

IBD Assignment 2 Report

Anna Gromova, DS-01

GitHub link:

Methodology

In this project, I implemented a search engine using Hadoop MapReduce, Cassandra, and Spark RDD to index and retrieve documents based on their relevance to user queries. The system consists of three main components: document indexing, query processing, and vector-based search.

1. Document Indexing

First, I processed a dataset of Wikipedia articles stored in Parquet format. Using PySpark, I extracted the text content, document IDs, and titles, then saved them in a structured format in HDFS. The indexing pipeline consists of two MapReduce jobs:

- **Term Frequency Job:** This job counts how often each word appears in every document (TF - Term Frequency). The mapper splits documents into words and emits (word, document_id, count, document_length) pairs. The reducer aggregates these counts and stores them in Cassandra.
- **Document Frequency Job:** This job calculates how many documents contain each word (DF - Document Frequency). The mapper processes the output from the first job and emits (word, document_count) pairs.

The results are stored in Cassandra tables (`term_index`, `vocabulary`, and `document_stats`) for fast retrieval during searches.

2. Query Processing

When a user submits a query, the system retrieves matching documents using the BM25 ranking algorithm, which considers:

- **Term Frequency (TF):** How often a query word appears in a document.
- **Document Frequency (DF):** How many documents contain the word (rarer words are more important).

- **Document Length Normalization:** Adjusts scores based on document length to prevent bias toward longer documents.

The algorithm calculates relevance scores for each document and returns the top 10 matches.

3. Vector Search Implementation

For the vector search part of the assignment, I extended the system to support TF-IDF vector similarity. Here's how it works:

- **Vector Indexing (`vector_indexer.py`):** I created a script that reads the term frequencies and vocabulary from Cassandra. For each document, it constructs a TF-IDF vector, where each dimension represents a word's importance in the document. The vector is stored in Cassandra as a list of floats in a `document_vectors` table.
- **Query Vectorization:** When a user submits a query, the system converts it into a TF-IDF vector using the same vocabulary. It then compares this vector against all document vectors using cosine similarity, which measures the angle between vectors (higher similarity means more relevant).
- **Cassandra Vector Storage:** I modified the Cassandra schema to include a `vector` column (type `list<float>`). The `vector_indexer.py` script computes and stores these vectors after the initial indexing phase.

Demonstration

To run the search engine, follow these steps:

First, start all services using Docker. Open a terminal and navigate to the project folder. Run the command `docker-compose up`. This will start three containers: one for Hadoop master, one for worker, and one for Cassandra. The system will automatically begin setting up everything needed.

Once the containers are running, it will:

1. Prepare the data by converting Wikipedia articles to text files
2. Run the MapReduce jobs to build the search index
3. Load all index data into Cassandra
4. Start ready to accept search queries

