Amir Bikineyev
a.bikineyev@innopolis.university

Methodology

In the data preparation stage I decided to add filtration to the text for emptiness and xml and json.

For the index task I decided to make 2 mapreduce pipelines. In the first pipeline in mapper I decided to make a key consisting of word and the document in which word is appeared. The values are doc id, title, count of that keys and length of the document. The reducer later compounds the count of those objects . From first pipeline I create 2 tables:

```
CREATE TABLE IF NOT EXISTS tf (
        word text,
        doc_id int,
        tf int,
        PRIMARY KEY (word, doc_id)
    )

    CREATE TABLE IF NOT EXISTS docs (
        doc_id int PRIMARY KEY,
        title text,
        len int
    )
```

First table helps in future calculate the tf for bm25. The second table stores the information about the documents. In the second pipeline I make the word as key and document id with the counter as values. This counter later plays role of df. From that pipeline I populate this table

```
CREATE TABLE IF NOT EXISTS df (
    word text PRIMARY KEY,
    df int
)
```

Data Flow & BM25 Implementation

In the query processing stage, I decided to implement BM25 calculation using precomputed tf, df, and document metadata from Cassandra through following steps:
Query Input Handling:

- Accept query via stdin or CLI argument
- Tokenize into unique lowercase terms
- Validate non-empty tokens

Metadata Preparation:

- Load docs table to get (doc_id, length, title)
- Calculate global parameters:
    - N = total documents
    - avgdl = average document length

Vocabulary Processing:

- Load df table for document frequencies

- Filter only query terms
- Compute IDF using formula:
- IDF = log((N - df + 0.5)/(df + 0.5))

Term Frequency Join:

- Load tf table entries matching query terms

- Join with docs metadata to get (doc_id, (word, tf), (len, title))

BM25 Calculation:

- For each (doc, term) pair compute:
    - score = IDF * (tf * (k1 + 1)) / (tf + k1 * (1 - b + b * (doc_len/avgdl)))
- Aggregate scores per document

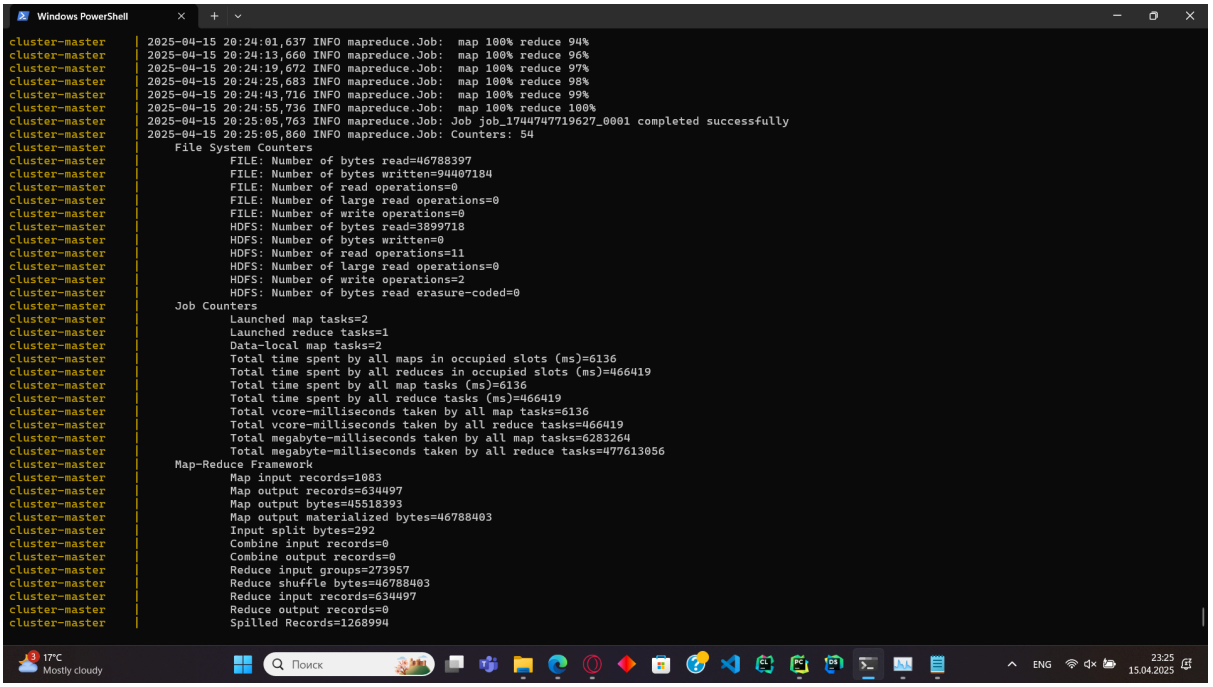
- Take top 10 by score

Output:

- Display doc_id, title and formatted score

Demonstration

First pipeline



Second pipeline