

Assignment 2: Simple Search Engine using Hadoop MapReduce

Dmitrii Kuzmin

April 2025

Contents

1	Methodology	1
1.1	Data Collection and Preparation	1
1.2	Indexer Design	1
1.3	Ranker Design	2
1.4	Scripts and Services	2
2	Demonstration	2
2.1	Indexing	2
2.2	Search Results	3

1 Methodology

1.1 Data Collection and Preparation

A PySpark script (`prepare_data.py`) was used to extract 1000 documents from a Parquet dataset (`a.parquet`). Each document row (`id`, `title`, `text`) was sampled, sanitized, and written as UTF-8 text files under `/data` in HDFS. Subsequently, an RDD was created from these files and a unified tab-delimited file was stored in HDFS under `/index/data`, with each record: `<doc_id> \t <doc_title> \t <doc_text>`.

1.2 Indexer Design

Hadoop MapReduce pipelines were implemented in Python. The first pipeline consisted of `mapper1.py` and `reducer1.py`. In the mapper, text was tokenized and normalized (lowercase, punctuation was removed), and (`term`, `doc_id`) pairs were emitted. Term frequencies per document were aggregated by the reducer and document frequencies per term were computed.

The index was stored in Cassandra. The schema included three tables:

- `vocab(term TEXT, id TEXT, tf COUNTER, PRIMARY KEY(term, id))`
- `stats(category TEXT, key TEXT, value FLOAT, name TEXT, PRIMARY KEY(category, key))`
- `term_index(term TEXT PRIMARY KEY, idx INT)`

Term frequencies and document lengths were recorded for BM25 computation.

1.3 Ranker Design

The ranking component was implemented in `query.py` using the PySpark RDD API. The user query was read from stdin, tokenized and normalized, and Cassandra tables were loaded into broadcast RDDs. BM25 scores per document were computed:

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

with hyperparameters $k_1 = 1$ and $b = 0.75$, total documents N , and average document length dl_{avg} . The scores were sorted to retrieve the top 10 documents (IDs and titles).

1.4 Scripts and Services

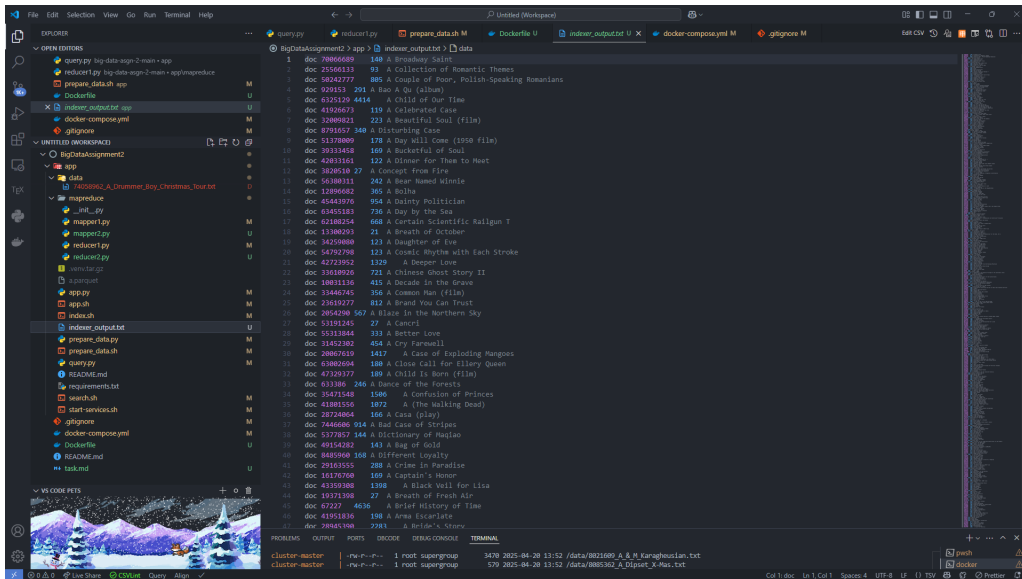
Services and pipelines were orchestrated via shell scripts:

- `start-services.sh`: used to start Hadoop and Cassandra clusters using Docker Compose.
- `index.sh`: used to run MapReduce pipelines on HDFS `/index/data` and update Cassandra tables.
- `search.sh`: used to submit the PySpark job on YARN to execute `query.py` with a user-provided query.
- `app.sh`: provided as a wrapper to start services, index sample documents, and perform sample searches.