Here is a screenshot showing the successful loading of 1000 documents:

```

cluster-master | data/5421527_A_Life_in_the_Theatre.txt saved!
cluster-master | data/63546662_A_Life_of_Sin.txt saved!
cluster-master | data/2847432_A_Life_on_the_Ocean_Wave.txt saved!
cluster-master | data/37487965_A_Life's_Morning.txt saved!
cluster-master | data/11871426_A_Lifetime_on_More.txt saved!
cluster-master | data/36907861_A_Light_Shines.txt saved!
cluster-master | data/26346249_A_Light_in_the_Black.txt saved!
cluster-master | data/24569183_A_Limousine_the_Colour_of_Midsummer's_Eve.txt saved!
cluster-master | data/47457018_A_Line_That_Connects.txt saved!
cluster-master | data/23853328_A_Line_a_Day_Must_BeEnough!.txt saved!
cluster-master | data/25810323_A_Line_in_the_Dirt.txt saved!
cluster-master | data/53864903_A_Linguistic_Atlas_of_Early_Middle_English.txt saved!
cluster-master | data/1280654_A_Lion_Among_Men.txt saved!
cluster-master | data/4669253_A_Lion's_Hall.txt saved!
cluster-master | data/3878521_A_Little_Ain't_Enough.txt saved!
cluster-master | data/16968794_A_Little_Bit_Longer.txt saved!
cluster-master | data/37641439_A_Little_Bit_Longer_(song).txt saved!
cluster-master | data/41766912_A_Little_Bit_Zombie.txt saved!
cluster-master | data/33568894_A_Little_Bit_of_Cucumber.txt saved!
cluster-master | data/66734083_A_Little_Bit_of_Ecstasy.txt saved!
cluster-master | data/45682582_A_Little_Bit_of_Pluff_(1919_film).txt saved!
cluster-master | data/36037658_A_Little_Bit_of_Heaven_(Ronnie_Dove_song).txt saved!
cluster-master | data/42827102_A_Little_Bit_of_Love.txt saved!
cluster-master | data/41870969_A_Little_Bit_of_Love_(Andreas_Johnson_song).txt saved!
cluster-master | data/39710446_A_Little_Bit_of_Luck.txt saved!
cluster-master | data/37624429_A_Little_Bit_of_Soul_(1998_film).txt saved!
cluster-master | data/38182187_A_Little_Bit_of_Stitt.txt saved!
cluster-master | data/29033248_A_Little_Bit_of_You.txt saved!
cluster-master | data/46287936_A_Little_Boy_Lost.txt saved!

```

Строка 67, столбец 36 (выбрано 26) Пробелов: 2 UTF-8 LF Shell Script Макет: US

Here you can see the MapReduce jobs completing without errors for both Term Frequency and Document Frequency pipelines and the confirmation messages showing data loaded into Cassandra:

```

cluster-master | Done prep
cluster-master | Starting indexing...
cluster-master | Starting indexing pipeline...
cluster-master | Term Frequency pipeline
cluster-master | Deleted /tmp/term_frequencies
cluster-master | packageJobJar: [/tmp/hadoop-unjar161122644452569100/] [] /tmp/streamjob3057026759724003387.jar tmpD
ir=null
cluster-master | 2025-04-15 18:58:29,952 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager
r at cluster-master/172.18.0.4:8082
cluster-master | 2025-04-15 18:58:30,122 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager
r at cluster-master/172.18.0.4:8082
cluster-master | 2025-04-15 18:58:30,338 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/
hadoop-yarn/staging/root/_staging/job_1744743443639_0001
cluster-master | 2025-04-15 18:58:30,591 INFO mapred.FileInputFormat: Total input files to process : 0
cluster-master | 2025-04-15 18:58:31,426 INFO mapreduce.JobSubmitter: number of splits:0
cluster-master | 2025-04-15 18:58:32,002 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1744743443639_00
01
cluster-master | 2025-04-15 18:58:32,002 INFO mapreduce.JobSubmitter: Executing with tokens: []
cluster-master | 2025-04-15 18:58:32,157 INFO conf.Configuration: resource-types.xml not found
cluster-master | 2025-04-15 18:58:32,157 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
cluster-master | 2025-04-15 18:58:32,548 INFO impl.YarnClientImpl: Submitted application application_1744743443639_00
01
cluster-master | 2025-04-15 18:58:32,582 INFO mapreduce.Job: The url to track the job: http://cluster-master:8088/pro
xy/application_1744743443639_0001
cluster-master | 2025-04-15 18:58:32,583 INFO mapreduce.Job: Running job: job_1744743443639_0001
cluster-master | 2025-04-15 18:58:48,702 INFO mapreduce.Job: Job job_1744743443639_0001 running in uber mode : false
cluster-master | 2025-04-15 18:58:48,702 INFO mapreduce.Job: map 0% reduce 0%
cluster-master | 2025-04-15 18:58:45,744 INFO mapreduce.Job: map 0% reduce 100%
cluster-master | 2025-04-15 18:58:45,751 INFO mapreduce.Job: Job job_1744743443639_0001 completed successfully

