Report for Assignment №1. MapReduce. Simple Search Engine

Team and Contacts

- Abdurasul Rakhimov
    - Email: a.rahimov@innopolis.ru
    - Telegram: @C230BL
- Svyatoslav Semenyuk
    - Email: s.prometeev@innopolis.ru
    - Telegram: @Prometheus3375
- Temur Kholmatov
    - Email: t.holmatov@innopolis.ru
    - Telegram: @temur_kholmatov

Github Repository

- Link: https://github.com/temur-kh/big-data-homework-01

Contents

Motivation

The main goal of this assignment is to have a closer look at a Hadoop cluster and the MapReduce framework as well as explore the practical usage of it for complex distributed computations. The task is to implement a simple search engine with two main parts: an indexing engine and a ranker engine. Usually, searching engines tend to operate upon a huge amount of data (pages, websites, documents, etc.). That is why an application of distributed computations is essentially important for such a problem.

Achieved Results

To begin with, we implemented the whole system and it runs successfully. Mostly, we followed all suggestions from the guide [1] regarding the algorithms and overall architecture. In the Appendix there is an instruction on how to run the Indexer and Query operations. The same instruction you could also find in our repository. Here are some points worth to mention:

- **Architecture of the System** is very similar to what was presented in the guide (Fig. 1).

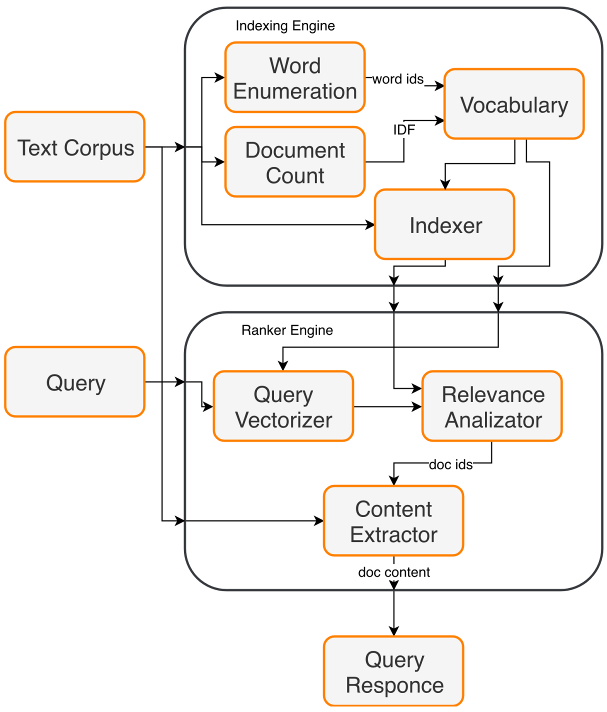

Figure 1. The Search Engine Architecture

However, we would have edited this picture. Firstly, there was an important issue of reading a text corpus: we had to be able to read and parse a set of input files containing JSONs to get two structure formats (`docId -> docText` for Indexing Engine and `docId -> (docURL, docTitle)` for Ranker Engine). So, we implemented one more module called CorpusParser which uses MapReduce jobs to parse the corpus into two different outcomes. Secondly, we really did not encounter neccessity of a Vocabulary module so we directly passed outputs from the Word Enumaration and the Document Count to the Indexer. Thirdly, we almost joined the Query Vectorizer, the Relevance Analizator and the Content Extractor into one module because two of them had a basic structure and implementation, whereas only one used MapReduce.

- **Relevance Function** is decided to be a simple one from the guide $$ r(q,d) = \sum_{i: i\in d, i\in q} q_i \cdot d_i.$$ The main advantage of it is its simplicity and ease of implementation compared to BM25 [2]. Indeed, there was no need to complicate the system since the main goal is to explore usage of MapReduce.
- **Text Parser**. We decided that it would be useful to implemented our own text parser instead of using a simple StringTokenizer. Though our parser is not much complicated, it is supposed to prevent some unwanted symbols and substrings to appear.
- **TextToMap and MapToText**. We encountered some issues with writing and reading of a document's TF/IDF weights. So, since we didn't manage to find any "native" solutions, we implemented our own methods to convert and decode the map data structure to/from a text.

Participants Contribution

Honestly, it would be difficult to strictly separate the parts of the system by the authors because each team member had more or less contribution to each subsystem. We often held status meetings to discuss work done and further plans as well as tent to follow best practices of git usage to make our lives less painful. Anyway, we could define main contributions of each team member just to report:

- *Abdurasul Rakhimov*: Corpus Parser, Word Enumeration modules, cluster administration

- *Svyatoslav Semenyuk*: VectorGen (also known as Indexer), Ranker (also known as Query Vectorizer and Extractor) modules, Text Parser
- *Temur Kholmatov*: Document Count, Relevance Analizer modules, report

Conclusion

In conclusion, we had good experience working with MapReduce framework on an interesting task - a search engine. During the workflow, we encountered several issues and found solutions to. Though the system we created is quite simple, now it seems to be not much difficult to upgrade it module by module.

Appendix

Usage Guide

To run Indexer use next command:

```
$hadoop jar <jar_name>.jar Indexer <path to docs on HDFS> <path to output directory on HDFS>
```

Example: `$hadoop jar IBD_HW1.jar Indexer /docs /indexed_docs`

To run Query use next command:

```
$hadoop jar <jar_name>.jar Query <path to output directory of Indexer on HDFS> <query in quotes> <number of most relevant docs>
```

Example: `$hadoop jar IBD_HW1.jar Query /indexed_docs "what is big data" 3`

External References

1. Assignment #1. MapReduce. Simple Search Engine. link: https://hackmd.io/BxoTvclHQFWAS9-2ZpABMQ?view
2. Okapi BM25 - The ranking function. link: https://en.wikipedia.org/wiki/Okapi_BM25