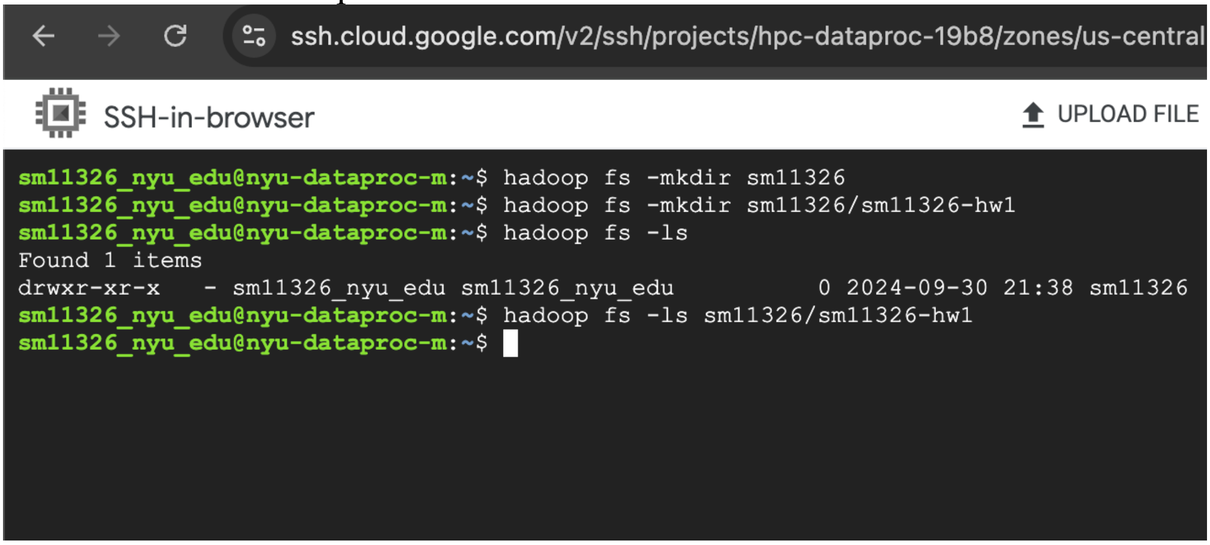
Big Data
Assignment 1 – Documentation
Sai Nishanth Mettu
sm11326

Question 1. HDFS File Manipulation (20 Points)

Part a

I have created a folder system of
Net-id/Net-id-hw1/data/
sm11326 is my root folder
sm11326/sm11326-hw1/ is my homework folder
I have used the Hadoop fs -mkdir commands

