Prachi Kotadia(A20549927)

CSP554—Big Data Technologies

Assignment #3 (Modules 03a & 03b, 15 points)

6. (3 points) Submit (1) a copy of this modified program and (2) a screenshot of the results of the program's execution as the output of your assignment.

Code:

```
# WordCount2.py

from mrjob.job import MRJob
import re

WORD_RE = re.compile(r"[\w']+")

class MRWordCount(MRJob):

    def mapper(self, _, line):
        for word in WORD_RE.findall(line):
            if word[0] >= 'a' and word[0] <= 'n':
                yield 'a_to_n', 1
            else:
                yield 'other', 1

    def combiner(self, word, counts):
        yield word, sum(counts)

    def reducer(self, word, counts):
        yield word, sum(counts)

if __name__ == '__main__':
    MRWordCount.run()
```

Query:

```
$ python WordCount2.py -r hadoop hdfs:///user/hadoop/w.data
```

Output:

```
job output is in hdfs:///user/hadoop/tmp/mrjob/WordCount2.hadoop.20240208.171949.978132/output
Streaming final output from hdfs:///user/hadoop/tmp/mrjob/WordCount2.hadoop.20240208.171949.978132/output...
"a_to_n"        46
"other" 49
Removing HDFS temp directory hdfs:///user/hadoop/tmp/mrjob/WordCount2.hadoop.20240208.171949.978132...
Removing temp directory /tmp/WordCount2.hadoop.20240208.171949.978132...
[hadoop@ip-172-31-60-244 ~]$ exit
logout
Connection to 54.162.40.182 closed.
```

8. (4 points) When you have accomplished this, please submit the following, (1) a copy of your MRJob code and (2) a copy of the output of the execution of that code.
Code:

```
#WordCount3.py

from mrjob.job import MRJob
import re

WORD_RE = re.compile(r"[\w']+")

class MRWordCount(MRJob):
    def mapper(self, _, line):
        for word in WORD_RE.findall(line):
            yield len(word), 1

    def combiner(self, word, counts):
        yield word, sum(counts)

    def reducer(self, word, counts):
        yield word, sum(counts)

if __name__ == '__main__':
    MRWordCount.run()
```

Query:

```
$ python WordCount3.py -r hadoop hdfs:///user/hadoop/
```

Output:

```
2       23
5       4
8       6
12      1
3       19
6       8
9       5
1       3
10      1
4       16
7       9
```

10. (5 points) When you have accomplished this, please submit the following, (1) a copy of your MRJob code and (2) a copy of the output of the execution of that code for at least the first 10 bigram key value pairs.

Code:

```
#WordCount4.py

from mrjob.job import MRJob
import re

WORD_RE = re.compile(r"[\w']+")

class MRWordCount(MRJob):

    def mapper(self, _, line):
        line = line.lower()
        words = WORD_RE.findall(line)
        for i in range(len(words) - 1):
            yield words[i] + ' ' + words[i + 1], 1
    def combiner(self, word, counts):
        # Sum up the counts for each word pair
        yield word, sum(counts)
    def reducer(self, word, counts):
        # Sum up the counts for each word pair
        yield word, sum(counts)
if __name__ == '__main__':
    MRWordCount.run()
```

Query:

```
$ python WordCount4.py -r hadoop hdfs:///user/hadoop/
```

Output:

```
"How your"      1
"The following" 1
"are more"      1
"as well"       1
"contained within"      1
"executed on"   1
"explains how"  1
"file, available"       1
"following two" 1
"how to"        1
"is run"        1
"more reference-oriented"       1
"on your"       1
"or reduce"     1
"reduce task."  1
"submitted. (Runners"   1
"task nodes,"   1
"task. (See"    1
"to be" 1
"to do" 1
"within the"    1
"your machine"  1
"your program"  1
"your second"   1
```

14. (3 points) Submit (1) a copy of this modified program and (2) a screenshot of the results of the program's execution as the output of your assignment.

Code:

```python
# Salaries2.py

from mrjob.job import MRJob

class MRSalaries2(MRJob):

    def mapper(self, _, line):
        # Split the input line into fields
        fields = line.split('\t')
        # Extract the annual salary from the fields
        annual_salary = float(fields[5])

        # Determine the salary group based on the annual salary
        if annual_salary >= 100000.00:
            salary_group = 'High'
        elif 50000.00 <= annual_salary <= 99999.99:
            salary_group = 'Medium'
        else:
            salary_group = 'Low'

        # Emit the salary group as the key and a count of 1 as the value
        yield salary_group, 1

    def combiner(self, salary_group, counts):
        # Sum up the counts for each salary group
        yield salary_group, sum(counts)

    def reducer(self, salary_group, counts):
        # Sum up the counts for each salary group and yield the result
        yield salary_group, sum(counts)


if __name__ == '__main__':
    MRSalaries2.run()
```

Query:

```
$ python Salaries2.py -r hadoop hdfs:///user/hadoop/Salaries.tsv
```

Output:

```
job output is in hdfs:///user/hadoop/tmp/mrjob/Salaries2.hadoop.20240208.174451.907295/output
Streaming final output from hdfs:///user/hadoop/tmp/mrjob/Salaries2.hadoop.20240208.174451.907295/output...
"High"  442
"Low"   7064
"Medium"        6312
Removing HDFS temp directory hdfs:///user/hadoop/tmp/mrjob/Salaries2.hadoop.20240208.174451.907295...
Removing temp directory /tmp/Salaries2.hadoop.20240208.174451.907295...
[hadoop@ip-172-31-60-244 ~]$ exit
logout
Connection to 54.162.40.182 closed.
```

15. Remember to terminate your EMR cluster and remove your S3 bucket!