Akshat Mishra | netid: am15111 | Spring 2025

Big Data Assignment 2 GHW#2

Question 1. Sentence Count [50 Points] Write a MapReduce program to count the total number of sentences in a given input text file. Use the input file InputFiles.txt.

Step 1: Upload the input text file (InputFiles.txt), mapper.py and reducer.py.

Step 2: Use **"ls"** command to view the files if uploaded correctly.
Step 3: Use **"hdfs dfs -put InputFiles.txt"** to uploads a file from local to HDFS.

Step 4: Use **"hdfs dfs -ls"** to verify that the file is uploaded

```
Linux nyu-dataproc-m 6.1.0-29-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.123-1 (2025-01-02) x86_64

5 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades.log
 __  __    __  __     __  __
/\ "-.\ \  /\ \_\ \   /\ \/\ \
\ \ \-.  \ \ \____ \  \ \ \_\ \
 \ \_\\"\_\ \/\_____\  \ \_____\
  \/_/ \/_/  \/_____/   \/_____/

 __  __     ______   ______
/\ \_\ \   /\  == \ /\  ___\
\ \  __ \  \ \  _-/ \ \ \____
 \ \_\ \_\  \ \_\    \ \_____\
  \/_/\/_/   \/_/     \/_____/
Last login: Sun Feb 16 00:30:23 2025 from 35.235.244.34
am15111_nyu_edu@nyu-dataproc-m:~$ ls
InputFiles.txt  mapper.py  reducer.py
am15111_nyu_edu@nyu-dataproc-m:~$ hdfs dfs -put InputFiles.txt
am15111_nyu_edu@nyu-dataproc-m:~$ hdfs dfs -ls
Found 1 items
-rw-r--r--   1 am15111_nyu_edu am15111_nyu_edu   10485760 2025-02-16 00:32 InputFiles.txt
am15111_nyu_edu@nyu-dataproc-m:~$
```

Step 5: Run command :
"hadoop jar $HADOOP_HOME/Hadoop-streaming-3.3.6.jar -input InputFiles.txt -output1 -mapper "python mapper.py" -reducer "python reducer.py" -file mapper.py -file reducer.py"
It executes a MapReduce job using Hadoop Streaming, where: mapper.py processes the input data to generate key-value pairs.reducer.py aggregates the key-value pairs and produces the final output.The results are stored in the -output1 directory.

```
am15111_nyu_edu@nyu-dataproc-m:~$ hadoop jar $HADOOP_HOME/hadoop-streaming-3.3.6.jar -input InputFiles.txt -output output1 -mapper "python mapper.py" -reducer "python reducer.py" -file mapp
er.py -file reducer.py
2025-02-16 00:34:38,744 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapper.py, reducer.py] [/usr/lib/hadoop/hadoop-streaming-3.3.6.jar] /tmp/streamjob1787844713358342935.jar tmpDir=null
2025-02-16 00:34:39,942 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at nyu-dataproc-m.local./192.168.1.93:8032
2025-02-16 00:34:40,108 INFO client.AHSProxy: Connecting to Application History server at nyu-dataproc-m.local./192.168.1.93:10200
2025-02-16 00:34:40,835 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at nyu-dataproc-m.local./192.168.1.93:8032
2025-02-16 00:34:40,836 INFO client.AHSProxy: Connecting to Application History server at nyu-dataproc-m.local./192.168.1.93:10200
2025-02-16 00:34:41,069 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/am15111_nyu_edu/.staging/job_1737068921170_1591
2025-02-16 00:34:41,447 INFO mapred.FileInputFormat: Total input files to process : 1
2025-02-16 00:34:41,509 INFO mapreduce.JobSubmitter: number of splits:6
2025-02-16 00:34:41,684 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1737068921170_1591
2025-02-16 00:34:41,685 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-02-16 00:34:41,862 INFO conf.Configuration: resource-types.xml not found
2025-02-16 00:34:41,862 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-02-16 00:34:41,963 INFO impl.YarnClientImpl: Submitted application application_1737068921170_1591
2025-02-16 00:34:41,996 INFO mapreduce.Job: The url to track the job: http://nyu-dataproc-m.local.:8088/proxy/application_1737068921170_1591/
2025-02-16 00:34:41,998 INFO mapreduce.Job: Running job: job_1737068921170_1591
2025-02-16 00:34:52,104 INFO mapreduce.Job: Job job_1737068921170_1591 running in uber mode : false
2025-02-16 00:34:52,105 INFO mapreduce.Job:  map 0% reduce 0%
2025-02-16 00:35:03,201 INFO mapreduce.Job:  map 17% reduce 0%
2025-02-16 00:35:04,207 INFO mapreduce.Job:  map 100% reduce 0%
2025-02-16 00:35:12,251 INFO mapreduce.Job:  map 100% reduce 50%
2025-02-16 00:35:13,256 INFO mapreduce.Job:  map 100% reduce 100%
2025-02-16 00:35:15,274 INFO mapreduce.Job: Job job_1737068921170_1591 completed successfully
2025-02-16 00:35:15,366 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=1723929
```