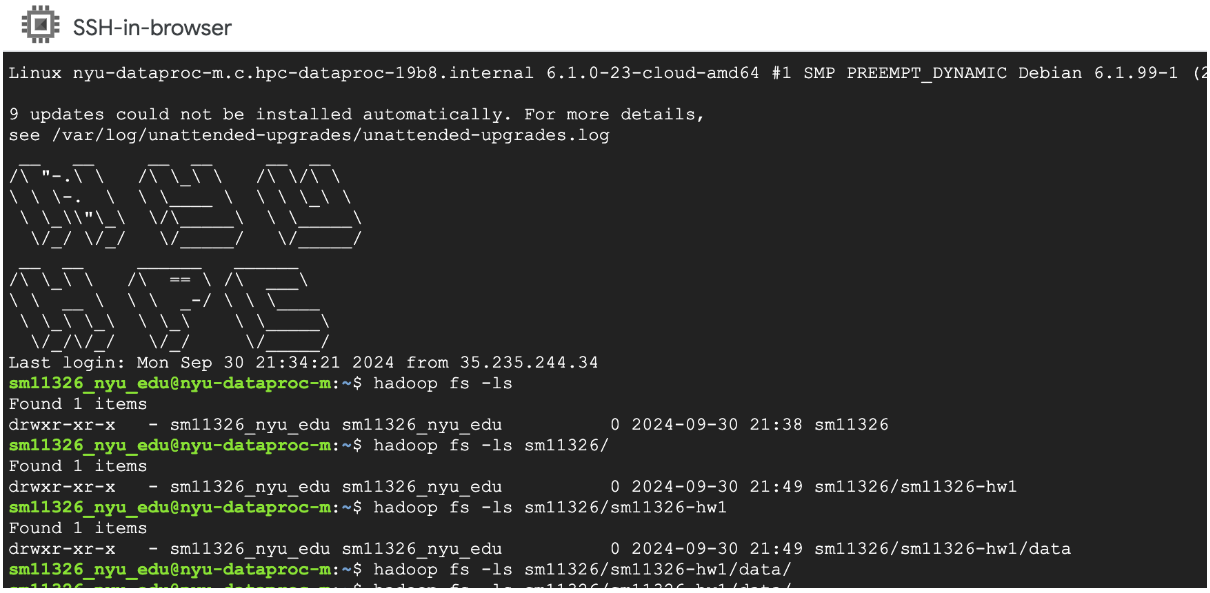

ssh.cloud.google.com/v2/ssh/projects/hpc-dataproc-19b8/zones/us-central
SSH-in-browser
UPLOAD FILE

```
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -mkdir sm11326
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -mkdir sm11326/sm11326-hw1
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -ls
Found 1 items
drwxr-xr-x   - sm11326_nyu_edu sm11326_nyu_edu          0 2024-09-30 21:38 sm11326
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -ls sm11326/sm11326-hw1
sm11326_nyu_edu@nyu-dataproc-m:~$
```

Part b

I have created a sub-directory called data within sm11326{root}/sm11326-hw1{homework-dir}/data/

I have used the command hadoop fs -mkdir sm11326/sm11326-hw1/data



SSH-in-browser

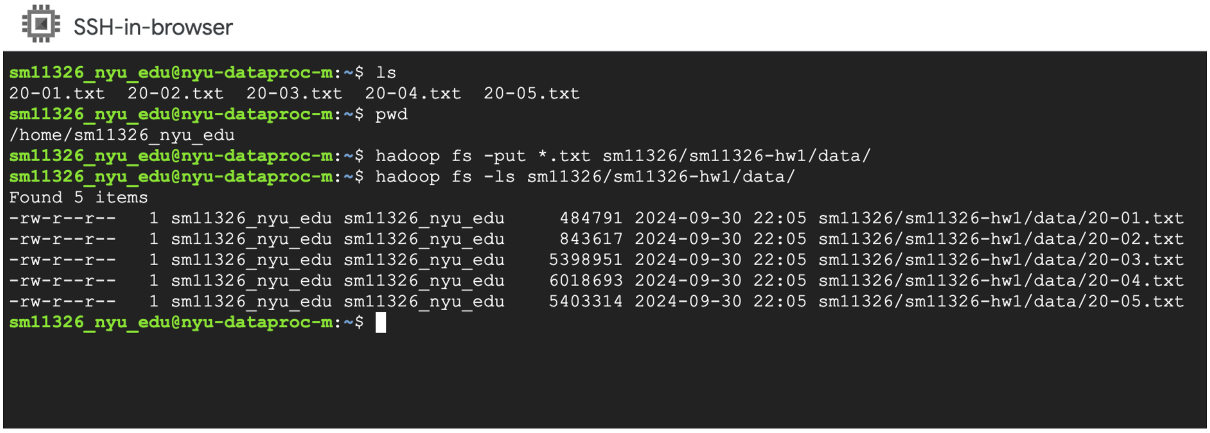
```
Linux nyu-dataproc-m.c.hpc-dataproc-19b8.internal 6.1.0-23-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.99-1 (2

9 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log
 __  __     __  __     __  __
/\ "-.\ \   /\ \_\ \   /\ \/\ \
\ \ \-.  \  \ \____ \  \ \ \_\ \
 \ \_\\"\_\  \/\_____\  \ \_____\
  \/_/ \/_/   \/_____/   \/_____/
 __  __     ______   ______
/\ \_\ \   /\  == \ /\  ___\
\ \  __ \  \ \  _-/ \ \ \____
 \ \_\ \_\  \ \_\    \ \_____\
  \/_/\/_/   \/_/     \/_____/
Last login: Mon Sep 30 21:34:21 2024 from 35.235.244.34
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -ls
Found 1 items
drwxr-xr-x   - sm11326_nyu_edu sm11326_nyu_edu          0 2024-09-30 21:38 sm11326
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -ls sm11326/
Found 1 items
drwxr-xr-x   - sm11326_nyu_edu sm11326_nyu_edu          0 2024-09-30 21:49 sm11326/sm11326-hw1
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -ls sm11326/sm11326-hw1
Found 1 items
drwxr-xr-x   - sm11326_nyu_edu sm11326_nyu_edu          0 2024-09-30 21:49 sm11326/sm11326-hw1/data
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -ls sm11326/sm11326-hw1/data/
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -ls sm11326/sm11326-hw1/data/
```

<u>Part b continued</u>

I have transferred the files into the data folder.

I used the upload button in dataproc to upload my files from my local filesystem to nyu-dataproc. Currently I am on /home/sm11326_nyu_edu as you can see from the pwd command. From there I used **<u>hadoop fs -put *.txt sm11326/sm11326-hw1/data/</u>** command to transfer the files

I did an hadoop fs -ls sm11326/sm11326-hw1/data/ to verify



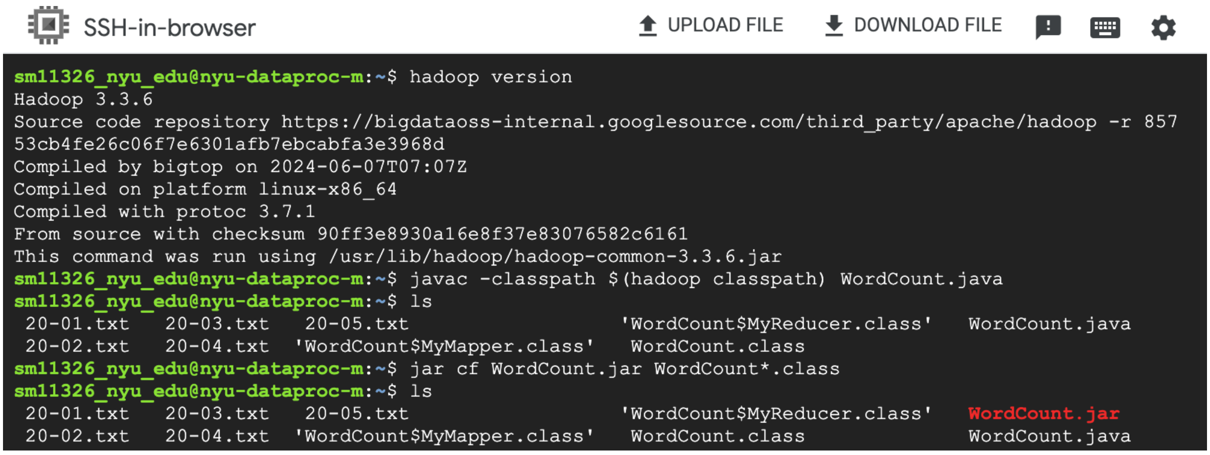
SSH-in-browser

```
sm11326_nyu_edu@nyu-dataproc-m:~$ ls
20-01.txt  20-02.txt  20-03.txt  20-04.txt  20-05.txt
sm11326_nyu_edu@nyu-dataproc-m:~$ pwd
/home/sm11326_nyu_edu
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -put *.txt sm11326/sm11326-hw1/data/
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -ls sm11326/sm11326-hw1/data/
Found 5 items
-rw-r--r--   1 sm11326_nyu_edu sm11326_nyu_edu     484791 2024-09-30 22:05 sm11326/sm11326-hw1/data/20-01.txt
-rw-r--r--   1 sm11326_nyu_edu sm11326_nyu_edu     843617 2024-09-30 22:05 sm11326/sm11326-hw1/data/20-02.txt
-rw-r--r--   1 sm11326_nyu_edu sm11326_nyu_edu    5398951 2024-09-30 22:05 sm11326/sm11326-hw1/data/20-03.txt
-rw-r--r--   1 sm11326_nyu_edu sm11326_nyu_edu    6018693 2024-09-30 22:05 sm11326/sm11326-hw1/data/20-04.txt
-rw-r--r--   1 sm11326_nyu_edu sm11326_nyu_edu    5403314 2024-09-30 22:05 sm11326/sm11326-hw1/data/20-05.txt
sm11326_nyu_edu@nyu-dataproc-m:~$
```

<u>Question 2. Word Count + TopKWords (100 Points)</u>

JOB 1 – Finding all the word counts

Step 1:- Compiling and creating a WordCount.jar



```
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop version
Hadoop 3.3.6
Source code repository https://bigdataoss-internal.googlesource.com/third_party/apache/hadoop -r 857
53cb4fe26c06f7e6301afb7ebcabfa3e3968d
Compiled by bigtop on 2024-06-07T07:07Z
Compiled on platform linux-x86_64
Compiled with protoc 3.7.1
From source with checksum 90ff3e8930a16e8f37e83076582c6161
This command was run using /usr/lib/hadoop/hadoop-common-3.3.6.jar
sm11326_nyu_edu@nyu-dataproc-m:~$ javac -classpath $(hadoop classpath) WordCount.java
sm11326_nyu_edu@nyu-dataproc-m:~$ ls
 20-01.txt   20-03.txt   20-05.txt                   'WordCount$MyReducer.class'   WordCount.java
 20-02.txt   20-04.txt  'WordCount$MyMapper.class'    WordCount.class
sm11326_nyu_edu@nyu-dataproc-m:~$ jar cf WordCount.jar WordCount*.class
sm11326_nyu_edu@nyu-dataproc-m:~$ ls
 20-01.txt   20-03.txt   20-05.txt                   'WordCount$MyReducer.class'   WordCount.jar
 20-02.txt   20-04.txt  'WordCount$MyMapper.class'    WordCount.class              WordCount.java
```

Commands

1. javac version check the Javac version (11x)
2. hadoop version (3.3.6)
3. hadoop classpath (check the path at which hadoop is installed)
4. javac -classpath $(hadoop classpath) WordCount.java
   (This is to compile the code to generate classfiles .class)
5. jar cf WordCount.jar WordCount*.class (to compile the class files into jars)
6. We have the WordCount.jar ready

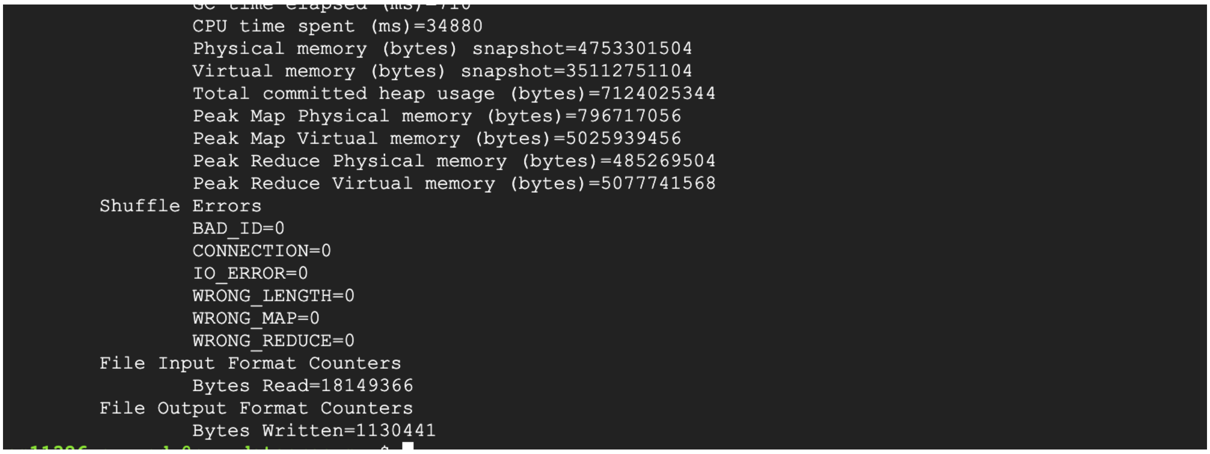
Now we use the jar and run the command

hadoop jar WordCount.jar WordCount sm11326/sm11326-hw1/data/ sm11326/sm11326-hw1/data/output

here the is the path where the files are stored inside HDFS : sm11326/sm11326-hw1/data/

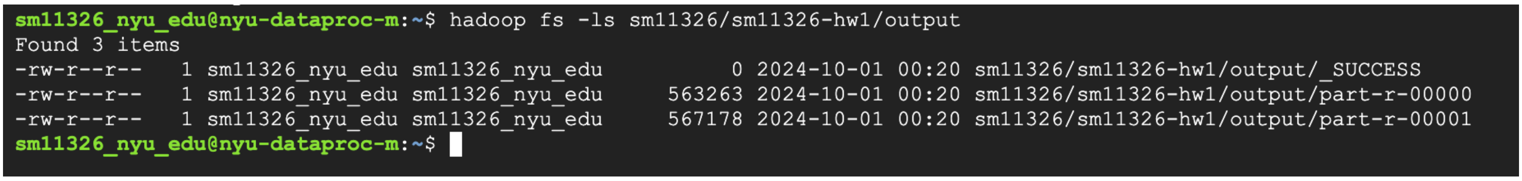
This is the directory where im storing my output : sm11326/sm11326-hw1/data/output

