CSE427S

HOMEWORK 4

M. Neumann

Due: THU 27 SEPT 2018 4PM

Getting Started

Update your SVN repository.

When needed, you will find additional materials for *homework x* in the folder `hwx`. So, for the current assignment the folder is `hw4`.

Hint: You can **check your submission** to the SVN repository by viewing `https://svn.seas.wustl.edu/repositories/<yourwustlkey>/cse427s_fl18` in a web browser.

SUBMISSION INSTRUCTIONS

WRITTEN:

- all written work needs to be submitted electronically in *pdf format*[1] via GRADESCOPE
- provide the following information on *every* page of your `pdf` file:
  - name
  - student ID
  - wustlkey (your `wustlkey` indicates the location (SVN repository) for your code submissions; without this information we will not be able to grade your code!)
- start every problem on a *new page*
- **FOR GROUPS**: make a **group submission** on GRADESCOPE and provide the following information for **all group members** on every page of your `pdf` file: names, student IDs, and location of code submissions (**one** student's `wustlkey`)[2]

CODE:

- code needs to be submitted via SVN repository commit (detailed submission instructions are provided whenever code submissions are required)
- make sure to always use the required *file name(s)* and *submission format(s)*
- **comment your code** to receive maximum credit

[1] Please, **type your solutions** or use **clear hand-writing**. If we cannot read your answer, we cannot give you credit nor will we be able to meet any regrade requests concerning your writing.

[2] It is sufficient to commit code to one SVN repository. If you do not specify the repository for your group's code submission clearly, we will only check the first repository, sorting wustlkeys alphabetically.

Problem 1: Passing a Parameter via the Command Line (40%)

You will write an Average Word Length program that uses a Boolean parameter called *caseSensitive* to determine whether the Mapper class should treat upper and lower case letters as different (*case-sensitive*) or whether all letters should be converted to lower case (*case-insensitive*).
Preparation: Copy the package including the Driver, Mapper, and Reducer code you have written earlier from `~/workspace/avgwordlength/src/` to the following directory:

```
~/workspace/toolrunner/src/
```

Note on Copying Source Files
You can use Eclipse to copy a Java source file from one project or package to another by right-clicking on the file and selecting Copy, then right-clicking the new package and selecting Paste. If the packages have different names (e.g. if you copy from average-wordlength.solution to toolrunner.stubs), Eclipse will automatically change the package directive at the top of the file. If you copy the file using a file browser or the shell, you will have to do that manually.

Use the following **test input** to test your implementation:

```
No now is definitely not the best time
```

The case-sensitive output for this test input (using one Reducer) would be:

```
N    2.0
b    4.0
d    10.0
i    2.0
n    3.0
t    3.5
```

(a) Write down the Mapper output, Reducer input, and Reducer output using the case-insensitive version of your program for the **test input** provided above.

(b) Modify the `AvgWordLength` driver to use ToolRunner. As before, test your implementation on the **test input** provided above. Comment your code to receive maximum credit. Are there any differences in the job execution?

(c) Modify the `LetterMapper` to use the configuration parameter `caseSensitive` to either perform *case-sensitive* processing or *case-insensitive* processing by writing a `setup()` method to get the value of the parameter. Case sensitive processing should be your **default**. As before, test your implementation on the **test input**. Comment your code to receive maximum credit. Are there any differences in the job execution?

(d) Run your implementation (using the default parameter) on the `shakespeare` folder in HDFS (make sure the folder only contains the following files: poems, histories, comedies, and tragedies). What are the average word lengths for the following letters:

- A

- W
- a
- t
- z

(e) Run your implementation providing `caseSenstive=false` as a runtime parameter on the `shakespeare` folder in HDFS (make sure the folder only contains the following files: poems, histories, comedies, and tragedies). Provide the command for this the `hw4.pdf` file. Does the order of command line inputs matter (what happens if you specify the parameter after the output directory)? What are the average word lengths for the following letters:

- a
- w
- z

Submit the modified `.java` files **`AvgWordLength.java`** and **`LetterMapper.java`** to the hw4 folder to your SVN repository.

To add the .java files to your SVN repo before committing run:

```
$ svn add AvgWordLength.java
$ svn add LetterMapper.java
$ svn commit -m 'hw4 problem 1 submission' .
```

Problem 2: Word Count with a Partitioner (60%)

In this problem you will implement a Partitioner for the WordCount MAPREDUCE program that assigns positive, negative, and neutral words to different reducers.

(a) Implement a Partitioner using the stubs `SentimentPartitioner.java` provided in your SVN repository. Your Partitioner should send each key-value pair to one out of three Reducers based on whether the key is appearing in the list of positive words (`positive-words.txt`), in the list of negative words (`negative-words.txt`), or in neither of them. The files including the word lists are also in your SVN folder. Make sure you ignore all lines in the .txt files starting with `;`. Use **Distributed Cache** to access the files in the Partitioner. Remove the provided comments and add your own comments to your code to receive maximum credit.

(b) Observe the `SentimentPartitionerTest.java` program provided in your SVN repository. You can use it to test if your partitioner works correctly. It contains an example to test for the correct assignment of one example positive word.

Add the three tests specified in the stub file to the program to test your Partitioner and run it. Comment your code to receive maximum credit.

(c) Modify the WordCount Driver to use your Partitioner and the appropriate number of Reducers. Modify the Mapper to transform the input words to all lower-case letters (i.e., make WordCount *case-insensitive*, **no** parameter handling via ToolRunner

required). Run the MAPREDUCE job on the `shakespeare/poems` data (in HDFS). How many different *positive*, *negative*, and *neutral* words did Shakespeare use in his poems? Using the counts for positive and negative words compute the sentiment score s and the positivity score p using:

$$s = \frac{positive - negative}{postitive + negative}, \quad p = \frac{positive}{postitive + negative}.$$

Hint: use a pipe of `cat` in HDFS and `wc -l` to count the number of lines in the respective Reducer output files (you do not need to copy the result files from HDFS to your local file system).

Based on these statistics, do you think Shakespeare's poems suggest positive or negative emotions?

(d) Those sentiment statistics are not the best way to measure emotions from text. Describe two cases where it actually breaks (i.e., the above statistics are not a meaningful measure for the true emotion). Suggest an improved statistic or way to measure sentiments in text documents overcoming both limitations you identified.

Make sure you update **`SentimentPartitioner.java`** and **`SentimentPartitionerTest.java`** in the `hw4` folder in your SVN repository with your solution. **Then commit your solution:**

```
$ svn commit -m 'hw4 problem 2 submission' .
```

Bonus Problem (5% up to a max. of 100%) - no group work!

Write a review for this homework and store it in the file `hw4_review.txt` provided in your SVN repository (and commit your changes). This file should only include the review, **no other information** such as name, wustlkey, etc. Remember that you are not graded for the content of your review, solely it's completion.

Provide the star rating for `hw4` via this link: `https://wustl.az1.qualtrics.com/jfe/form/SV_aaAc0ILmBleyKzj`.
You can only earn bonus points if you write a meaningful review of **at least 50 words** and provide the corresponding star rating. Bonus points are given to the **owner of the repository only** (no group work!).

Submit your review in the file **`hw4_review.txt`** provided in the `hw4` folder in your SVN repository. **To commit the file run:**

```
$ svn commit -m 'hw4 review submission' .
```

Take 1 minute to provide a star rating for this homework – your star rating should match