# Report

# Big Data: Assignment 2

# Simple Search Engine using Hadoop MapReduce

Mariia Shmakova

April 14, 2025

# I Methodology

## A. Data Collection and Preparation

- Initially, a parquet file containing the ID, title, and text fields for each document was downloaded. Using PySpark, data was extracted from 1000 documents and saved in a specified file format.
- Each document was named using its id and title, with spaces replaced by underscores.
- The processed documents were stored in HDFS in the data folder and an intermediate RDD was created to store the document metadata in /index/data.

## B. Indexer task

Documents are stored in HDFS (/index/data) as plain text files. Each document is processed line-by-line to extract words. Indexing implemented using Hadoop Streaming with Python-based mapper and reducer scripts.

Hadoop Streaming was chosen for its flexibility in integrating custom mapper and reducer scripts written in Python. This allows us to process large datasets efficiently while maintaining simplicity in implementation.

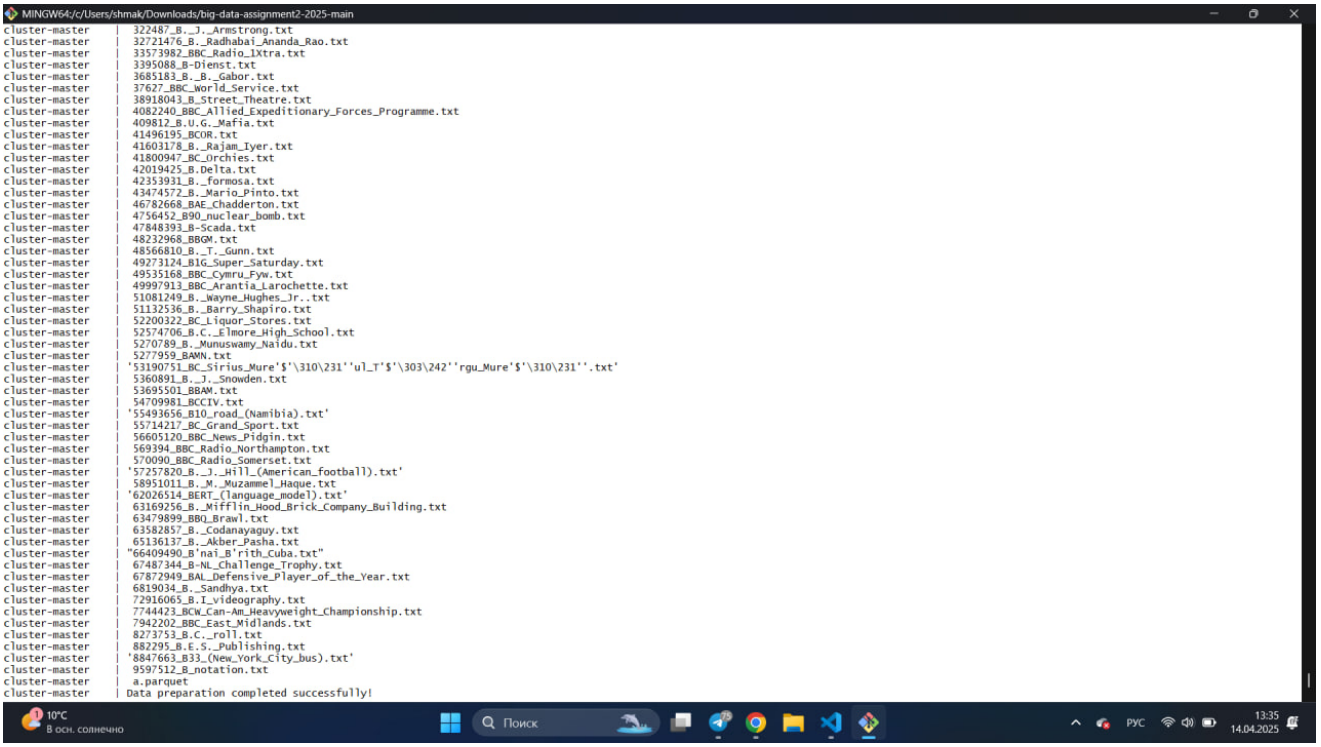
The mapper tokenizes the text into words and emits key-value pairs (word, document ID). The reducer aggregates these pairs to build the inverted index.

## C. Ranker task

- BM25 Implementation : The ranker script (query.py) calculates BM25 scores for all documents based on the user query. It retrieves data from Cassandra, computes scores, and ranks the top 10 documents.
- Broadcast Variables : To optimize performance, we used Spark broadcast variables to share BM25 statistics across all nodes.
- RDD API : The implementation relies on PySpark's RDD API for distributed computation.

# II Demonstration

## A. *Data Preparation section*



## *B. Indexer tasks section*

Running indexer: