Lab 1 Exercise 2

Summarize sent emails

# Objective:

The objective of this exercise is to write a MapReduce program to calculate the total number of ***sent emails per email domain*** for a given person in the Enron data set. The output of your final solution must be separated into 2 different output directories (part-r-00000 and part-r-00001), where the first output directory contains email domains starting with the letters a-m and the second output directory contains email domains starting with the letters n-z. The exercise is a slight variation of the canonical word count program.

Below is an *example* of what your output should look like, where the domain is the output key and the output value is the total number of emails sent to that domain.

$ **cat part-r-00000**

2onesource.com 3

aol.com 99

cityofnapa.org 8

enron.com 2

gte.net 9

hotmail.com 31

ix.netcom.com 15

lycosmail.com 8

mindspring.com 10

$ **cat part-r-00001**

nextera-pcg.com 2

provdiv.ttm.bg 1

rwblaw.com 8

sbu.edu 8

sirius.com 14

sprynet.com 4

tribune.com 11

usa.net 2

worldnet.att.net 8

yahoo.com 12

# Data:

The data is excerpted from the infamous Enron email corpus (<https://en.wikipedia.org/wiki/Enron_Corpus>). The data is provided in a hierarchical directory structure, so you'll need to add input to the job recursively. Also note that the data provided for this exercise pertains to only one sender ([jskilling@enron.com](mailto:jskilling@enron.com)) -- the entire data is way too large to process on our virtual machine.

# Driver:

Partial code for the driver has been provided. Implement the TODO's in the driver code provided to properly construct and submit the job. Do NOT hard-code the number of reducers in the driver – this is to specified at the command line.

# Mapper:

Partial code for the mapper has been provided. Implement the TODO's in the mapper code provided to perform the map phase.

# Reducer:

Complete code for the reducer has been provided. You do not need to modify any code in the reducer, but you should read the code to determine what it's doing.

# Partitioner

Partial code for the partitioner has been provided. Implement the TODO's in the partitioner code to partition the data as required. Note that the partitioner must support the following: 0 reducers (map-only), 1 reducer (put all output in single directory), and 2 reducers (put all output in 2 directories as previously defined). All other numbers of reducers should return -1 from the getPartition() method.

# Scripts:

Complete code for the rebuild script has been provided. You do not need to modify any code in the rebuild script for your solution, but you should read the script to determine what it's doing. Syntax for the rebuild script is as follows:

$ **./rebuild.sh**

Complete code for the rerun script has been provided. You do not need to modify any code in the rerun script for your solution, but you should read the script to determine what it's doing. Syntax for the rerun script is as follows (where supported values of n are 0, 1, and 2).

$ **./rerun.sh –Dmapreduce.job.reduces=*n***

# How to begin:

Follow these steps for beginning work on your exercise:

1. Login to your sandbox as the user01 user.
2. Copy the ENRON.zip file to the /user/user01 directory
3. Unzip the ENRON.zip file and perform your work from there.
4. Read the rebuild.sh and rerun.sh scripts so you know what they are doing.

# Hints:

Consider the hints below when writing your solution to this exercise.

1. You should first implement the mapper in its entirety before implementing any code in the partitioner. In order to test the correctness of your mapper, you should run the launcher as follows:

$ **./rerun.sh –Dmapreduce.job.reduces=0**

1. Once your mapper code is complete, implement and test your driver and partitioner.
2. Do not modify any other code except that which has been specified in the TODO lists.
3. You may, but do not need to, use an IDE to implement this code. You can simply use a text editor along with the rebuild and rerun scripts to determine if your code is compiling and executing correctly.
4. The graders will be using the same rebuild.sh and rerun.sh scripts as have been provided to you when they grade your submissions. Make sure your code builds and executes correctly with these scripts before submitting.

# Submission:

Read very carefully the following instructions for how to create and submit your solution to this exercise. If you have any questions at all, please ask them well in advance of your submission. You will lose points if your submission does not conform to these criteria.

1. Create a zip file ***in your sandbox*** which contains the EnronDriver.java , EnronMapper.java, and EnronPartitioner.java code as follows.

$ **cd /user/user01/ENRON**

$ **zip LNAME\_FNAME\_L1E2.zip ./EnronDriver.java ./EnronMapper.java \**

**./EnronPartitioner.java**

Replace LNAME with your last name, and FNAME with your first name. So, for a student named John Doe:

$ **zip DOE\_JOHN\_L1E2.zip ./EnronDriver.java ./EnronMapper.java \**

**./EnronPartitioner.java**

1. Upload the zip file to the CANVAS system.