CSP554—Big Data Technologies

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

1) As some places in this assignment you may want to create or edit a python file on the EMR master node. The editor that is available by default is call "vi." If you are unfamiliar with it, please refer to online documentation/tutorials:

- The vi editor tutorial (start here)
- Learning the vi and Vim Editors (an entire free book)
- vi command cheat sheet

2) Please read the documentation for "mrjob" from the "readthedocs" website as a reference – you will be using basic/core functionality, so you do not need to read about more advanced functionality.

3) Create a new EMR cluster the same as you did previously. Since you already have a security key (".pem" file) just use that one during cluster creation. Or, if you deleted your security key, just create a new one.

4) Install the mrjob library on your EMR master node.

a) ssh to the master node (/home/hadoop) as you did in assignment #2
b) Enter the following (note if the first command does not work, try the second)

sudo /usr/bin/pip3.7 install mrjob[aws]

or try:

sudo /usr/bin/pip3 install mrjob[aws]

5) Next you will set up to execute the provided WordCount.py map reduce program found in the "Assignments" section of the Blackboard.

Step 1:

Download the two files "w.data" and "WordCount.py" to your PC or Mac. They are part of the documents included with the assignment.

Step 2:

Note to prevent confusion: the default directory of your Linux account on the Hadoop master node is "/home/hadoop." But when we want to copy something to HDFS we will sometimes copy it to an HDFS directory beginning with "/user/hadoop." Be aware, the Linux and HDFS file system path names have nothing to do with one another. Any similarity in naming (such as the use of the directory name "hadoop") is just coincidental.

Now open another terminal window (but don't use it to ssh to the master node). This will allow you to access files on your PC or MAC to upload them to the Hadoop master node.

From this terminal window use the secure copy (scp) program to move the WordCount.py file to the /home/hadoop directory of the master node.

Step 3:

Do the same for the assignment file w.data. That is move it to the directory /home/Hadoop on the Hadoop master node Linux file system.

In this case copy the file from the Linux "/home/hadoop" directory to the Hadoop file system (HDFS), say to the directory "/user/hadoop"

Step 4:

Now execute the following

python WordCount.py -r hadoop hdfs:///user/hadoop/w.data

Note there must be three slashes in "hdfs:///" as "hdfs://" indicates that the file you are reading from is in the hadoop file system and the "/user" is the first part of the path to that file. Also note that sometimes copying and pasting this command from the assignment document does not work and it needs to be entered manually.

Check that it produces some reasonable output.

Note, the above command will erase all output files in hdfs. If you want to keep the output use the following command instead:

python WordCount.py -r hadoop hdfs:///user/hadoop/w.data **- -output-dir /user/hadoop/some-non-existent-directory**

6) Now slightly modify the WordCount.py program. Call the new program WordCount2.py.

Instead of counting how many words there are in the input documents (w.data), modify the program to count how many words begin with the small letters a-n and how many begin with anything else.

The output file should look something like

a_to_n, 12

other, 21

Now execute the program and see what happens.

7) (5 points) Submit a copy of this modified program and a screen shot of the results of the program's execution as the output of your assignment.

8) Now do the same as the above for the files Salaries.py and Salaries.tsv. The ".tsv" file holds department and salary information for Baltimore municipal workers. Have a look at Salaries.py for the layout of the ".tsv" file and how to read it in to our map reduce program.

9) Execute the Salaries.py program to make sure it works. It should print out how many workers share each job title.

10) Now modify the Salaries.py program. Call it Salaries2.py

Instead of counting the number of workers per department, change the program to provide the number of workers having High, Medium or Low annual salaries. This is defined as follows:

High	100,000.00 and above
Medium	50,000.00 to 99,999.99
Low	0.00 to 49,999.99

The output of the program should be something like the following (in any order):

High 20

Medium 30

Low 10

Some important hints:

- The annual salary is a string that will need to be converted to a float.
- The mapper should output tuples with one of three keys depending on the annual salary: High, Medium and Low
- The value part of the tuple is not a salary. (What should it be?)

Now execute the program and see what happens.

11) (5 points) Submit a copy of this modified program and a screen shot of the results of the program's execution as the output of your assignment.

12) Now copy the file u.data from the assignment to /user/hadoop. **NOTE: this version of u.data has fields separated by commas and not tabs.**

13) (5 points) Write a program to perform the task of outputting a count of the number of movies each user (identified via their user id) reviewed.

Output might look something like the following:

186: 2

192: 2

112: 1

etc.

Submit a copy of this program and a screen shot of the results of the program's execution (only 10 lines or so of the result) as the output of your assignment.

14) Remember to terminate your EMR cluster and remove your S3 bucket.

6th answer

```
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].lower() in 'abcdefghijklmno':
                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()
```

```
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 re.match(r'^[a-nA-N]\w*', word):
                yield 'a-n', 1
            else:
                yield 'other', 1

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

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

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