Assignment 2

Arina Goncharova B22-DS-02

April 2025

1 Methodology

1.1 Data preparation

In this stage the data is formatted to include document id, title and content as tab separated values. After that it is stored in hdfs.

1.2 Indexing

I decided to use only 1 MapReduce pipeline because it was enough for me.

- mapper1.py

  Processes the files from the folder /index/data and produces the lines in the format: word \t document id. .

- reducer1.py

  Takes as input the lines produced by mapper, reduces them to find values needed for BM25 calculation and stores them in Cassandra tables.

1.2.1 Cassandra schema

I decided to use improved version of tf-idf formula - BM25.

$$BM25(q,d) = \sum_{t \in q} log \frac{N}{df(t)} \frac{k_1 \cdot \dot{tf}(t,d)}{k_1[(1-b) + b \cdot \frac{dl(d)}{dl_{avg}}] + tf(t,d)}$$

To calculate it I need:

- table term_doc_freq - $tf(t,d)$

  This table is required to get the number of times the given term occurs in the given document.

```
CREATE TABLE IF NOT EXISTS term_doc_freq (
        term TEXT,
        document_id TEXT,
        term_in_doc_frequency INT,
        PRIMARY KEY (term, document_id)
);
```

Since Cassandra is not efficient in aggregations, I cannot use term_doc_freq table for the $df(t)$, $dl(d)$, dl_{avg} and N calculation. Therefore, I decided to create separate table for this values.

- table term_freq - $df(t)$ This table stores the number of documents for each term where it occured.

```
CREATE TABLE IF NOT EXISTS term_freq (
    term TEXT PRIMARY KEY,
   term_freq INT
);
```

- table doc_length - $dl(d)$

  This table stores the length of each document.

```
CREATE TABLE IF NOT EXISTS doc_length (
        document_id TEXT PRIMARY KEY,
        document_length INT
);
```

- table general_stats - dl_{avg} and N

  This table stores the average document length and the total number of documents in the corpus. I decided to keep the average length of the document directly, since the total length of the document is not needed in this task. If it is needed, we can simply multiply the average length by the total number of documents

```
CREATE TABLE IF NOT EXISTS general_stats (
        statistics_name TEXT PRIMARY KEY,
        statistics_value DOUBLE
    );
```

1.3 Ranking

Spark RDD allows us to parallelize document processing and scoring, which makes query searching more efficient when we have many large documents.

1.3.1 Score calculation

To calculate score of the document for the given query I am using BM25 score. I added $epsilon = 1e-6$ to the denominators to avoid division by 0.

```
# epsilon is added to avoid division by 0
def compute_BM25(total_num_of_documents, dl_avg, q_words,
term_freq_dict, term_doc_freq_dict, doc_length_dict, document_id,
k=1, b=0.75, epsilon=1e-6):
    result = 0
```

```
if document_id in doc_length_dict:
    dl = doc_length_dict[document_id]
else:
    dl = 0

for word in q_words:
    if word in term_freq_dict:
        df_t = len(term_doc_freq_dict[word])
        if document_id in term_doc_freq_dict[word]:
            tf_t_d = term_freq_dict[word][document_id]
        else:
            tf_t_d = 0
    else:
        df_t = 0
        tf_t_d = 0
    result += math.log(total_num_of_documents/(df_t+epsilon)) *
    (k + 1) * tf_t_d /(k*(1-b + b*dl/dl_avg) + tf_t_d + epsilon)
return result
```

2 Demonstration

2.1 Run the repository

2.1.1 Use ready search engine with predefined query

1. Clone the repository

```
git clone https://github.com/arinagoncharova2005/big-
    data-assignment2-2025.git
```

2. Go to main directory

```
cd big-data-assignment2-2025
```

3. Start the app

```
docker-compose up
```

2.1.2 Search documents with custom query

1. Open master node terminal ans start the script

```
    docker exec -it cluster-master
    bash search.sh "<your query>"
```