Step 6: Use **"hdfs dfs -get output1"** it copies the **output1** directory from **HDFS to the local filesystem**.

And use **“ls output1”** to verify the files are made or not

```
am15111_nyu_edu@nyu-dataproc-m:~$ hdfs dfs -get output1
am15111_nyu_edu@nyu-dataproc-m:~$ ls output1
_SUCCESS  part-00000  part-00001
```

Step 7: **hdfs dfs -cat output1/part* > total_lines.txt** This concatenates and prints the contents of all part files (e.g., part-00000, part-00001, etc.) inside output1 from HDFS. The output is redirected (>) into a local file called total_lines.txt on your system.

cp total_lines.txt . This copies total_lines.txt to another location in your local filesystem.

```
am15111_nyu_edu@nyu-dataproc-m:~$ hdfs dfs -cat output1/part* > total_lines.txt
am15111_nyu_edu@nyu-dataproc-m:~$ cp total_lines.txt .
cp: 'total_lines.txt' and './total_lines.txt' are the same file
am15111_nyu_edu@nyu-dataproc-m:~$ pwd
/home/am15111_nyu_edu
```

Step 8: To see the output use **cat total_lines.txt**

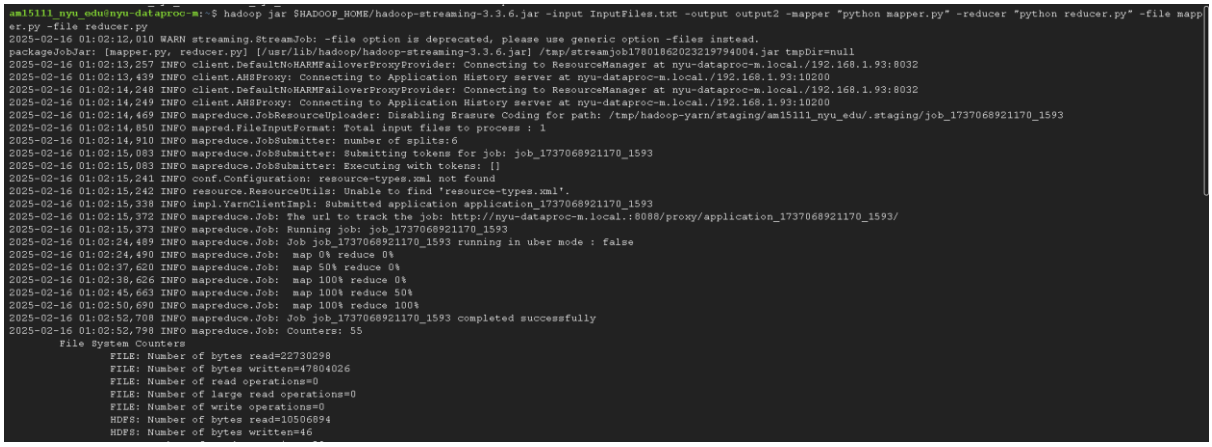
```
am15111_nyu_edu@nyu-dataproc-m:~$ cat total_lines.txt
Total Sentences: 132609
```

The file is present in Question1>Output Question 1> total_lines.txt

Question 2. Vowel Count [50 Points] Write a MapReduce program to count the occurrences of each vowel (a, e, i, o, u) in a given input text file. Use the input file InputFiles.txt.

