IERG 4300 Fall 2019 Homework #1

Release date: Sep 20, 2019

Due date: Oct 2, 2019 (Wed) 11:59am. (i.e. noon-time)

The solution will be posted soon after the deadline. No late homework will be accepted!

Every Student MUST include the following statement, together with his/her signature in the submitted homework.

I declare that the assignment submitted on Elearning system is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/.

Signed (Student_____) Date:_____

Name_____ SID_____

Submission notice:

- Submit your homework via the elearning system

General homework policies:

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value, and justify any assumptions you make. You will be graded not only on whether your answer is correct, but also on whether you have done an intelligent analysis.

Q1[100 marks + 20 bonus marks]: Similar Users Detection in the MovieLens Dataset

Similar user detection has drawn lots of attention in machine learning field which is aimed to group users with similar interests, behaviours, actions or general patterns. In this homework, you will implement a similar users detection algorithm for the online movie rating system. Basically, users who rate similar scores for the same movies may have common tastes or interests and be grouped as similar users.

To detect similar user, we need to calculate the similarity between each user pair. In this homework, the similarity between a given pair of users is measured as **(the total number of movies which received the same rating/scores from the two users)** divided by **(the total number of (distinct) movies watched by two users).** The following is the formal definition of similarity: Let M(A) be the set of all the movies user A has watched and let $R_A(x)$ be the rating of movie x by user A. Then the similarity between user A and user B is defined as:

$$\text{Similarity}(A,B) = \frac{|\{x \in M(A) \cap M(B) : R_A(x) = R_B(x)\}|}{|M(A) \cup M(B)|} \quad \text{.....(**)}$$

where |S| means the cardinality of set S.

(Note: if $|M(A) \cup M(B)| = 0$, we set the similarity to be 0.)

The following figure illustrates the idea:

UserId	MovieId	Rating
A	a	3
A	b	3
A	c	1
B	b	3
B	c	1
B	d	1
C	b	3
C	d	2
D	b	2
D	c	1

Fig(a): The format of the data file

	A	B	C	D
A		4	4	3
B	4		3	3
C	4	3		3
D	3	3	3	

Fig(b): The total number of movies watched by the given pair of users

	A	B	C	D
A		2	1	1
B	2		1	1
C	1	1		0
D	1	1	0	

Fig(c): Total number of common movies where the two users rate the same

	A	B	C	D
A		0.5	0.25	0.33
B	0.5		0.33	0.33
C	0.25	0.33		0
D	0.33	0.33	0	

Fig(d): pairwise similarity

Two datasets [1][2] with different sizes are provided by MovieLens. The small dataset contains 600 users, 9000 movies and 100,000 ratings while the large one contains around 280,000 users, 58,000 movies and 27,000,000 ratings. Each user is represented by its unique userID and each movie is represented by its unique movieID. The format of the data set is as follows:

<userID>, <movieID>, <rating>

You are required to detect the TOP K similar users for each user. An example of the output format (K=2) should be (different similar users separated by space):

A: B D
B: A D (or C)
C: B A
D: A B

To facilitate your code development/design, you can take the following two-step approach:

a. **[30 marks]** For each pair of users in the datasets of [1] and [2], use one or more MapReduce job(s) to output the number of movies they have both rated with the same score.

   For your homework submission, you only need to submit i) your MapReduce codes and ii) The list of the 10 user-pairs which have the largest number of same-scored movies within the corresponding dataset. The format of your answer should be as follows:

   <userID A, userID B>: <the number of same-scored movies> // top1
   …
   <userID X, userID Y>: <the number of same-scored movies> // top10

b. **[30 marks]** By modifying/ extending part of your codes in part (a), find the Top-K (K=3) most similar users (as defined by Equation (**)) for every user in the datasets [1] and [2].
   For your homework, you only need to submit your MapReduce codes together with the similar user lists for
   i) the users in [1] whose ID shares the same last 2 digits of your CUHK student ID number. For example, if your CUHK student ID number is ****13, you will need to output the Top-3 similar user list for Users 013, 113, 213, 313, …., 513 (if such users exist in the small dataset [1]), i.e.

   <013>: <similar userID 1>, …, <similar userID K>
   <113>: <similar userID 1>, ..., <similar userID K>
   …
   <513>: <similar userID 1>, ..., <similar userID K>

ii) Similarly, for the dataset in [2], only output the Top-3 similar user list for the users whose ID shares the same last 4 digits of your CUHK student ID number.

(Hint: In part (b), to facilitate the computation of the similarity measure as defined in (**), you can use the inclusion-exclusion principle, i.e. $|S \cup T| = |S| + |T| - |S \cap T|$ **)**

c. **[40 marks]** Run part (a) for the **large** dataset multiple times while modifying the number of mappers and reducers for your MapReduce job(s) each time. You need to examine and report the performance of your program for at least **3** different runs. Each run should use a different combination of number of mappers and reducers. For each run, performance statistics to be reported should include: (i) the time consumed by the entire MapReduce job(s) and the maximum, minimum and average time consumed by (ii) mapper tasks and (iii) Tabulate the time consumption for each MapReduce job and its tasks. One example is given in the following table. Explain your observations.