Once we finish running, we are checking shuffle errors.



```
		GC time elapsed (ms)=710
		CPU time spent (ms)=34880
		Physical memory (bytes) snapshot=4753301504
		Virtual memory (bytes) snapshot=35112751104
		Total committed heap usage (bytes)=7124025344
		Peak Map Physical memory (bytes)=796717056
		Peak Map Virtual memory (bytes)=5025939456
		Peak Reduce Physical memory (bytes)=485269504
		Peak Reduce Virtual memory (bytes)=5077741568
	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=18149366
	File Output Format Counters
		Bytes Written=1130441
```

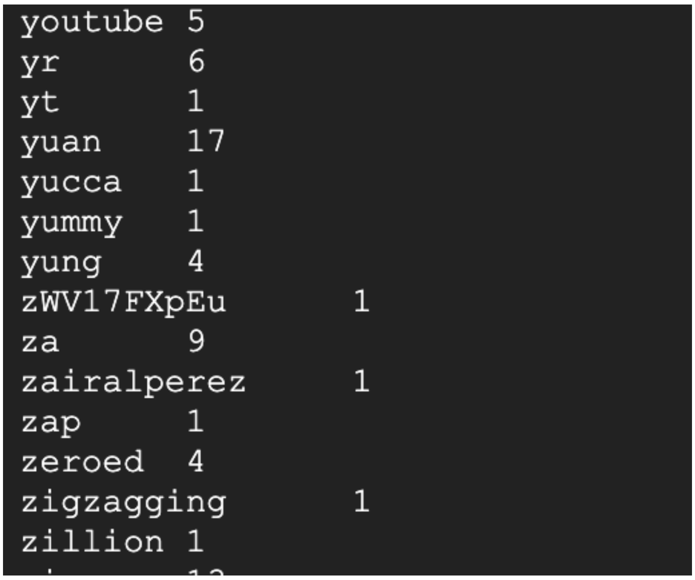
Locate the output files under
hadoop fs -ls sm11326/sm11326-hw1/data/output



```
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -ls sm11326/sm11326-hw1/output
Found 3 items
-rw-r--r--   1 sm11326_nyu_edu sm11326_nyu_edu          0 2024-10-01 00:20 sm11326/sm11326-hw1/output/_SUCCESS
-rw-r--r--   1 sm11326_nyu_edu sm11326_nyu_edu     563263 2024-10-01 00:20 sm11326/sm11326-hw1/output/part-r-00000
-rw-r--r--   1 sm11326_nyu_edu sm11326_nyu_edu     567178 2024-10-01 00:20 sm11326/sm11326-hw1/output/part-r-00001
sm11326_nyu_edu@nyu-dataproc-m:~$ 
```

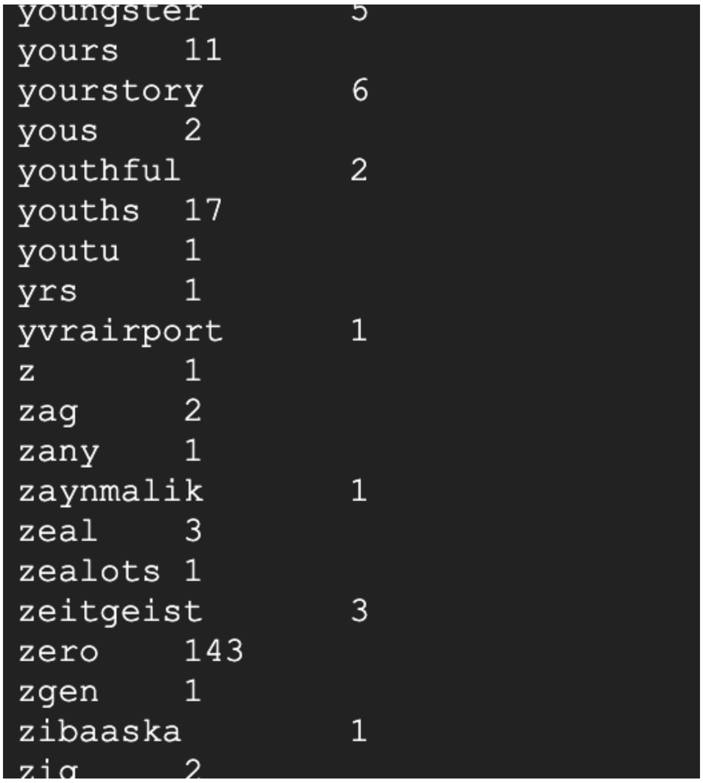
<u>We can cat out the output for both the files, to view the answer</u>

hadoop fs -cat sm11326/sm11326-hw1/data/output/part-r-00000
hadoop fs -cat sm11326/sm11326-hw1/data/output/part-r-00001
part-r-00000



```
youtube	5
yr	6
yt	1
yuan	17
yucca	1
yummy	1
yung	4
zWV17FXpEu	1
za	9
zairalperez	1
zap	1
zeroed	4
zigzagging	1
zillion	1
```

part-r-00001 output



```
youngster       5
yours   11
yourstory       6
yous    2
youthful        2
youths  17
youtu   1
yrs     1
yvrairport      1
z       1
zag     2
zany    1
zaynmalik       1
zeal    3
zealots 1
zeitgeist       3
zero    143
zgen    1
zibaaska        1
zig     2
```

