Lab 4: An Introduction to Spark

The purpose of this lab is to get some experience with Spark. You might use the lecture on Spark as a reference. Here are the two subtasks.

Subtask One: Running WordCount on Spark

First, fire up an Amazon AWS cluster. Make sure when you create your cluster that it can run Spark (that is, click the Spark checkbox). Also, **and this is important**, *you need to create your cluster in the US East region (N. Virginia), or else there is a chance you won't be able to directly access my S3 data (it didn't used to be like this, but now it is!).* To sign into a particular region, look at the top left of the console when you sign in (next to your name)… it will give your region.

After you have created your cluster, connect to your master node using SSH.

1) At the Linux command prompt, type `pyspark`. This will open up a Python shell that is connected to your Spark cluster. Note that sometimes it takes a few minutes before `pyspark` will work, as you have to wait for Spark to start up. If you get a bunch of errors when you try to run the code, exit `pyspark` and then try again in a few minutes.

2) In the shell, copy and past the following, simple function, that uses Spark to count the top 200 most frequently occurring words in a text file:

```
def countWords (fileName):
     textfile = sc.textFile(fileName)
     lines = textfile.flatMap(lambda line: line.split(" "))
     counts = lines.map (lambda word: (word, 1))
     aggregatedCounts = counts.reduceByKey (lambda a, b: a + b)
     return aggregatedCounts.top (200, key=lambda p : p[1])
```

3) Now that you've got that function all set, check out the following four files, which contain a bunch of text data that I've loaded onto Amazon's S3 cloud storage service:

```
https://s3.amazonaws.com/chrisjermainebucket/text/Holmes.txt
https://s3.amazonaws.com/chrisjermainebucket/text/dictionary.txt
https://s3.amazonaws.com/chrisjermainebucket/text/war.txt
https://s3.amazonaws.com/chrisjermainebucket/text/william.txt
```

I'll let you guess what these files contain! You can count the top 200 words in any of them by simply typing, at the Python command prompt, for example:

```
countWords ("s3://chrisjermainebucket/text/Holmes.txt")
```

(Be careful if you copy/paste the above from the PDF; you don't want to get weird

characters like smart quotes). Or, if you want to count all of them at the same time:

```
countWords ("s3://chrisjermainebucket/text/")
```

4) Type control-d to exit the shell when you are done.

Subtask Two: Completing a Spark Program

In this task, I've written an almost-complete code that analyzes a corpus of real text documents. Your task us to complete my code.

You can start by taking a look at the data set. Check out

https://s3.amazonaws.com/chrisjermainebucket/comp330_A6/20_news_same_line.txt.

(again, be careful with copy and paste here!). There are 19997 lines in this file, each corresponding to a different text document. The goal here is to build, as an RDD, a dictionary. The dictionary will have as its key a number from 0 to 19,999 (this is a rank) and the value is the the word corresponding to that rank. The words will be ranked according to their frequency in the corpus, with 0 being the most frequent, 19,999 being the 20-thousandth-most frequent.

Start by reading my code (I've put a copy in Canvas). Then, you can copy and paste my code (up to but not including the line with the question marks) into Spark, line by line. Note that nothing interesting happens until you enter the line that tries to collect the results (using the call to `top`), at which point Spark goes off, performs all of the computations, and tries collect the results into `topWords`. At this point, `topWords` is a local variable. In the Python shell, you can manipulate it, change it, and print it, just as you would any other local variable.

Once you've made it to the question marks, come up with an appropriate lambda that is going to attach the correct word to each number, putting the results into an RDD. Replace the question marks with your lambda. The key of the tuple you create and put into the RDD via the lambda should be the word; the value is the index (the number from 0 to 19,999). Your lambda will refer to the local variable `topWords`.

After you run `dictionary.top (10)` copy and paste some of the results into Canvas. Note that this will return the top 10 words by name (the key), and not the index (the value).