CPSC 4330 Big Data Analytics
Winter 2021

Homework 2

Due: 6:00 pm, Monday, Feb. 8th

In this homework, you will again use the publicly available Amazon customer review datasets to do an analysis. This time we are interested in pairs of users with similar preferences. Specifically, we want to know the pairs of users ("customer_id" in the review data) who gave the rating ("star_rating" in the review data) of at least 4 to at least 3 common products. The output must include pairs of users ("customer_id") and the list of common products ("product_id" in the review data). The output must be sorted by the first customer_id in the pairs of users and must not contain duplicates. Also, a user pair does not have order. That is to say, for any two customer_id *u1* and *u2*, <*u1*, *u2*> and <*u2*, *u1*> are considered as the same pair.

To help you understand the problem, I simplify the input review data to make it only have three columns (customer_id, product_id, rating), as shown below. For example, the 1st line means user *u1* gave a rating 4 to the product *p1*. This simplified sample file is tab delimited.

```
u1	p1	4
u2	p1	4
u3	p2	4
u3	p1	4
u1	p2	4
u2	p2	4
u1	p3	4
u2	p3	4
u3	p3	5
u4	p1	4
u5	p2	4
u4	p2	5
u1	p1	5
u4	p2	4
u4	p3	3
```

For the simplified input file, the sample output file is provided below. For example, the 1st line means that the users *u1* and *u2* gave a rating of at least 4 to the products *p3, p2, p1* (note that in the product list, the order does not matter).

```
u1	u2	p3,p2,p1
u1	u3	p3,p2,p1
u2	u3	p3,p2,p1
```

Here are some hints to help you solve the problem:

1. You may need a custom WritableComparable class to hold the user pair. A file *UserPairWritable.java* is provided to you (posted on Canvas) for this purpose.
2. You may need two mapreduce jobs for this program.
   - The first job creates the pairs of users who gave a rating of at least 4 to a common product.
   - The 2nd job aggregates all products of the same pair of users and outputs the pair of users if there are at least 3 common products.
   - You need to chain the two jobs. That is to say, after job 1 finishes, job 2 automatically starts taking the output of job 1 as input.

You can test your code on the simplified input data. If the output is correct, you can get one actual data set file to test on. Here is a file https://amazon-reviews-pds.s3.amazonaws.com/tsv/amazon_reviews_us_Luggage_v1_00.tsv.gz (about 58 MB). You can download it from the URL with the web browser, or you can use wget on the command line to download it:

```
wget https://amazon-reviews-pds.s3.amazonaws.com/tsv/amazon_reviews_us_Luggage_v1_00.tsv.gz
```

Note that you need to modify your program a bit before testing it on the actual review data because

- the simplified input data and the actual Amazon review data have different data formats
- the simplified input data does not have a header; the actual Amazon review data has a header (i.e. the 1st line has all column names).

If your program works fine with the actual data file, you can upload your application to AWS, scale out worker nodes and run on some larger data sets. Since the 1st mapreduce job will output a large amount of data, you can just analyze the reviews in the file https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Electronics_v1_00.tsv.gz (about 667 MB).

Review exercise 4 on how to use AWS to run your mapreduce code, but make the following changes:

- For the input argument of your job, you can point it to the S3 bucket where data files are already made publicly available:
  - s3://amazon-reviews-pds/tsv/amazon_reviews_us_Electronics_v1_00.tsv.gz
- For the output argument (the output of the 2nd job), you'll still want to point it to your bucket, but you may want to name a new directory (to avoid collision with any existing directories in your bucket)
- Your program needs an additional argument, which is to save the output of the 1st job. For this argument, you still point to your bucket, but to a directory different from the output directory of the 2nd job.
- On the "Step 2: Hardware" page, you can increase the number of worker nodes (e.g. changing to 4 worker nodes)
- When your program is done, check the results to make sure everything looks good. Also,
  - **remember to make sure that the cluster is terminated when you're done!**
  - **on S3, remember to delete the folder that contains the output files of the 1st job. The files are huge and the storage on S3 does cost your AWS educate account's credit! But only delete those files when your program is done with producing the final output, because the 2nd job might be still using those files.**

Submission

Use Canvas to submit the following: (You can compress the files if that's convenient)

- All your source code files (.java)
  - Only include source files, not compiled files (i.e. no .class or .jar files)
  - Your submission should have 6 java files:
    - Mapper1.java
    - Reducer1.java
    - Mapper2.java
    - Reducer2.java
    - Hw2Driver.java
    - UserPairWritable.java (the one that is provided to you)
- All output files of running the program on the review file amazon_reviews_us_Electronics_v1_00.tsv.gz. Note that the output files that you submit should be from the final output directory of your entire program, not from the output directory of the 1st mapreduce job.