Converting and storing the output into text files

```
hadoop fs -text sm11326/sm11326-hw1/output/part-r-00001 | sort > output2.txt
hadoop fs -text sm11326/sm11326-hw1/output/part-r-00000 | sort > output1.txt
```

Now we have the word count stored in text files. (JOB1 :- WORD COUNT OUTPUTS)

JOB 2 :- Finding Top K Words

<u>Now to find the Top K Words.</u>
Logic :- I am trying to use the bucketing system (A custom data structure I created using a TreeMap) as explained by the professor in the class
I have internally a WordCountMapper, a TopKReducer, a Global Reducer for the buckets and the Global Mapper.

Job 1:- The WordCount Mapper + TopKReducer will intake the output of the WordCount job as inputs and will produce an intermediate output at sm11326/sm11326-hw1/data/intermediate.

Using TreeMap:

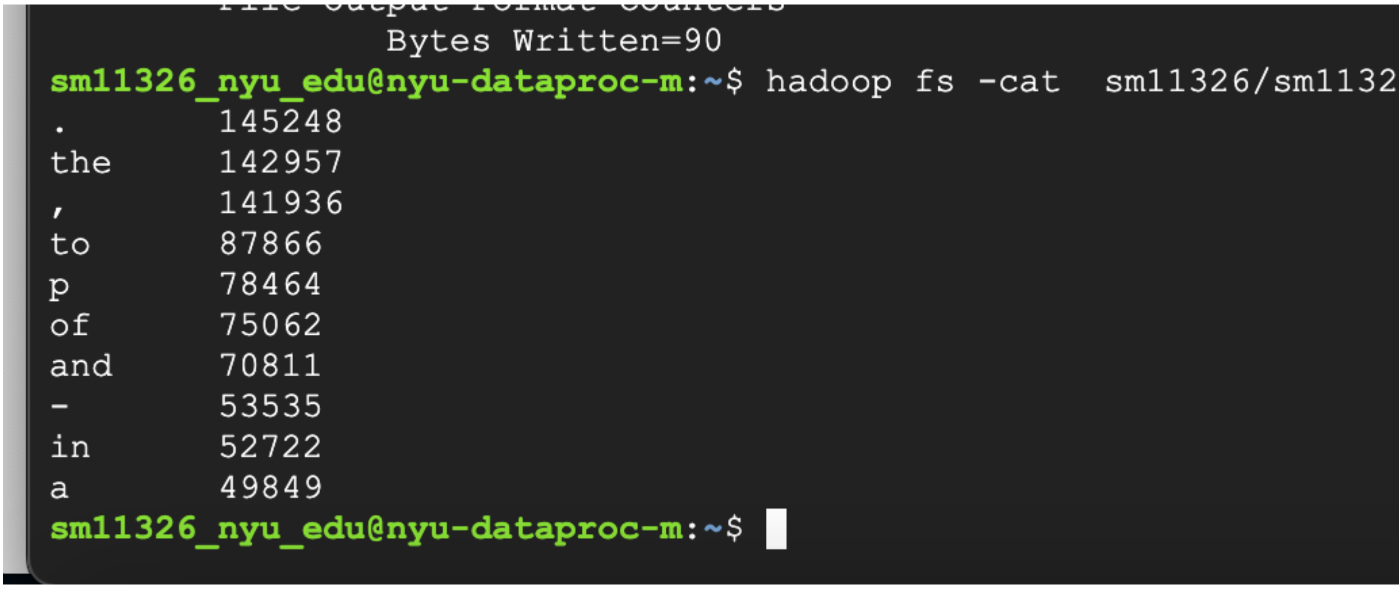
- The TreeMap was set up to maintain a sorted order of word counts. This allows you to efficiently manage the top K entries by always having the lowest count at the end.
- When the size of the TreeMap exceeded K, the entry with the lowest count was removed, maintaining only the top K words.
- private TreeMap<Integer, List<String>> topKMap = new TreeMap<>(Collections.reverseOrder());

