MAPREDUCE LAB, DUE: SEPTEMBER 17, 11:59PM EST

1. Instructions

1.1. **What to look at.** The only references you may use for this lab are:

- The mrjob api reference: `https://mrjob.readthedocs.io/en/latest/`
- Any material on Canvas

In particular, do not use stackoverflow or chegg for two reasons, they often give bad advice.

1.2. **How to collaborate.**

- Write down (in your code) the names of people you collaborate with.
- You can look at and discuss code with your collaborators *only for the purposes of debugging.*
- You *cannot* copy/email/transmit code. Your code must be typed by you without looking at other people's code.
- You can discuss algorithmic strategies (pictures are fine, pseudocode is not)

1.3. **Getting the code and submitting the code.** Use update-starter to get the code and submit to gradescope. You can submit multiple times based on autograder responses **Make sure to only commit human-generated files, and no other files into your repository.**

After running update-starter, you should see an mrlab directory for this assignment and an examples directory that has example mrjob code.

1.4. **The Assignment.**

Question -1. Do not put any tabs in your python code (use only spaces, not tabs). The reason is that different computers interpret tabs differently. This means python code with tabs can work differently on different computers. Autograder will reject code that has tabs in it.

Question 0. How to run your code:

(1) Your github repository has a directory called **mrjob_example** (it is not inside the homework directory) with sample mrjob code for doing wordcount (as covered in class slides) and a run script for how to run it using the worker nodes on the cluster. Use the linux command line utiltiy **cp** to copy files instead of using copy/paste. The reason for this is that copy/paste can introduce strange invisible characters that will mess up the result (typically this happens if you are using windows and has caused students to waste hours of their time in the past).
(2) Before running your code, fill in the testdata.txt and expected_test_output.txt files representing small test data (testdata.txt) and what the corresponding output should be for that test data (expected_test_output.txt).
(3) Check your code for obvious errors: **pylint -E mycode.py**
(4) To run your code on the test data, type **python3 mycode.py testdata.txt > actualresult.txt** (don't add the actual result to github, as it is generated by code, not handwritten by you).
(5) To run your code using the cluster workers, edit the run script (starting with the template in **mrjob_example**) to make the following changes:
    - Update the hdfs output location (2 places)
    - Update the name of your code file (1 location)
    - Update the hdfs input file name (1 location)
    - Optionally update the number of reducers (1 location)

    and then type **source run**.
(6) If you skip directly to step 5, you are 99% guaranteed to waste a lot of time trying to debug an error that would have been idnetified in steps 1-4.

Question 1. This question uses the war and peace hdfs dataset in /datasets/wap. In this question, we are going to do a variation of wordcount, where we only want to know the count for words that have 5 characters or more (i.e., we do not care how many times the word "omg!" appears because it has 4 characters).

- Edit the **mrlab/q1/testdata.txt** and **mrlab/q1/expected_test_output.txt** files as explained in Question 0.
- Edit the run script **mrlab/q1/run**.
- Put your mapreduce program in **mrlab/q1/notfour.py**
- Remember that reducers don't get a list of values (it is an iterator, not a list and you can only go through an iterator once)

Question 2. In this question, we are going to do a variation of Question 1, where (in addition to only wanting words with at least 5 characters) we also only care about words that appear more times than they have characters. For example, if the word "iPhone" appears 6 times in war and peace, we don't want it in the output because the number of total appearances (6) is not $>$ than the number of characters (6). If you are stuck, think carefully about where is the best part to do the appropriate checks in the mapreduce framework.

- Edit the **mrlab/q2/testdata.txt** and **mrlab/q2/expected_test_output.txt** files as explained in Question 0.
- Edit the run script **mrlab/q2/run**.
- Put your mapreduce program in **mrlab/q2/notiphone6.py**
- Remember that reducers don't get a list of values (it is an iterator, not a list and you can only go through an iterator once)

Question 3. In this question, we are also using the war and peace dataset. We are interested in questions like "how many times does the character '#' appear in position 0 of a word" (e.g., #mapreduce), how many times does the character *o* appear in position 3 of a word" (e.g., iPh**o**ne). So for each combination of character and position, we want to know how many words in war and peace have that character in that position. For example, if there is an output with key **["a", 2]** and value **999**, it means that the character 'a' appeared in position 2 exactly 999 times (which is the same as 999 words in war and peace had character 'a' in position 2).

- Edit the **mrlab/q3/testdata.txt** and **mrlab/q3/expected_test_output.txt** files as explained in Question 0.
- Edit the run script **mrlab/q3/run**.
- Put your mapreduce program in **mrlab/q3/characterposition.py**
- Remember that reducers don't get a list of values (it is an iterator, not a list and you can only go through an iterator once)