2.1.3 Index document

1. Open master node terminal ans start the script

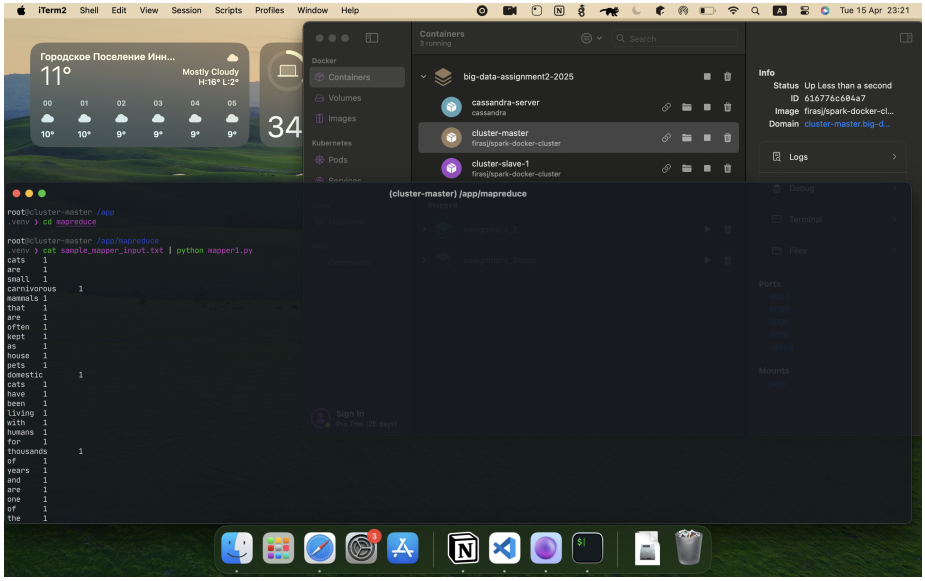
```
docker exec -it cluster-master
bash index.sh "<path to file>"(optional)
```

2.2 Results

2.2.1 Uploading data to hdfs

2.2.2 Mapper

To test the mapper I passed a small document in the format produced by data preparation step. File with this example can be found in the repository in folder mapreduce with name sample_mapper_input.txt.



2.2.3 Reducer

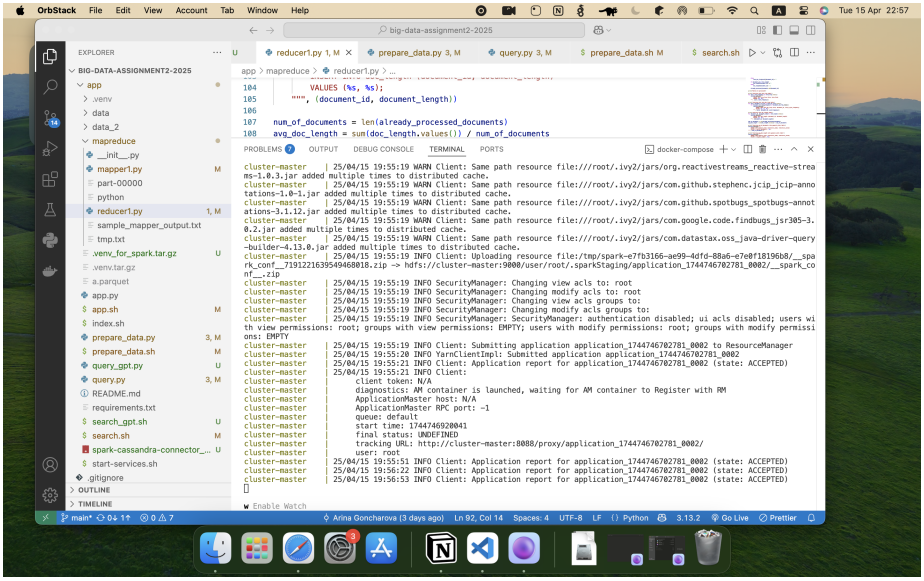
I passed a small document to the reducer to check that it works. This file with example can be found in the repository in folder mapreduce with name sample_mapper_output.txt.

2.2.4 Indexing

```
cd big-data-assignment2-2025
cd app
bash prepare_data.sh
bash index.sh
```

2.2.5 Search engine execution

I have a problem with execution of indexing on slave node. I suppose that problem occurs because of my arm processor.



But on master it runs (replaced –master yarn with –master local).

2.2.6 Reflection

During completion of this assignment I understood the purpose of using MapReduce framework and Spark RDD.

2.3 Difficulties and solutions

- Use OrbStack instead of Docker Desktop.