Now the intermediate output will be fed to the Global Mapper & Reducer and this chaining will result in the global top-k words

through chaining, it will trigger another map-reduce job internally which will keep a k-size bucket and will do compare-swap operations in memory, so that the memory will never exceed k !
if (topKMap.size() > K) { topKMap.remove(topKMap.lastKey()); }

- We don't need explicit Sorting here, or swapping as the TreeMap Data Structure maintains its elements in a sorted order based on the keys. This means that whenever you add a new key-value pair, it automatically places it in the correct position in the tree and because TreeMap keeps the keys sorted as you insert them, you don't need to perform an explicit sort operation after each insertion. It manages ordering internally.
- Again we don't need to swap and compare also, just removal of the last key is sufficient as the data structure internally handles that.

Output :-



```
        File Output Format Counters
                Bytes Written=90
sm11326_nyu_edu@nyu-dataproc-m:~$ hadoop fs -cat  sm11326/sm1132
.       145248
the     142957
,       141936
to      87866
p       78464
of      75062
and     70811
-       53535
in      52722
a       49849
sm11326_nyu_edu@nyu-dataproc-m:~$
```

REGEX LOGIC

```
public static class MyMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {
    private final Text word = new Text();
    private final static IntWritable one = new IntWritable(1);
    private static final Pattern pattern = Pattern.compile("[A-Za-z0-9]+|\\.|\\-
|\\[|\\]|\\(|\\)|,");
    public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
        Matcher matcher = pattern.matcher(value.toString());
        while (matcher.find()) {
            word.set(matcher.group());
            context.write(word, one);
        }
    }
}
```

I am using this REGEX – This will fetch the alphanumeric part and the [, . ( ) [ ] -] separately…i.e www.xyz.com-total will become www, xyz, com, - , total after parsing.

This is an important step in WordCount, and we can do a lot of variations here, Lexical mappings and so on for different use-cases but this a very simple REGEX parsing which I am doing

Commands to run

Similar to WordCount, we compile first to create the jar file

1. javac -classpath $(hadoop classpath) WordCount.java
   (This is to compile the code to generate classfiles .class)
2. jar cf WordCount.jar WordCount*.class (to compile the class files into jars)

Run Format

Hadoop jar TopKWords.jar TopKWords <input files path> <intermediate files path> <output files path>

Command :- hadoop jar TopKWords.jar TopKWords sm11326/sm11326-hw1/data/output sm11326/sm11326-hw1/data/intermediate sm11326/sm11326-hw1/data/top-k-output

Question 3: Bonus (100 Points)

Compilation Steps

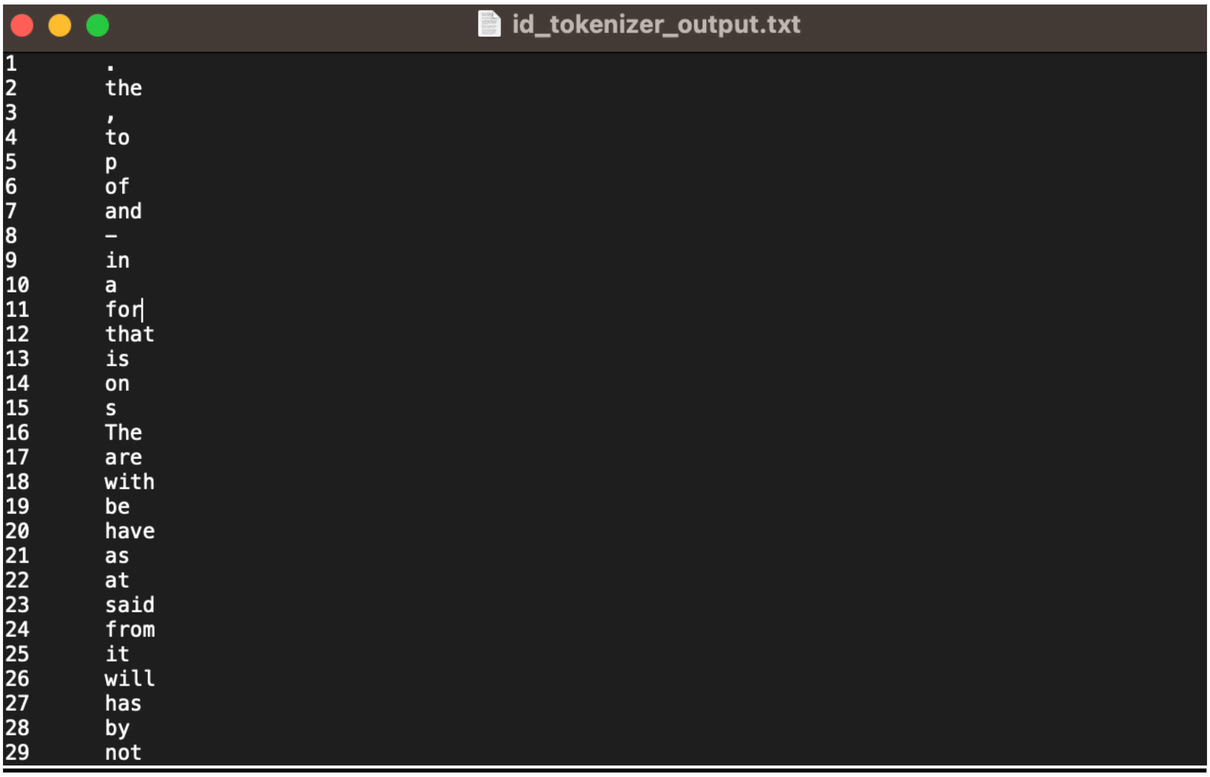
javac -classpath $(hadoop classpath) SimpleIDTokenizer.java
jar cf SimpleIDTokenizer.jar SimpleIDTokenizer*.class (to compile the class files into jars)

