Lab 01 Report: A Gentle Introduction to Hadoop

Teacher in charge: Nguyễn Ngọc Thảo.

Lab Instructors	Email
Đỗ Trọng Lễ	dtle@selab.hcmus.edu.vn
Bùi Huỳnh Trung Nam	huynhtrungnam2001@gmail.com

Group Name : Left4Dead

No.	Student ID	Student Name
1	21127329	Châu Tấn Kiệt
2	21127170	Nguyễn Thế Thiện
3	21127642	Trịnh Minh Long

Abstract

Lab Progress

No.	Task	Expected output	Progress
1.1	Text Cleaning and Term Frequency	Students can count terms with MapReduce and output an MTX file	100%
1.2	Low-Frequency Term Elimination	Students can read MTX file and perform basic filtering	100%
1.3	Top 10 Most Frequent Words	Students can read MTX file and perform basic filtering	100%
1.4	TF-IDF	Students can read MTX file and perform advanced calculations with MapReduce	100%
1.5	Highest average TF-IDF	Students can perform advanced calculations with MapReduce	0%
2.1	K-Means on 2D Data	Students can apply iterative algorithms into MapReduce	100%
2.2	K-Means on Preprocessed Data	Students can combine the applications of iterative algorithms and advanced calculations into MapReduce	0%
2.3	Scalable K-Means++ Initialization	Students can combine the applications of iterative algorithms and advanced calculations into MapReduce	0%

1. Data preprocessing

1.1. Text Cleaning and Term Frequency

- Data description:
  - Multiple folders containing lots of input file in ".txt" format.
  - stopwords.txt file.
- Execution: `bin/hadoop App.jar App <input/path> <output/path> -skip <path/to/stopwords/file>`
- Job implementation:
  1. Number of mapper: 1
  2. Number of reducer: 1
  3. Tasks:
  - Mapper:
    - It takes three parameters: Object key, Text value, and Context context. The key represents the input key, value represents the input value (usually a line of text), and context is used to emit output from the mapper.
    - First the input text "value" is tonkenized using StringTokenizer library
    - Then, we get the path of the input (path to the input file), then extract the name of the folder.
    - Check if the name of the current file is "stopwords.txt", ignore all next processes if it true.
    - Emits key-value pairs for each word, where the key includes the word and the folder name (Ex: "hello-economy"). This will be used in reducer to avoid counting duplicate word but is in another folder.
    - Write key-value pairs to context.
  - Reducer:
    - Takes three parameters: key, values, and context.
    - Iterates through the values, which are IntWritable objects.
    - Sums up the values to calculate the total count for each key.
    - Emits key-value pairs where the key look this this "hello economy" (hello+"\t"+economy), and the value is the total count.
    - Write key-value pairs to context.
  - Final tasks:
    - Then, write the values in context to output folder named "task_1_1-r-00000".
  4. Results:

- Run command:

```
tml_21127642@  > ⟨dragonfarm ~/hadoop/hadoop-3.3.6⟩ bin/hadoop jar App.jar App /input /output -skip /input/stopwords.txt
Mar 31, 2024 3:56:02 PM org.jline.utils.Log logr
INFO: Added file to cache: /input/stopwords.txt
2024-03-31 15:56:03,080 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-03-31 15:56:03,355 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Too
 interface and execute your application with ToolRunner to remedy this.
2024-03-31 15:56:03,375 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/dragonf
rm/.staging/job_1711867279535_0024
2024-03-31 15:56:03,769 INFO input.FileInputFormat: Total input files to process : 2226
2024-03-31 15:56:03,876 INFO mapreduce.JobSubmitter: number of splits:2226
2024-03-31 15:56:03,960 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1711867279535_0024
2024-03-31 15:56:03,960 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-03-31 15:56:04,086 INFO conf.Configuration: resource-types.xml not found
2024-03-31 15:56:04,087 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-03-31 15:56:04,134 INFO impl.YarnClientImpl: Submitted application application_1711867279535_0024
2024-03-31 15:56:04,163 INFO mapreduce.Job: The url to track the job: http://DESKTOP-DLGRKJK.:8088/proxy/application_1711867279
35_0024/
2024-03-31 15:56:04,163 INFO mapreduce.Job: Running job: job_1711867279535_0024
2024-03-31 15:56:09,245 INFO mapreduce.Job: Job job_1711867279535_0024 running in uber mode : false
2024-03-31 15:56:09,246 INFO mapreduce.Job:  map 0% reduce 0%
2024-03-31 15:56:25,541 INFO mapreduce.Job:  map 1% reduce 0%
2024-03-31 15:56:47,749 INFO mapreduce.Job:  map 2% reduce 0%
2024-03-31 15:57:08,027 INFO mapreduce.Job:  map 3% reduce 0%
2024-03-31 15:57:24,210 INFO mapreduce.Job:  map 4% reduce 0%
2024-03-31 15:57:45,391 INFO mapreduce.Job:  map 5% reduce 0%
2024-03-31 15:58:00,502 INFO mapreduce.Job:  map 6% reduce 0%
2024-03-31 15:58:20,702 INFO mapreduce.Job:  map 7% reduce 0%
2024-03-31 15:58:35,795 INFO mapreduce.Job:  map 8% reduce 0%
2024-03-31 15:58:55,950 INFO mapreduce.Job:  map 9% reduce 0%
2024-03-31 15:59:12,054 INFO mapreduce.Job:  map 10% reduce 0%
2024-03-31 15:59:31,219 INFO mapreduce.Job:  map 11% reduce 0%
2024-03-31 15:59:47,413 INFO mapreduce.Job:  map 12% reduce 0%
2024-03-31 16:00:07,544 INFO mapreduce.Job:  map 13% reduce 0%
2024-03-31 16:00:24,641 INFO mapreduce.Job:  map 14% reduce 0%
2024-03-31 16:00:44,774 INFO mapreduce.Job:  map 15% reduce 0%
2024-03-31 16:01:04,924 INFO mapreduce.Job:  map 16% reduce 0%
2024-03-31 16:01:21,019 INFO mapreduce.Job:  map 17% reduce 0%
2024-03-31 16:01:40,123 INFO mapreduce.Job:  map 18% reduce 0%
2024-03-31 16:01:43,145 INFO mapreduce.Job:  map 18% reduce 6%
```

- Job completed:

```
2024-03-31 16:26:22,329 INFO mapreduce.Job:  map 99% reduce 33%
2024-03-31 16:26:40,390 INFO mapreduce.Job:  map 100% reduce 33%
2024-03-31 16:26:51,423 INFO mapreduce.Job:  map 100% reduce 100%
2024-03-31 16:26:51,436 INFO mapreduce.Job: Job job_1711867279535_0024 completed successfully
2024-03-31 16:26:51,527 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=12678816
                FILE: Number of bytes written=642313397
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=5292308
                HDFS: Number of bytes written=5691204
                HDFS: Number of read operations=6683
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=2226
                Launched reduce tasks=1
                Data-local map tasks=2226
                Total time spent by all maps in occupied slots (ms)=7500230
                Total time spent by all reduces in occupied slots (ms)=1522705
                Total time spent by all map tasks (ms)=7500230
                Total time spent by all reduce tasks (ms)=1522705
                Total vcore-milliseconds taken by all map tasks=7500230
                Total vcore-milliseconds taken by all reduce tasks=1522705
                Total megabyte-milliseconds taken by all map tasks=7680235520
                Total megabyte-milliseconds taken by all reduce tasks=1559249920
        Map-Reduce Framework
                Map input records=23536
                Map output records=640157
                Map output bytes=11398496
```