Maximum mapper time	Minimum mapper time	Average mapper time	Maximum reducer time	Minimum reducer time	Average reducer time	Total job
60s	40s	50s	60s	40s	50s	2.5 min

d. **[Bonus 20 marks] One-click Hadoop-Cluster/Application deployment using Amazon EMR/ Google Cloud DataProc**

The AWS CLI[10]/ Google Cloud API[11] is a tool that provides command line interface to Amazon EC2/ Google Compute Engine. You can use this tool to write scripts to perform a series of recurring tasks and then repeat the entire process via one-click deployment, e.g. Write a script to i) Setup a multi-node Hadoop cluster, ii) deploy it and iii) use it to run a specific MapReduce job. In this part, you need to write a script using the AWS CLI/ Google Cloud API to setup and launch a multi-node Hadoop cluster (4VMs) in Amazon EMR[12]/ Google Cloud DataProc[13] and then run the Wordcount program to process the Shakespeare dataset[8]. Following command may be useful:

```
$ aws emr create-cluster --name “example-cluster”\
      --release-label emr-5.26.0 \
      --instance-type t2.large \
      --instance-count 4 \
       --applications Name=Hadoop
```

```
            //create a multi-node hadoop cluster in Amazon EMR,
where HADOOP_PATH is in the directory /home/hadoop

$ gcloud dataproc clusters create example-cluster \
      --master-machine-type n1-standard-2 \
      --num-workers 3 \
      --worker-machine-type n1-standard-2
            //create a multi-node hadoop cluster in google dataproc
$ gcloud dataproc jobs submit hadoop --cluster example-cluster \
      --jar wordcount.jar  input/*  output
            //submit a hadoop wordcount program to Google Dataproc
```

Submission requirement:

Please submit the script and output of the WordCount program **in one single PDF file.**

Hints:

1. Solve the above problem using one or more MapReduce jobs. The design of your program should be scalable enough to handle the two datasets. It may be more difficult to just use one MapReduce program to get the final results of the above problem due to the memory exhausting problem. You can either use the IE DIC or the Hadoop cluster you built for HW#0 to run your MapReduce program(s). You are free to use any programming language of choice (e.g., Java [4] or Python [5] or C/C++) to implement the required MapReduce components, i.e., mapper(s), reducer(s), etc. (e.g. by leveraging the "Hadoop Streaming" capability [6]).
2. For students who cannot setup the Hadoop cluster in HW#0, please contact the TAs. You can either choose to set up a hadoop cluster with TAs' help or you can use the IE DIC Cluster account to run MapReduce program. Hadoop has already been installed in IE DIC cluster. Please note that the Hadoop in IE DIC cluster is based on Hortonworks, and it is a little different from the standard Hadoop system. For more information, please refer to [9]. Hadoop is well installed in the IE DIC Cluster and you can login the cluster to submit jobs via the following command:

   ssh student_id@dic14.ie.cuhk.edu.hk

   where student_id is your student ID number. You can find the password on the grading board of the elearning system. Note that this machine can only be accessed within IE network. For those who are from other departments, your temporary IE account will be distributed on the elearning system. You may need to install IE VPN to get access to the IE network with your IE account.
3. If you use Java, you can specify the number of mapper with the following code: `job.setNumMapTasks(20).` If you use Hadoop streaming with Python, you can specify

it via the following command option: `-D mapred.map.tasks=20`. If this does not work, you may need to modify the split size in the `$hadoop/etc/hadoop/mapred-site.xml: mapred.min.split.size =268435456(256M).`

4. As for the large dataset, you may want to set mapreduce.map.output.compress=true to compress the intermediate results, in case you don't have enough local hard disk space (to hold the intermediate tuples).
5. The large dataset may take a long time to process even if everything is correct. **You should start doing this homework as early as possible**!

References:

[1] Small scale dataset
https://www.dropbox.com/s/qv939szcdn2nzks/small_dataset.csv?dl=0
[2] Large scale dataset
https://www.dropbox.com/s/0kgqyw1cng0whwc/large_dataset.csv?dl=0
[3] How many Mappers and Reducers?
https://wiki.apache.org/hadoop/HowManyMapsAndReduces
[4] Write a Hadoop program in Java
https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
[5] Write a Hadoop program in Python
http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/
[6] Hadoop Streaming
https://hadoop.apache.org/docs/r1.2.1/streaming.html
[7] Composite Key
http://tutorials.techmytalk.com/2014/11/14/mapreduce-composite-key-operation-part2/
[8] Secondary Sort
http://codingjunkie.net/secondary-sort/
[9] Hortonworks
https://hortonworks.com/
[10] AWS CLI
https://aws.amazon.com/cli/?nc1=h_ls
[11] Google Cloud API
https://cloud.google.com/sdk/gcloud/
[12] AWS EMR
https://aws.amazon.com/emr/?nc1=h_ls
[13] Google Cloud Dataproc
https://cloud.google.com/dataproc/