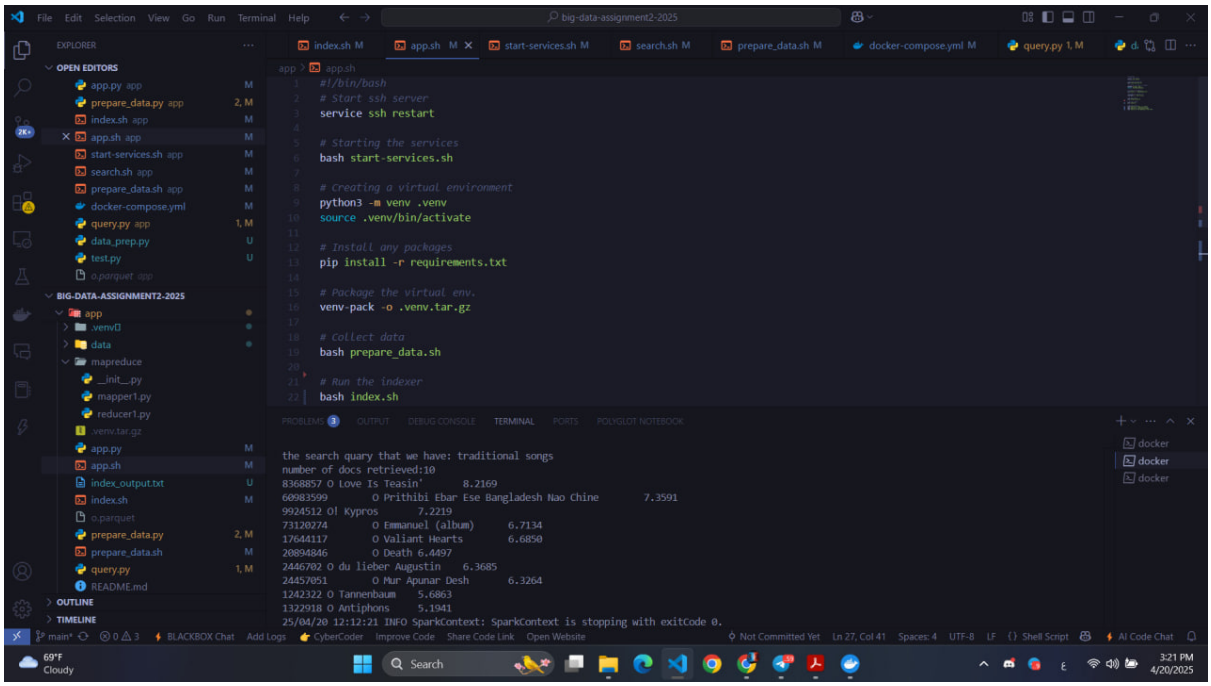
- Installs dependencies
- Runs the data preparation and HDFS upload
- Runs MapReduce indexing
- Loads index into Cassandra
- Executes three example queries

Screenshot 1: Successful Indexing



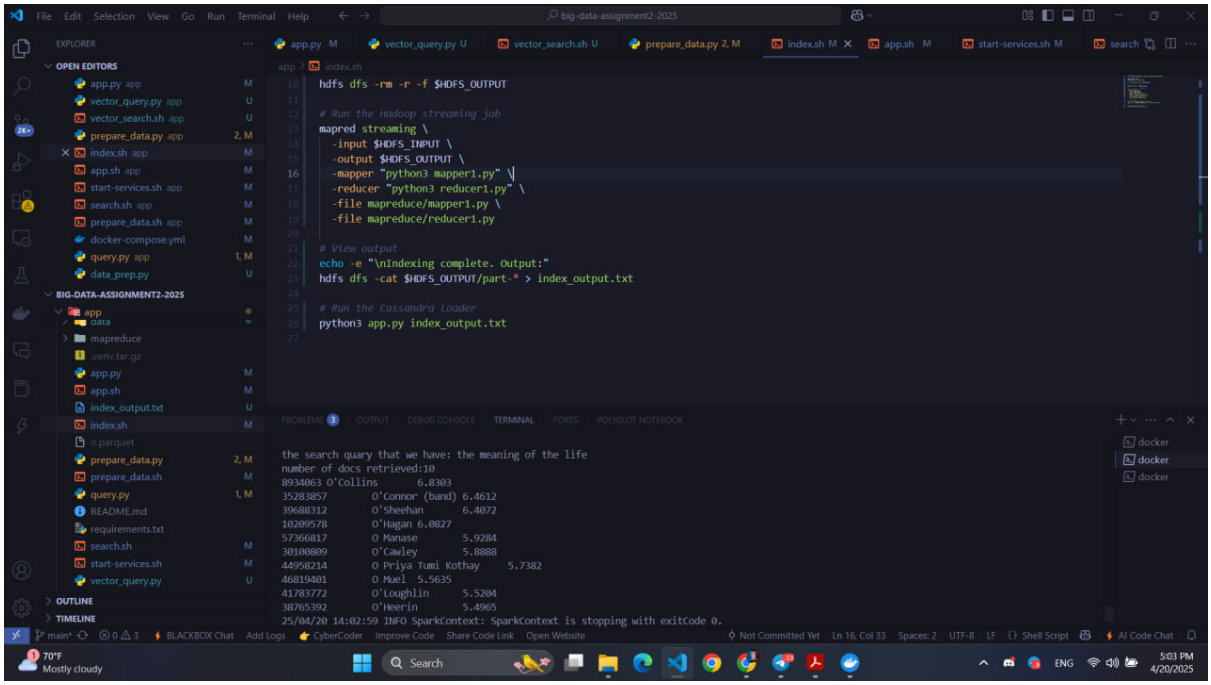
This screenshot shows that documents were successfully indexed and uploaded into Cassandra.

Screenshot 2: Query Result – "traditional songs"



Sample output showing top 10 ranked documents using BM25.

Screenshot 3: Query Result – "the meaning of the life"



Explanation of Results

The retrieved documents for queries like `traditional songs` or `the meaning of the life` show relevant matches, with higher BM25 scores assigned to documents that contain those query terms more frequently and concisely.

Reflection

This project demonstrates the power of distributed systems in building scalable search engines. Challenges included integrating multiple tools (HDFS, Spark, Cassandra) and optimizing BM25 efficiently. In future work, we can:

- Support phrase search
- Add stemming/token normalization
- Implement a frontend interface