- Extra information:

```
                Total time spent by all maps in occupied slots (ms)=7500230
                Total time spent by all reduces in occupied slots (ms)=1522705
                Total time spent by all map tasks (ms)=7500230
                Total time spent by all reduce tasks (ms)=1522705
                Total vcore-milliseconds taken by all map tasks=7500230
                Total vcore-milliseconds taken by all reduce tasks=1522705
                Total megabyte-milliseconds taken by all map tasks=7680235520
                Total megabyte-milliseconds taken by all reduce tasks=1559249920
        Map-Reduce Framework
                Map input records=23536
                Map output records=640157
                Map output bytes=11398496
                Map output materialized bytes=12692166
                Input split bytes=250325
                Combine input records=0
                Combine output records=0
                Reduce input groups=333399
                Reduce shuffle bytes=12692166
                Reduce input records=640157
                Reduce output records=333399
                Spilled Records=1280314
                Shuffled Maps =2226
                Failed Shuffles=0
                Merged Map outputs=2226
                GC time elapsed (ms)=235681
                CPU time spent (ms)=990120
                Physical memory (bytes) snapshot=664030183424
                Virtual memory (bytes) snapshot=5681236492288
                Total committed heap usage (bytes)=544936034304
                Peak Map Physical memory (bytes)=572395520
                Peak Map Virtual memory (bytes)=2562400256
                Peak Reduce Physical memory (bytes)=387162112
                Peak Reduce Virtual memory (bytes)=2569035776
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=5041983
        File Output Format Counters
                Bytes Written=5691204
tml_21127642@  > (dragonfarm) ~/hadoop/hadoop-3.3.6)
```

- Output files:

```
tml_21127642@  > (dragonfarm) ~/hadoop/hadoop-3.3.6) hdfs dfs -ls /output
Found 2 items
-rw-r--r--   1 dragonfarm supergroup          0 2024-03-31 16:26 /output/_SUCCESS
-rw-r--r--   1 dragonfarm supergroup    5691204 2024-03-31 16:26 /output/task_1_1-r-00000
tml_21127642@  > (dragonfarm) ~/hadoop/hadoop-3.3.6)
```

5. Challenges faced:

> - Hard to implement the code for getting input path
> - Can't find any source for write the ouput to custom file. For example: "task_1_1.mtx" instead of "task_1_1-r-00000". (a lots of methods are from old version and won't work with my current hadoop version: 3.3.6)

