CSE427 – Homework 4

M. Neumann

Due THU 02/18/2016 10am

Getting Started

Update your SVN repository. Find instructions on how to checkout, update, and commit to your SVN repository here: `http://sites.wustl.edu/neumann/resources/cse427s_resources/`

> 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_sp16` in a web browser (mind browser caching!).

Indicating Group Work

Use the file `partners.txt` to indicate group work. Follow these instructions exactly, to ensure to get credit!

- `partners.txt` needs to include up to 2 wustlkeys in the first two lines (one line per wustlkey)
- **first line/wustlkey is the repository, where the solution is located**. We will **only** consider the submission in this repository!
- Every student in a group needs to have **the same** `partners.txt` in the hwx folder in their repository (indicating that the partnership are **mutually accepted**)!
- If you do not have a partner, try to find one. If you want to submit on your own, indicate your wusltkey in the first line of `partners.txt` and leave the second line blank.

Problem 1: WordCount with a Combiner (40%)

For this problem you will add a Combiner to the WordCount MapReduce program.

Preparation: Copy the `WordMapper.java` and `SumReducer.java` from `~/workspace/wordcount/src/stubs` to the following directory:

```
~/workspace/combiner/src/stubs
```

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

```
To be or not to be that is not
```

The output for this test input would be:

```
To     1
be     2
is     1
not    2
or     1
that   1
to     1
```

(a) Why can you use the `SumReducer` as Combiner for the WordCount problem? Come up with an example Reducer operation (which is **not** the mean/average computation discussed in the lecture) where you cannot use the Reducer as Combiner.

CAUTION: before proceeding, make sure you have a .jar of the original WordCount program (not using a Combiner). You will need this for part (c)!

(b) Add a Combiner to the WordCount program (your Combiner should be a "*SumCombiner*") and rename your job to "Word Count with Combiner". Provide the respective lines of code in the `hw4.txt` file. Test your solution on the **test input**. Does the result of your job execution change compared to the original WordCount computation?

(c) Now, run both jobs on the `shakespeare` data and observe the job execution statistics of both jobs. To do so, open the **Hadoop JobTracker web interface** in a web browser (`http://localhost:50030/jobtracker.jsp`). On this page you will find an overview of completed jobs. Find the ones with names "Word Count Driver" and "Word Count with Combiner". Open both and compare the total number of counters (last column). Make sure that you compare the jobs using the `shakespeare` data and **not** the test input!

Do the following statistics (counters change)? If yes, provide the residual (original − with Combiner). I.e., whenever the counter is higher for the job using a Combiner, then the residual should be negative otherwise is should be positive.

- number of bytes read
- number of bytes written
- map output records
- combine input records
- combine output records
- reduce input groups
- reduce input records

How many key-value pairs could be combined using the Combiner?

(d) By comparing **CPU time spent** and **physical memory snapshot** of both job executions, argue whether is was a good or bad idea to use a Combiner for counting the words in the `shakespeare` data. Come up with a case where the use of a Combiner is inevitable. Why is the total time spent for the Mappers higher when using a Combiner?

Submit your answer to (a-d) by editing the **`hw4.txt`** file in your SVN repository. **Do not submit any `jar` files, Drivers, Mappers, or Reducers for this problem!**

Problem 2: Word Count with a Partitioner (40%)

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.

(b) Write a test program to test your Partitioner using the stubs provided in your SVN repository named `SentimentPartitionerTest.java`. Run your test and take a screen-shot of the result.

(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).

Submit your answer to (a) and (b) by updating the modified **`SentimentPartitioner.java`** and **`SentimentPartitionerTest.java`** in the `hw4` folder in your SVN repository. Submit the second part of answer (b) by renaming the screen-shot in **`problem2b.png`** (you can also use .jpg) and `svn add problem2b.png` to the `hw4` folder in your SVN repository. Submit your answer to (c) by editing the **`hw4.txt`** file in your SVN repository. **Do not submit any `jar` files, Drivers, Mappers, or Reducers for this problem!**

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. 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.

You can only earn bonus points if you write **at least 50 words**. Bonus points are given to the **owner of the repository only** (no group work!).