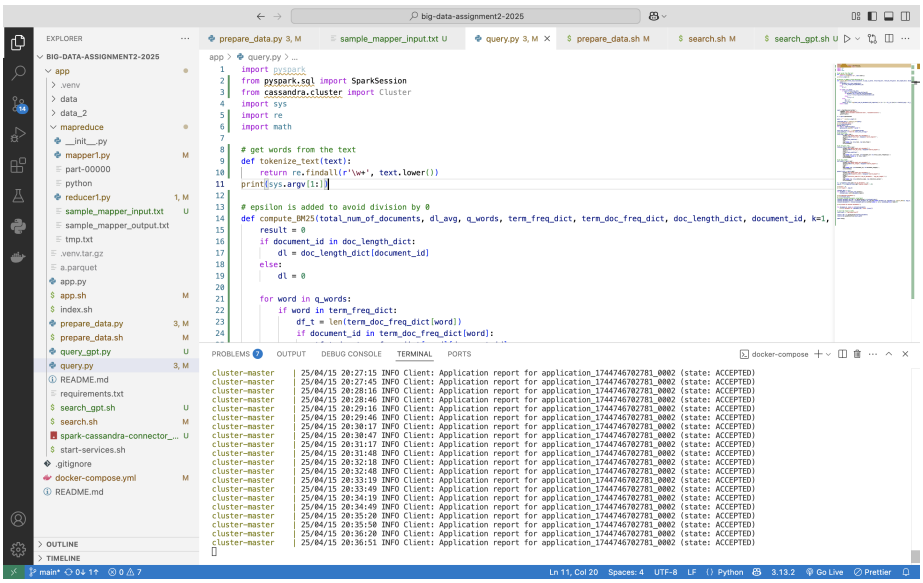
  This program offers optimized Docker usage for macOS.

- Passing python environment to slave node.

  Added arguments to the script to pass the archive with the environment

- The problem with execution on slave nodes because of the arm processor.

  I spend several evenings trying to debug my code and understand the problem. But only at the last evening I found that on linux the program executes but on my laptop it is just logging ACCEPTED.


big-data-assignment2-2025
EXPLORER
BIG-DATA-ASSIGNMENT2-2025
app
.venv
data
data_2
mapreduce
__init__.py
mapper1.py
part-00000
python
reducer1.py
sample_mapper_input.txt
sample_mapper_output.txt
tmp.txt
.venv.tar.gz
a.parquet
app.py
app.sh
index.sh
prepare_data.py
prepare_data.sh
query_gpt.py
query.py
README.md
requirements.txt
search_gpt.sh
search.sh
spark-cassandra-connector_...
start-services.sh
.gitignore
docker-compose.yml
README.md
OUTLINE
TIMELINE
prepare_data.py 3, M
sample_mapper_input.txt U
query.py 3, M
prepare_data.sh M
search.sh M
search_gpt.sh U
app > query.py > ...
import pyspark
from pyspark.sql import SparkSession
from cassandra.cluster import Cluster
import sys
import re
import math
# get words from the text
def tokenize_text(text):
return re.findall(r'\w+', text.lower())
print(sys.argv[1:])
# epsilon is added to avoid division by 0
def compute_BM25(total_num_of_documents, dl_avg, q_words, term_freq_dict, term_doc_freq_dict, doc_length_dict, document_id, k=1,
result = 0
if document_id in doc_length_dict:
dl = doc_length_dict[document_id]
else:
dl = 0
for word in q_words:
if word in term_freq_dict:
df_t = len(term_doc_freq_dict[word])
if document_id in term_doc_freq_dict[word]:
PROBLEMS 7
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
docker-compose
cluster-master | 25/04/15 20:27:15 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:27:45 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:28:16 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:28:46 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:29:16 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:29:46 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:30:17 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:30:47 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:31:17 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:31:48 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:32:18 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:32:48 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:33:19 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:33:49 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:34:19 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:34:49 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:35:20 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:35:50 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:36:20 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
cluster-master | 25/04/15 20:36:51 INFO Client: Application report for application_1744746702781_0002 (state: ACCEPTED)
main* 04 1 0 7
Ln 11, Col 20
Spaces: 4
UTF-8
LF
Python
3.13.2
Go Live
Prettier