1.2. Low-Frequency Term Elimination

1. Tasks:

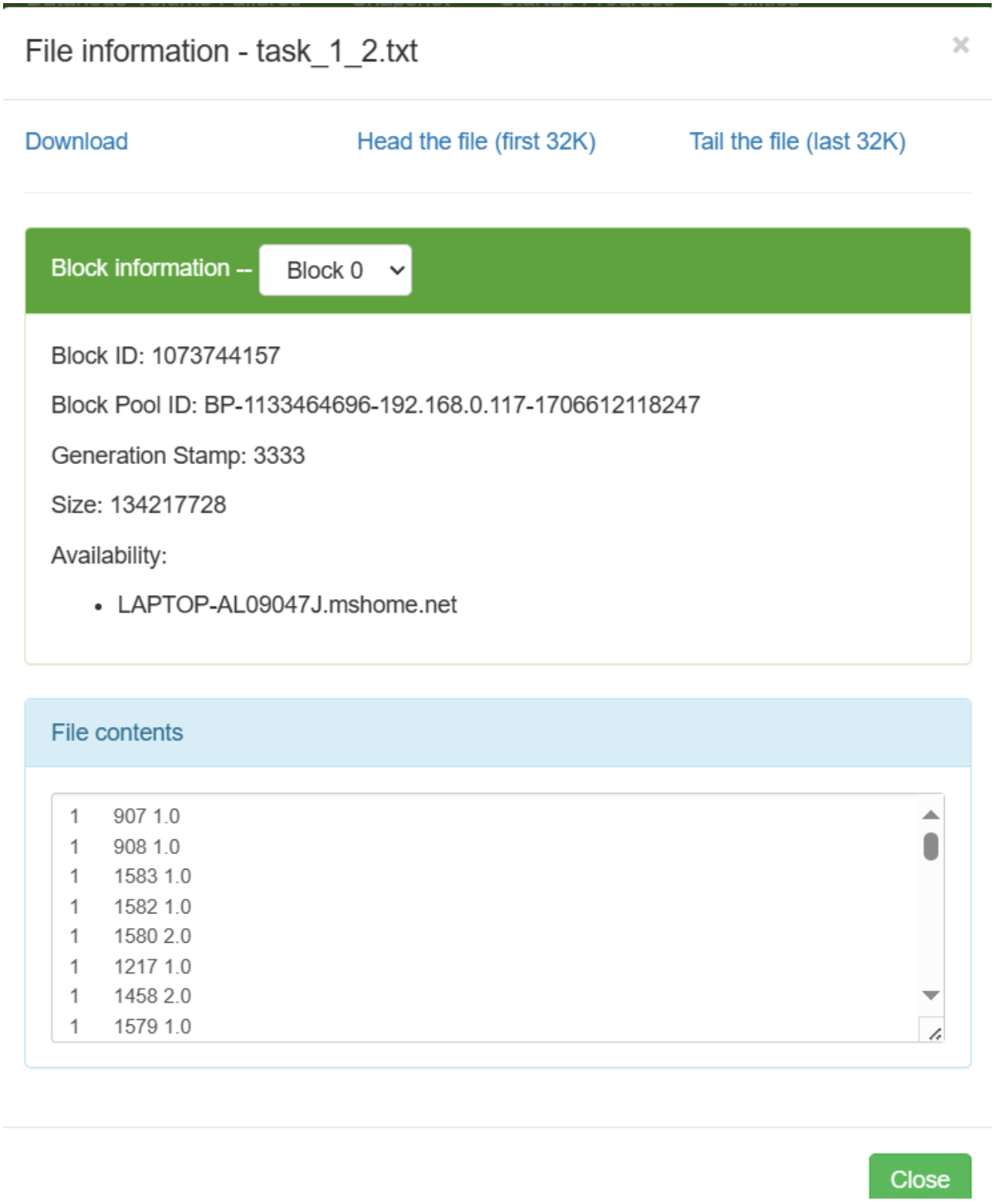
- Mapper:
  - Read the input file

    - Use Stringtokenizer to read each line
    - Set the termID as key, docID and frequency as value
- Reducer:
    - Split the value to docID and frequency and put them into a Map
    - Count the frequency of each term with the frequency
    - If the frequency of the word is > 3, print out both the docID and frequency

2. Challenges

> - At first I didn;'t know how to count without interfering the docID, but I realized there's a Map data type

- Output:



1.3. Top 10 Most Frequent Words

1. Tasks:

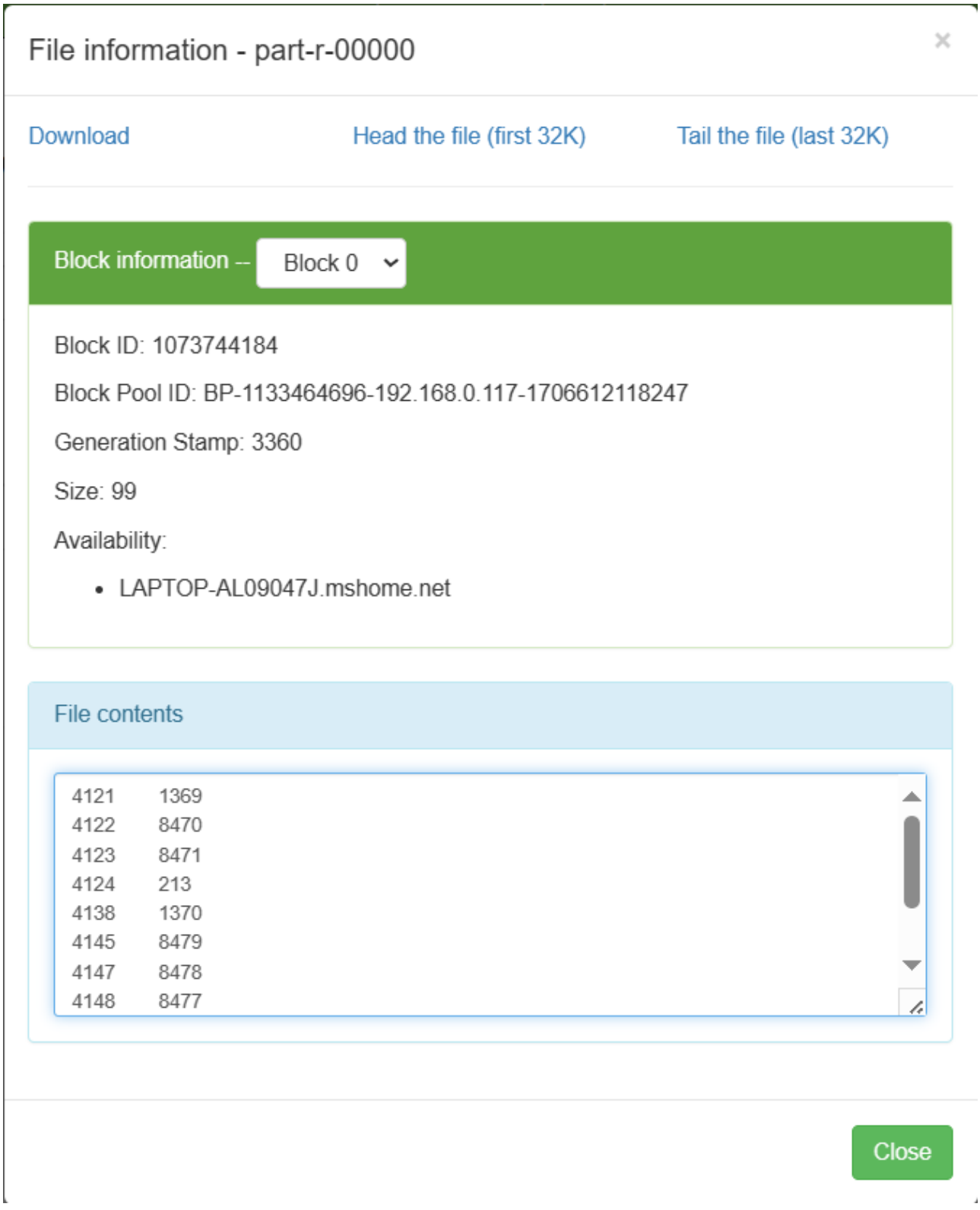
- Mapper:
  - Read the input file
  - Use Stringtokenizer to read each line
  - Set the termID as key, frequency as value
- Reducer:

- Initialize a TreeMap to store the word and its frequency
- Count the frequency of each term by adding the frequency
- Put the termID and frequency to the TreeMap
- If length of TreeMap > 10, we remove its first key, or the smallest component
- Loop until 10 to show the top 10 most frequent terms

2. Challenges

> - It's challenging to find the way to rank the frequency

- Output:



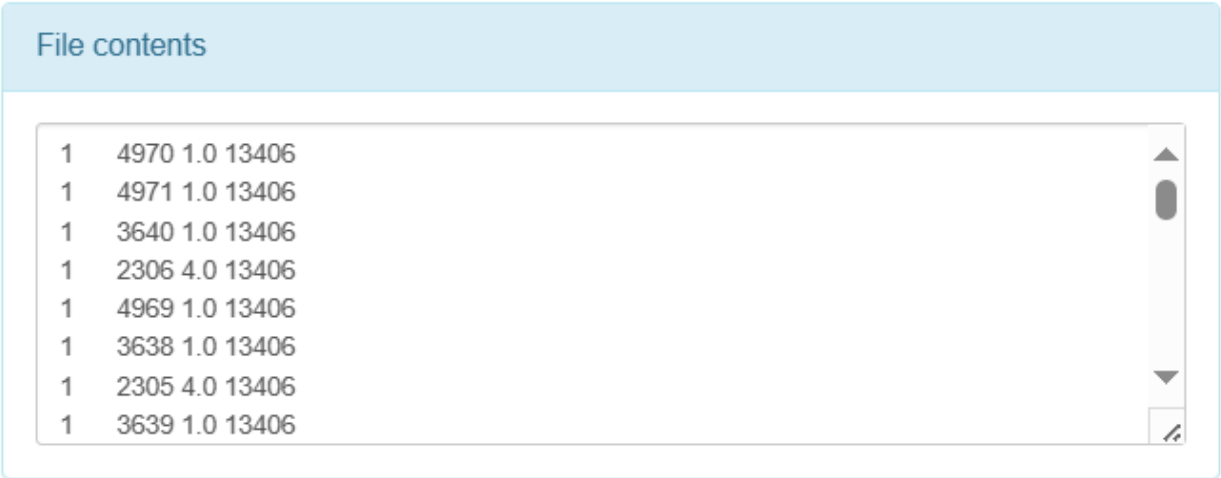
Output:

