

# Big Data Assignment 2 Report

Arsenii Pavlov [ars.pavlov@innopolis.university](mailto:ars.pavlov@innopolis.university)

## Methodology

### Data Preparation

- 1) Used a Parquet file containing Wikipedia articles. Selected 4000 documents using PySpark's `sample()` and `limit()` functions.
- 2) Each document is saved as `<doc_id>_<doc_title>.txt` in HDFS. Spaces in titles are replaced with `_`.

pipeline:

- 1) Read the Parquet file into a Spark DataFrame.
- 2) Extracted id, title, and text columns.
- 3) Saved documents to HDFS to `/data` using tab-separated format `<doc_id>\t<doc_title>\t<doc_text>`.

### Indexer Tasks

The indexer uses two Hadoop MapReduce pipelines and save results in Cassandra.

#### 1) TF Calculation

##### Mapper1

- 1) Read input documents from HDFS.
- 2) Tokenizes text cast to lowercase and splits on non-alphanumeric characters.
- 3) Convert to key-value pairs: `<term>#<doc_id>` to 1.

##### Reducer1

- 1) Aggregates counts for `<term>#<doc_id>` to compute TF.
- 2) Output: `<term> <doc_id> <tf>` to `/tmp/index/pipeline1`

#### 2) DF Calculation

##### Mapper2

- 1) Reads TF data from Pipeline 1.
- 2) Convert term to 1 for each unique term-document pair.

##### Reducer2:

- 1) Sums counts to compute DF.
- 2) Output `<term>\t<df>` to `/tmp/index/pipeline2`

#### 3) Saving to Cassandra

Tables:

vocabulary: words and DF values  
inverted\_index words and TF values  
documents: map document IDs and titles

**Loading Data:** app.py reads HDFS outputs and inserts data into Cassandra using batch queries.

## Ranker Tasks

ranker use PySpark to compute BM25 scores for queries.

## BM25 Calculation

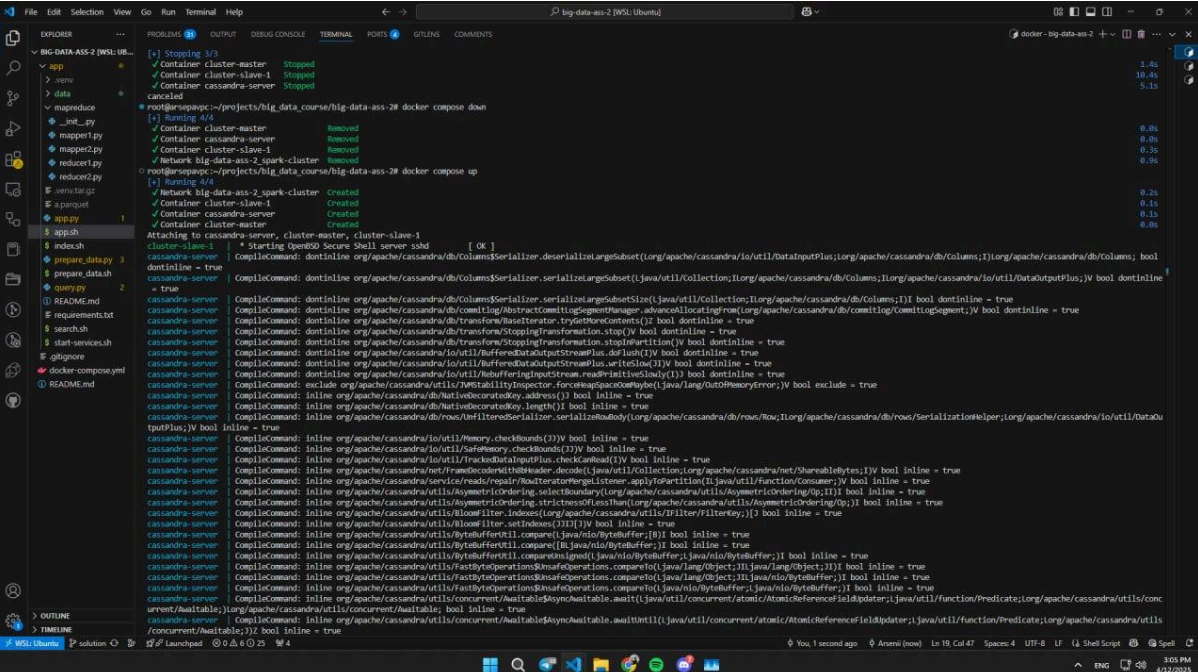
- 1) Fetch DF for query terms from vocabulary table.
- 2) Fetch TF and document lengths from inverted\_index and documents.
- 3) Compute BM25 score using provided formula
- 4) Sum scores across all query terms.
- 5) Rank documents by score and return top 10.