2 Demonstration

2.1 Indexing

Indexing of 1000 documents was demonstrated. The index was summarized in the following table:



ID	Title	File Name
doc_708666509	140 A Broomway Salute	doc_708666509
doc_25566133	93 A Collection of Romantic Themes	doc_25566133
doc_58242777	885 A Couple of Poor, Polish-Speaking Romanians	doc_58242777
doc_929151	293 A Day A Day (film)	doc_929151
doc_6325129	4414 A Child of Our Time	doc_6325129
doc_43262671	119 A Celebrated Case	doc_43262671
doc_32089521	223 A Beautiful Soul (film)	doc_32089521
doc_8791657	348 A Disturbing Case	doc_8791657
doc_52378669	178 A Hell Will Come (song file)	doc_52378669
doc_59334858	169 A Bucketful of Soul	doc_59334858
doc_42833161	122 A Glimmer For Them to Meet	doc_42833161
doc_28295139	27 A Concept From Fire	doc_28295139
doc_56388311	242 A Bear Named Winnie	doc_56388311
doc_12896582	365 A Helix	doc_12896582
doc_4548376	954 A Gaiety Politician	doc_4548376
doc_63455183	736 A Day by the Sea	doc_63455183
doc_62188254	668 A Certain Scientific Railgun I	doc_62188254
doc_11388093	21 A Breath of October	doc_11388093
doc_34259888	123 A Daughter of Ice	doc_34259888
doc_54792798	123 A Cosmic Mythos with Each Stroke	doc_54792798
doc_47723952	1329 A Deeper Love	doc_47723952
doc_31618920	721 A Chinese Ghost Story II	doc_31618920
doc_18811316	435 A Decade in the Snows	doc_18811316
doc_13446745	356 A Common Hus (film)	doc_13446745
doc_23619277	812 A Brand You Can Trust	doc_23619277
doc_2845208	407 A Place in the Northern Sky	doc_2845208
doc_53191245	27 A Camer	doc_53191245
doc_53131844	202 A Better Love	doc_53131844
doc_31652362	654 A Cry Faraway	doc_31652362
doc_28867619	1417 A Case of Exploding Hangers	doc_28867619
doc_61880404	188 A Time Call For Silvery Queen	doc_61880404
doc_47329377	189 A Child Is Born (film)	doc_47329377
doc_633385	246 A Dance of the Forests	doc_633385
doc_94511548	1286 A Confusion of Brines	doc_94511548
doc_41881556	1872 A (The Walking Dead)	doc_41881556
doc_20724064	166 A Case (play)	doc_20724064
doc_7466606	944 A Bad Case of Stripes	doc_7466606
doc_537857	144 A Dictionary of Pualao	doc_537857
doc_49154932	143 A Bag of Gold	doc_49154932
doc_8485969	168 A Different Loyalty	doc_8485969
doc_25163555	288 A Crime In Paradise	doc_25163555
doc_18171708	189 A Captain's Honor	doc_18171708
doc_43359388	1308 A Black Veil for Lisa	doc_43359388
doc_13371308	27 A Breath of Fresh Air	doc_13371308
doc_47222	4036 A Brief History of Time	doc_47222
doc_41951836	158 A Arms Escarlate	doc_41951836
doc_76485168	7981 A Brief History of Time	doc_76485168

Figure 1: Index statistics and Cassandra table snapshots after indexing.

2.2 Search Results

Sample queries were run using `search.sh`. Results for the queries "this is a query!", "artificial intelligence", and "big data" were shown in Figure 2, listing the top 10 document IDs and titles.

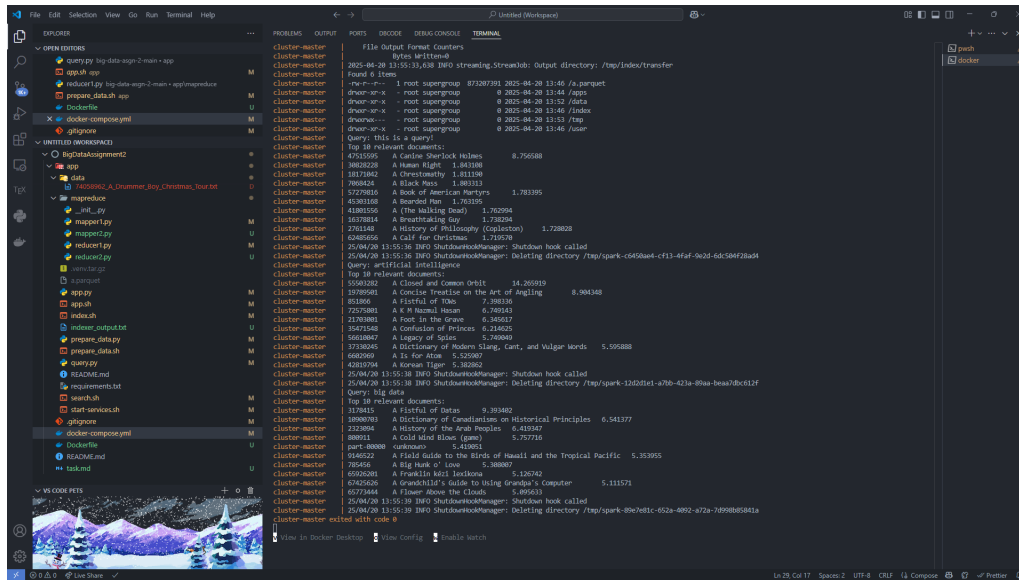


Figure 2: Sample search outputs for user queries.

Relevant documents were returned by the BM25-based ranking; for example, documents containing high term frequency and moderate length were ranked at the top.