1.4. TF-IDF

In this section, there are 2 Mappers and 2 Reducers

1. Tasks:

- Mapper 1:
  - Read the input file
  - Use Stringtokenizer to read each line
  - Set the docID as key, termID and frequency as value
- Reducer 1:
  - Initialize a TreeMap to store the word and its frequency
  - Count the number of words in each document
- Output:

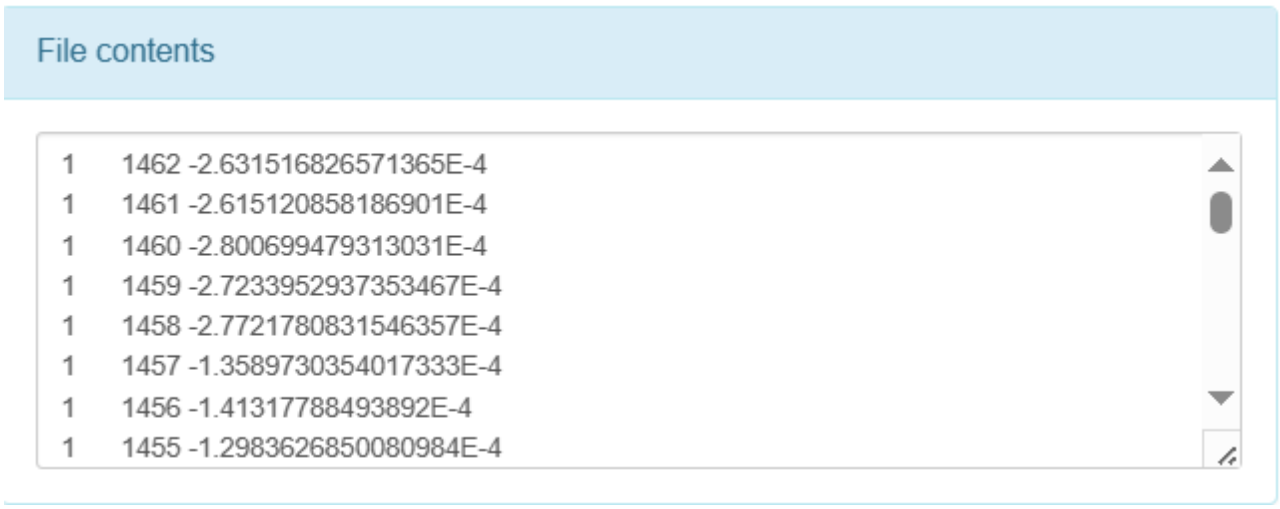


- Mapper 2:
  - Read the input file
  - Use Stringtokenizer to read each line
  - Set the docID as key, termID and frequency and docWords as value
- Reducer 2:
  - Calculate TF-IDF using the formula:
  - `tf(t,d)`: count of t in d / number of words in d
  - `idf(t)`: log(N/ df(t))
  - `tf-idf`: `tf` * `idf`

2. Challenges

> - I couldn't use 2 Mappers and Reducer and apply that into 1 output file, which means I have to create 1 text file as an intermediate file to the final output, this cost a large amount of disk usage.