## Spark

- 1) Join vocabulary and inverted\_index.
- 2) Apply BM25 formula using Spark SQL functions.
- 3) Join results with `documents` table to get titles.

## Demonstration

To start containers run  
docker compose up



It will run all containers and run data preparation indexing and 1 query

[illegible][illegible]

```
root@arsapac:~/projects/big_data_course/big-data-ss-2# git add .
root@arsapac:~/projects/big_data_course/big-data-ss-2# git commit -m "f"
[main 888888f]
2 files changed, 4 insertions(+), 4 deletions(-)
root@arsapac:~/projects/big_data_course/big-data-ss-2# git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 421 bytes | 421.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving url: https://github.com:root@arsapac:~/projects/big_data_course/big-data-ss-2.git
To github.com:arsapac:~/projects/big_data_course/big-data-ss-2
root@arsapac:~/projects/big_data_course/big-data-ss-2# docker exec -it cluster-master bash
root@cluster-master:/app# bash search.sh hello
This script will include commands to search for documents given the query using Spark RDD
BIG_DATA_APP: doc: 529355, title: A Chorus Line
BIG_DATA_APP: doc: 6421145, title: A Couple of Cuckoos
BIG_DATA_APP: doc: 407262, title: A Connecticut Yankee in King Arthur's Court
BIG_DATA_APP: doc: 2588060, title: A Crazy Steal
BIG_DATA_APP: doc: 11985911, title: A Flash Flood of Colour
BIG_DATA_APP: doc: 5372634, title: A Gun of Day
BIG_DATA_APP: doc: 53756723, title: A Doll's House, Part 2
BIG_DATA_APP: doc: 3075232, title: A Girl Called Sonny
BIG_DATA_APP: doc: 12581726, title: A Beautiful Morning
BIG_DATA_APP: doc: 55178018, title: A Boogie wit da Hoodie discography
root@cluster-master:/app# bash search.sh mouse
This script will include commands to search for documents given the query using Spark RDD
BIG_DATA_APP: Search result:
BIG_DATA_APP: doc: 10988823, title: A Caprice
BIG_DATA_APP: doc: 57647084, title: A Close Call
BIG_DATA_APP: doc: 69184219, title: A Disney Halloween (1983 special)
BIG_DATA_APP: doc: 266862, title: A Kid in King Arthur's Court
BIG_DATA_APP: doc: 80541045, title: A Boy Called Christmas
BIG_DATA_APP: doc: 2064880, title: A Healthy Distrust
BIG_DATA_APP: doc: 3881893, title: A (Pretty Little) Liar(s)
BIG_DATA_APP: doc: 28721446, title: A Day in the Dime Potomac Grand Prix
BIG_DATA_APP: doc: 14183608, title: A Disney Christmas Gift
BIG_DATA_APP: doc: 10576284, title: A Gun Called Tension
root@cluster-master:/app# bash search.sh "computer mouse"
This script will include commands to search for documents given the query using Spark RDD
BIG_DATA_APP: Search result:
BIG_DATA_APP: doc: 14115572, title: A Computer Animated Hand
BIG_DATA_APP: doc: 469842, title: A Bug's Life
BIG_DATA_APP: doc: 48227628, title: A B.P. Shocker: Ali
BIG_DATA_APP: doc: 39381662, title: A Glitch Is a Glitch
BIG_DATA_APP: doc: 38279, title: A Commentary on the UNIX Operating System
BIG_DATA_APP: doc: 714151, title: A Deepness in the Sky
BIG_DATA_APP: doc: 7862753, title: A Christmas Carol (2006 film)
BIG_DATA_APP: doc: 12588454, title: A.K. Peters
BIG_DATA_APP: doc: 67425628, title: A Grandchild's Guide to Using Grandpa's Computer
BIG_DATA_APP: doc: 40347186, title: A Line in the Sand (video game)
```

The results are actually good it can be seen in the example with computer mouse titles (they all connected to computer)