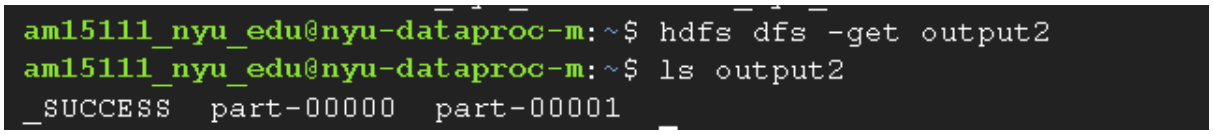
Repeat Steps 1-4 from question 1.

Step 5: **"hadoop jar $HADOOP_HOME/Hadoop-streaming-3.3.6.jar -input InputFiles.txt -output2 -mapper "python mapper.py" -reducer "python reducer.py" -file mapper.py -file reducer.py"**
It executes a MapReduce job using Hadoop Streaming, where: mapper.py processes the input data to generate key-value pairs.reducer.py aggregates the key-value pairs and produces the final output.The results are stored in the -output1 directory.

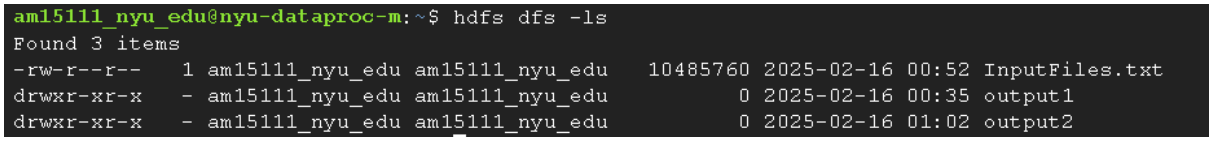


Step 6: Use **"hdfs dfs -get output2"** it copies the **output2** directory from **HDFS to the local filesystem**.

And use **"ls output2"** to verify the files are made or not



Step 7: **hdfs dfs -ls** is used to list files and directories in HDFS



Step 8: **hdfs dfs -cat output1/part* > vowel_count.txt** This concatenates and prints the contents of all part files (e.g., part-00000, part-00001, etc.) inside output1 from HDFS. The output is redirected (>) into a local file called total_lines.txt on your system.

cp vowel_count.txt . This copies total_lines.txt to another location in your local filesystem. Same as question 1.

To see the output use **cat vowel_count.txt**

```
am15111_nyu_edu@nyu-dataproc-m:~$ hdfs dfs -cat output1/part* > vowel_count.txt
am15111_nyu_edu@nyu-dataproc-m:~$ cp vowel_count.txt .
cp: 'vowel_count.txt' and './vowel_count.txt' are the same file
```

```
am15111_nyu_edu@nyu-dataproc-m:~$ cat vowel_count.txt
a       889179
e       1187879
i       724451
o       452262
u       534610
```

The file is present in Question2>Output Question 2> vowel_count.txt

Learnings from this Assignment:

1. Understanding Hadoop Streaming:
   - Learned how to execute MapReduce jobs using Python via Hadoop Streaming.
   - Understood the role of mapper.py in breaking down input data and reducer.py in aggregating results.
2. HDFS File Management:
   - Gained hands-on experience with HDFS commands like hdfs dfs -put, hdfs dfs -ls, and hdfs dfs -get.
   - Realized the importance of verifying file uploads before running jobs to avoid errors.
3. Debugging and Output Verification:
   - Used hdfs dfs -cat output1/part* to view output files and ensure correctness.
   - Learned that results need to be copied from HDFS to the local system for further use.
4. Automation & Command Efficiency:
   - Understood how redirection (>) helps store Hadoop output into local files for easy access.
   - Saw how repetitive steps can be streamlined to quickly validate results.
5. Sentence and Vowel Counting with MapReduce:
   - Explored text processing in Hadoop, applying different logic for sentence counting and vowel counting.
   - Observed how different problems require different key-value structures in MapReduce.
6. File and Output Organization:
   - Learned to maintain structured output directories (output1, output2) to keep results separate.
   - Understood the importance of naming files correctly (total_lines.txt, vowel_count.txt) for clarity.
7. Common Mistakes & Debugging:
   - Faced issues like incorrect paths, and syntax errors in Hadoop commands.
   - Realized how hdfs dfs -ls helps in troubleshooting missing or misnamed files.