- Output:



1.5. Highest average TF-IDF

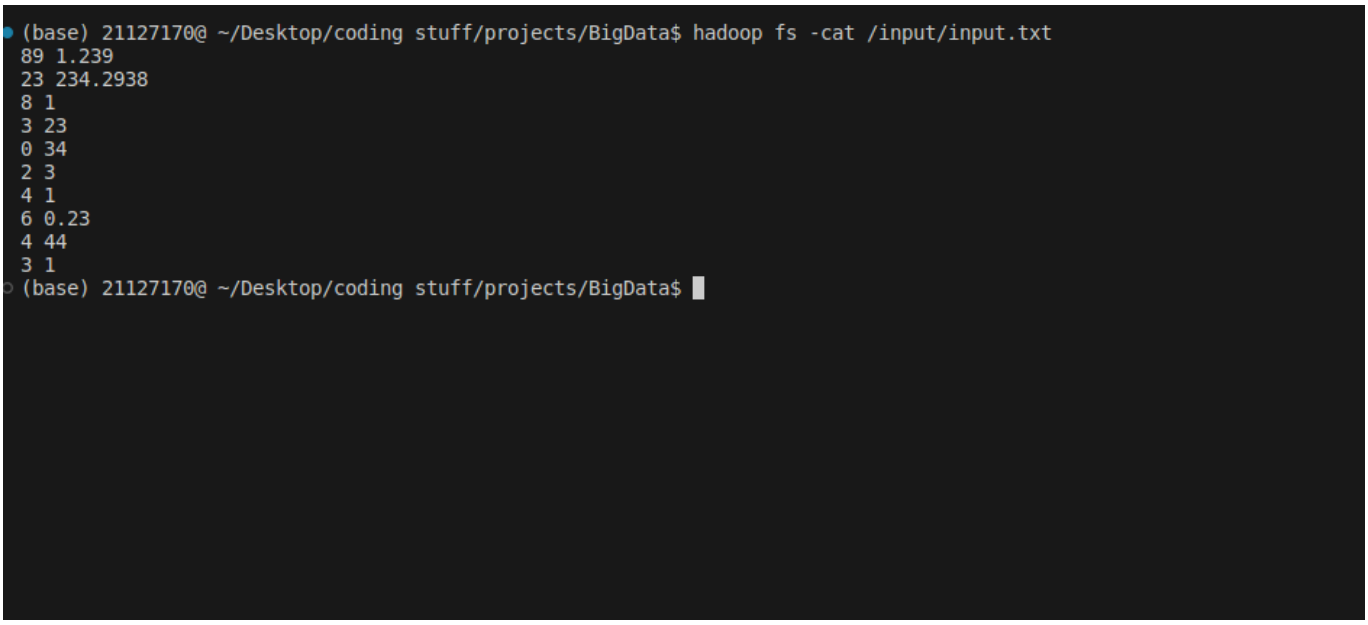
We did not accomplish this exercise.

2. K-Means Algorithm

2.1. K-Means on 2D data

Note: The code is referenced based on [1]. Credit to "seraogianluca".

Data description: Data is only one text file consists of some 2D points. Each line consists of two float numbers divided by a space indicating the coordinations of a 2D point.



For the use of a small input file, the number of mapper and reducer shall be 1 (one) for the job. The main idea is to initialize randomized unique centroids and then perform Iterative Task, either though a fixed number of times or until the centroids converges, then performs the Final Task to output the class assignments after checking and achieving the completion criterias through MapReduce, and output the final centroids through HDFS DataOutputStream.