```

Строка 321, столбец 1 Пробелов: 4 UTF-8 LF Python Макет: US

```

cluster-master | Document Frequency pipeline
cluster-master | Deleted /tmp/document_frequencies
cluster-master | packageJobJar: [/tmp/hadoop-unjar6350836768422686776/] [] /tmp/streamjob4663337580664821790.jar tmpD
ir=null
cluster-master | 2025-04-15 18:58:48,964 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager
r at cluster-master/172.18.0.4:8082
cluster-master | 2025-04-15 18:58:49,187 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager
r at cluster-master/172.18.0.4:8082
cluster-master | 2025-04-15 18:58:49,293 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/
hadoop-yarn/staging/root/_staging/job_1744743443639_0002
cluster-master | 2025-04-15 18:58:49,514 INFO mapred.FileInputFormat: Total input files to process : 1
cluster-master | 2025-04-15 18:58:49,944 INFO mapreduce.JobSubmitter: number of splits:1
cluster-master | 2025-04-15 18:58:50,476 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1744743443639_00
02
cluster-master | 2025-04-15 18:58:50,706 INFO mapreduce.JobSubmitter: Executing with tokens: []
cluster-master | 2025-04-15 18:58:50,625 INFO conf.Configuration: resource-types.xml not found
cluster-master | 2025-04-15 18:58:50,625 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
cluster-master | 2025-04-15 18:58:50,675 INFO impl.YarnClientImpl: Submitted application application_1744743443639_00
02
cluster-master | 2025-04-15 18:58:50,706 INFO mapreduce.Job: The url to track the job: http://cluster-master:8088/pro
xy/application_1744743443639_0002
cluster-master | 2025-04-15 18:58:50,707 INFO mapreduce.Job: Running job: job_1744743443639_0002
cluster-master | 2025-04-15 18:58:56,792 INFO mapreduce.Job: Job job_1744743443639_0002 running in uber mode : false
cluster-master | 2025-04-15 18:58:56,793 INFO mapreduce.Job: map 0% reduce 0%
cluster-master | 2025-04-15 18:59:00,844 INFO mapreduce.Job: map 100% reduce 0%
cluster-master | 2025-04-15 18:59:03,853 INFO mapreduce.Job: map 100% reduce 100%
cluster-master | 2025-04-15 18:59:05,864 INFO mapreduce.Job: Job job_1744743443639_0002 completed successfully
cluster-master | 2025-04-15 18:59:05,929 INFO mapreduce.Job: Counters: 54
File System Counters