Now that we have a jar,
I am feeding the output of WordCount to SimpleTokenizer.

Command

hadoop jar SimpleIDTokenizer.jar SimpleIDTokenizer sm11326/sm11326-hw1/data/output sm11326/sm11326-hw1/data/token-output

Output



```
1       .
2       the
3       ,
4       to
5       p
6       of
7       and
8       –
9       in
10      a
11      for
12      that
13      is
14      on
15      s
16      The
17      are
18      with
19      be
20      have
21      as
22      at
23      said
24      from
25      it
26      will
27      has
28      by
29      not
```

Logic and Intention

Note - I am using a custom comparator to sort in descending order.

```
    @Override
            public int compare(Map.Entry<Text, Integer> o1, Map.Entry<Text,
Integer> o2) {
               return o2.getValue().compareTo(o1.getValue()); // Sort in
descending order
            }
         });
```

Job Mapper

- **Input**: Each line of the input consists of a word and its corresponding count (e.g., "word count").
- **Output**: For each word, the mapper emits a key-value pair where the key is the word (as Text) and the value is the count (as IntWritable).

- **Process**: The mapper splits the line into parts, checks if it has exactly two parts, and then sets the wordWritable and countWritable accordingly. It writes these pairs to the context.

Job Reducer (IDReducer):

```
 private PriorityQueue<Map.Entry<Text, Long>> countQueue = new
PriorityQueue<>(

          (a, b) -> Long.compare(a.getValue(), b.getValue())

      );
```

- **Input**: Receives a key (word) and an iterable of counts for that word.
- **Output**: After summing counts for each word, it emits the rank and the word.
- **Process**:
  - It uses a **Priority Queue** to accumulate the total count for each word, key-value mapping.
  - In the reduce method, it iterates through the counts for each word and sums them up.
  - In the cleanup method, it uses a CountMap and sorts the entries in descending order based on their total counts.
  - Finally, it assigns ranks starting from 1 to each word and writes the rank along with the word to the context.

Salutations

Since it was told in the class that I can take the help of AI to understand debug and code, I have used ChatGPT 3.5 & Perplexity to debug, generate lines, test and to understand the concepts. I don’t know exactly how much I used where, but in combination, these were the two AI tools which I have used for this assignment to generate pieces of code, to test it, to fix some errors in my code by uploading it to GPT and then asking it to correct me, and so on…

But in this process and also with the post-class conversations with the professor, I do understand the trick(bucket-iterator) way of doing things), essence of map-reduce, what the code is meant to do, how to re-construct a problem as a map-reduce problem and so on..

I was on the wrong track initially as I was confused with how combining local top-k works..but I understand it now after the assignment.

I also understood thoroughly, the program WordCount.java given by the professor and also the programs I have coded myself alongwith the help of AI tools, i,e SimpleIDTokenizer.java & TopKWords.java

- Developing and optimizing machine learning models to support ongoing research projects.
- Working closely with professors and academic teams on innovative AI solutions, specializing in deep learning and computer vision.
- Collaborating to publish papers on cutting-edge research findings in top-tier journals.

Thank you,
sm11326
Sai Nishanth