Execution: hadoop jar kMeansMain.jar kMeansMain /input /output [num_of_clusters] [num_of_iterations]

```
(base) 21127170@ ~/Desktop/coding stuff/projects/BigData/lab02/src/2.1$ hadoop com.sun.tools.javac.Main kMeansMain
.java Point.java kMeansMapper.java kMeansReducer.java kMeansFinal.java | jar cf kMeansMain.jar kMeans*.class Point
*.class kMeansMapper*.class kMeansReducer*.class kMeansFinal*.class
(base) 21127170@ ~/Desktop/coding stuff/projects/BigData/lab02/src/2.1$ hadoop jar kMeansMain.jar kMeansMain /inpu
t /output 3 20
2024-03-31 16:59:43,789 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:
8032
2024-03-31 16:59:44,425 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Impl
ement the Tool interface and execute your application with ToolRunner to remedy this.
2024-03-31 16:59:44,473 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/st
aging/p4stwi2x/.staging/job_1711878437052_0005
2024-03-31 16:59:45,314 INFO input.FileInputFormat: Total input files to process : 1
2024-03-31 16:59:46,220 INFO mapreduce.JobSubmitter: number of splits:1
2024-03-31 16:59:46,928 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1711878437052_0005
2024-03-31 16:59:46,928 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-03-31 16:59:47,223 INFO conf.Configuration: resource-types.xml not found
Follow link (ctrl + click) ,223 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
,437 INFO impl.YarnClientImpl: Submitted application application_1711878437052_0005
2024-03-31 16:59:47,523 INFO mapreduce.Job: The url to track the job: http://p4stwi2x-Dell-System-Vostro-5460:8088
/proxy/application_1711878437052_0005/
2024-03-31 16:59:47,524 INFO mapreduce.Job: Running job: job_1711878437052_0005
```

Iterative Task:

- Mapper (kMeansMapper.java): Using the Mapper.Context class to contain centroids info, we input the datapoints from the input directory, convert them to a Point class object, and then calculate the closest centroid for each datapoint. Then we output the key-value pair of <key=centroid_index, value=datapoint> [Int, Text].
- Reducer (kMeansReducer.java): Reducer shall group the key-value pairs that represent datapoints-centroids assignment, use that to calculate the average coordinations of such datapoints of a same group, resulting in new coordinations of the centroid of the group. We output the key-value pair of <key=centroid_index, value=new_coordination> [Int, Text] to folder "\temp\iter_0\", "\temp\iter_1\", ...

```
(base) 21127170@ ~/Desktop/coding stuff/projects/BigData$ hadoop fs -cat /temp/iter_0/part-r-00000
0       89.0 1.239
1       4.6 1.246
2       7.5 83.82345
(base) 21127170@ ~/Desktop/coding stuff/projects/BigData$
```

Final Task:

- Mapper (kMeansMapper.java): Same as Mapper in Iterative Task.
- Reducer (kMeansFinal.java): We do not do anything except outputing every key-value pair of <key=centroid_index, value=new_coordination> we gain from the Mapper that already does the job of assigning points to centroids (clusters), to file "part-r-00000" in the output directory.

```
(base) 21127170@ ~/Desktop/coding stuff/projects/BigData$ hadoop fs -cat /output/part-r-00000
0       89.0 1.239
1       3.0 1.0
1       6.0 0.23
1       4.0 1.0
1       2.0 3.0
1       8.0 1.0
2       4.0 44.0
2       0.0 34.0
2       3.0 23.0
2       23.0 234.2938
(base) 21127170@ ~/Desktop/coding stuff/projects/BigData$
```

- Challenges faced:

> - **No prior experience in configuring a multi-task Hadoop job.** It was indeed difficult to think of and implement a way to pass shared variables such as the centroids between Mapper and Reducer, and between the MapReduce tasks. This took us the most days in this entire lab to figure out Context class and HDFS DataIOStream.

2.2. K-Means on Preprocessed Data

We did not accomplish this exercise.

2.3. Scalable K-Means++

We did not accomplish this exercise.

References

- [1] seraogianluca (2021), https://github.com/seraogianluca/k-means-mapreduce/tree/master, last visited: Mar 29th, 2024.
- [2] https://stackoverflow.com/questions/19012482/how-to-get-the-input-file-name-in-the-mapper-in-a-hadoop-program. Thanks Hans Brende for the workaround of getting input path