# CSP554—Big Data Technologies

**Note: Cutting and pasting the commands given below sometimes does not work, as occasionally there are some non-printing characters in this file. Just type the commands in manually.**

## Assignment #3 (15 points)

## Due by the start of the next class period

Assignments can uploaded via the Blackboard portal

Note: There may be short quiz questions about readings, assignments or articles (except extra credit) in the class period when they are due.

1. Read from (TW)

* Chapter 8
* Chapter 9
* Chapter 17

2) As some places in this assignment you may want to create or edit a python file on an EMR primary node. The editor that is available by default is call “vi.” If you are unfamiliar with its use some tutorial material has been placed on the Blackboard, in the “Free Books and Chapters” section, for your reference (and there are plenty more on line). The information includes:

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

3) Please read the document “mrjob Documentation,” which is located in the “Free Books and Chapters” section of the Blackboard, through the middle of page 23. But not every detail is important. I provide you with the exact commands needed to execute mrjob programs below.

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

5) Install the mrjob library on your EMR primary node.

1. ssh to the primary node (/home/hadoop) as you did in assignment #2
2. 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]

6) Next you will set up to execute the provided WordCount.py map reduce program found in the “Assignments” section of the Blackboard. This is the exact same program we saw in class.

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 primary 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 primary node). This will allow you to access files on your PC or MAC to upload them to the Hadoop primary node.

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

Step 3:

Do the same for the assignment file w.data. That is, move it to the directory /home/hadoop on the Hadoop primary 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”

To check make sure the file w.data is where you think it is in HDFS by executing:

hadoop fs -ls /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. If all is well you should see information in the output similar to this when the program finishes correctly:

"well" 1

"when" 1

"will" 1

"within" 1

"writing" 2

"your" 5

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**

5) 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 lower case letters a-n (a through n inclusive) and how many begin with anything else.

The output file should look something like

a\_to\_n, 12

other, 21

Note, do not force words to all lower case. Now execute the program and see what happens.

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

7) Let’s modify the WordCount.py program again. Call the new program WordCount3.py.

Instead of counting words, calculate the count of words having the same number of letters. For example, if we have a file consisting of one record of the form:

hello there joe

our job should output key value pairs similar to the following:

3, 1

5, 2

Hint, the key in a key-value pair can be an integer just as well as a string.

So, your task is to write a MrJob MapReduce program which again accepts the following file as input

hdfs:///user/hadoop/w.data

and outputs key value pairs where each one has a key with is some number of characters, and the value a count of words having that many characters. Note, please convert all words to lower case on input, so “Hello” and “hello” become the same word.

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.

9) Again, modify the WordCount.py program. Call the new program WordCount4.py.

Now we will write a MapReduce job to calculate the count of unique per record word bigrams. A word bigram is a two word sequence. For example, if we have a file consisting of records of the form:

hello there joe

hi there

there joe go

joe

Bigrams for these records are create by sliding a two word “window” across the words of the record.

hello there joe => “hello there”, “there joe”

hi there => “hi there”

there joe there => “there joe”, “joe there”

joe => *Note, this record has no bigrams*

Notice, that there are 2 instances of the word bigram “there Joe”.

So, your task is to write a MrJob MapReduce program which accepts the following file as input

hdfs:///user/hadoop/w.data

and outputs key value pairs where each one has a key which is some word bigram string, and the value a count of the number of occurrences of that word bigram. Note, please convert all words to lower case on input, so Hello and hello become the same word.

Our job should output key value pairs similar to the following:

“hello there”, 1

“hi there”, 1

“joe there”, 1

“there joe”, 2

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.

11) 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.

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

13) 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.

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

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