```

Строка 321, столбец 1 Пробелов: 4 UTF-8 LF Python Макет: US

The screenshot shows a GitHub Codespace interface with a terminal window open. The terminal output indicates that an indexing operation has completed successfully:

```

flush of system_schema.tables, Reason: INTERNALLY_FORCED, Usage: 7868 bytes
cassandra-server | INFO [PerDiskMemtableFlushWriter_0_4] 28 Indexing completed successfully.
36854775808), max(9223372036854775807)
cassandra-server | INFO [PerDiskMemtableFlushWriter_0_4] 2025-04-15 18:59:08,330 Flushing.java:179 - Completed flushin
g /var/lib/cassandra/data/system_schema/tables-afddfb9dc1e30688056eed6c302be09/nb-10-big-Data.db (441B) for commitlog p
osition CommitLogPosition(segmentId=1744743429150, position=123832)
cassandra-server | INFO [Native-Transport-Requests-1] 2025-04-15 18:59:08,355 ColumnFamilyStore.java:1052 - Enqueuing
flush of system schema.keyspaces, Reason: INTERNALLY_FORCED, Usage: 7298 (0%) on-heap, 0B (0%) off-heap
cassandra-server | INFO [PerDiskMemtableFlushWriter_0_3] 2025-04-15 18:59:08,365 Flushing.java:153 - Writing Memtable-
keyspaces@411115928 (157B) serialized bytes, 1 ops, 7298 (0%) on-heap, 0B (0%) off-heap, flushed range = [min(-9223372036
854775808), max(9223372036854775807)]
cassandra-server | INFO [PerDiskMemtableFlushWriter_0_3] 2025-04-15 18:59:08,366 Flushing.java:179 - Completed flushin
g /var/lib/cassandra/data/system_schema/keyspaces-abac5682deab31c5b53b3d6cffdfb5/nb-10-big-Data.db (123B) for commitl
o position CommitLogPosition(segmentId=1744743429150, position=123832)
cassandra-server | INFO [Native-Transport-Requests-1] 2025-04-15 18:59:08,416 Keyspace.java:379 - Creating replication
strategy search_engine params: KeyspaceParams{durable_writes=true, replication=ReplicationParams{class=org.apache.cassan
dra.locator.SimpleStrategy, replication_factor=1}}
cassandra-server | INFO [Native-Transport-Requests-1] 2025-04-15 18:59:08,418 ColumnFamilyStore.java:499 - Initializin
g search_engine.vocabulary
cluster-master | Schema created successfully
cassandra-server | INFO [Native-Transport-Requests-1] 2025-04-15 18:59:08,928 QueryProcessor.java:654 - Fully upgraded
to at least 5.0.4
cluster-master | Vocabulary loaded successfully
cluster-master | Term index loaded successfully
cluster-master | Indexing completed successfully
cluster-master | Done indexer
cluster-master | Data science query running
cluster-master | This script will include commands to search for documents given the query using Spark RDD

```

After indexing completes, the system automatically runs a sample search for "data science". Here is the results screenshot:

The screenshot shows a GitHub Codespace interface with a terminal window open. The terminal output shows the execution of an app.py script which performs a search for "data science" using a YARN application:

```

$ ./app.py
def load_vocabularv():
    after waiting maxRegisteredResourcesWaitingTime: 300000000
    Applying BM25...
    25/04/15 18:59:54 INFO YarnSchedulerBackend$YarnSchedulerEndpoint: ApplicationMaster registered as N
ettyRpcEndpointRef(spark-client://YarnAM)
    cluster-master | Connected to cluster for BM25
    cassandra-server | WARN [Native-Transport-Requests-2] 2025-04-15 18:59:54,779 SelectStatement.java:557 - Aggregation q
    uery used without partition key
    cassandra-server | WARN [Native-Transport-Requests-2] 2025-04-15 18:59:54,785 NoSpamLogger.java:107 - Aggregation quer
    y used without partition key on table search_engine.document_stats, aggregation type: AGGREGATE_EVERYTHING
    cluster-master | avg_d1_row done Row{avg=0}
    cassandra-server | WARN [Native-Transport-Requests-1] 2025-04-15 18:59:54,805 SelectStatement.java:557 - Aggregation q
    uery used without partition key
    cluster-master | total_docs_row done 0
    cluster-master | query_terms done defaultdict(<class 'float'>, {})
    cluster-master | query_term_data
    df_row done 0
    idf done 0.69314718805599453
    term_rows done <cassandra.cluster.ResultSet object at 0x772eb41ae700>
    query_term_science
    df_row done 0
    idf done 0.69314718805599453
    term_rows done <cassandra.cluster.ResultSet object at 0x772eb41b5670>
    top_docs []
Top 10 relevant documents (BM25): []

```

The complete system shows how different big data technologies can work together. Hadoop handles the large-scale processing, Cassandra stores the index efficiently, and Spark provides fast query responses. While simple, this implementation demonstrates the core ideas behind real search engines.

Challenges

The most challenging part was making sure all components work together correctly. For example, issue was Cassandra connections timing out. I solved this by adding proper waiting checks and proper ports for Cassandra before loading data. The main issue is that due to some constraints of my hardware I had to use GitHub Codespaces to use Docker. This has lead to unpredictable behavior regarding temporary files managing. Due to this the ranker returns empty set of retrieved documents. However, the algorithms works correctly if used on a test example.

