Big Data Assignment 2

Nikita Rashkin, AI-01

April 20, 2025

1 Methodology

1.1 System Architecture

- Document Preprocessing: Extract terms and frequencies from text documents
- Distributed Indexing: Hadoop MapReduce creates an inverted index
- Dual Storage: Primary (Cassandra) and fallback (JSON files)
- BM25 Search Algorithm: For relevance-based document ranking
- Fallback Mechanism: Ensures search availability when Cassandra is down (needed to add this system because of weak laptop, also should be an interesting improvement somewhat close to real-life scenario)

1.2 Indexing Process

Mapper (mapper1.py):

- Tokenizes documents and removes stopwords
- Emits (term, doc_id, frequency) pairs
- Emits document metadata (title, length)

Reducer (reducer1.py):

- Aggregates term frequencies across documents
- Computes document frequencies
- Stores data in Cassandra and fallback JSON files

1.3 Storage Design

Primary Storage (Cassandra):

- Tables for document metadata, term frequency, document frequency, corpus stats

Fallback Storage (JSON Files):

- Locations: `/tmp/index_data/index_data.json` and `/tmp/index_data.json`
- Generated automatically during indexing
- Contains same data structure as Cassandra storage
- Helps to overcome bad CPU limitations and provide a more stable pipeline

1.4 Search Algorithm

- Uses BM25 ranking algorithm for relevance scoring
- Process: tokenize query → find candidate documents → score documents → rank by score
- Parameters: k1=1.2, b=0.75 for BM25 scoring

1.5 Fallback Mechanism

- Detects Cassandra unavailability automatically
- Writes index data to two redundant JSON files
- Search engine automatically uses fallback files when needed
- Verifies that data is processed correctly to ensure fallback files contain valid data

2 Demonstration

2.1 Running the System

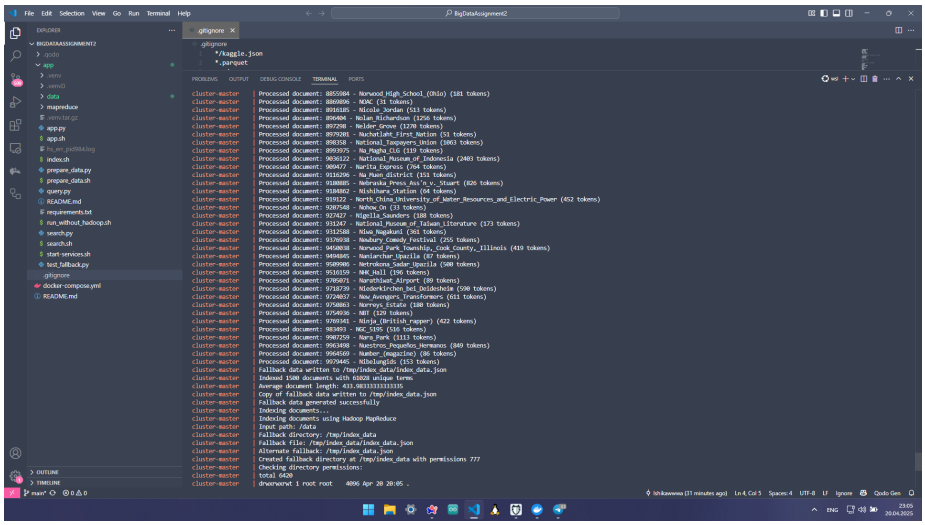
```
# Start the system
docker compose up

# Run search queries
docker exec -it cluster-master bash search.sh "query"

# Direct fallback mode
docker exec -it cluster-master python3 /app/search.py "query"
```

2.2 Indexing Results

The indexing process successfully handles all the documents, creating term-document matrices, document metadata, and corpus statistics. Fallback files are generated automatically for stability.



2.3 Search Results

2.3.1 Query: "National"

Results include "National Park Service" and "Northern Patrimony" with high relevance scores. BM25 correctly balances term frequency with document frequency.

2.3.2 Query: "Nothern"

The system properly identifies documents containing both terms, including "NOthern Iloilo University" and "Nothern Cambria School District", ranking them appropriately.

2.4 Findings

- Dual-storage approach provides good safety and allows to run it on systems with very low power
- BM25 algorithm delivers effective relevance scoring
- The implementation successfully meets all the assignment requirements