IERG4330/ESTR4316/IEMS5730 Spring 2022 Homework #1 (Similar Users Detection)

Release date: Jan 28, 2022

Due date: Feb 16, 2022 (Wed) 11:59:00 p.m. (i.e. midnight)

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.
- Only students who have not taken IERG4300/ESTR4300/IEMS5709 are required to submit this assignment.

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]: Similar Users Detection in the MovieLens Dataset

Similar user detection has drawn lots of attention in the machine learning field which is aimed at grouping users with similar interests, behaviors, 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 users, we need to calculate the similarity between each user pair. In this homework, the similarity between a given pair of users (e.g. A and B) is measured as **the total number of movies both A and B have watched** divided by **the total number of movies watched by either A or B.** The following is the formal definition of similarity: Let M(A) be the set of all the movies user A has watched. Then the similarity between user A and user B is defined as:

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

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
A	a
A	b
A	c
B	b
B	c
B	d
C	b
C	d
D	b
D	c

Fig(a): The format of the data file.

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

Fig(b): The total number of movies both users in the pair have watched.

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

Fig(c): The total number of movies watched by either one of the users in the pair.

	A	B	C	D
A		0.5	0.25	0.66
B	0.5		0.66	0.66
C	0.25	0.66		0.33
D	0.66	0.66	0.33	

Fig(d): Pairwise similarity

Two datasets [1][2] with different sizes are provided by MovieLens. The small dataset contains 600 users and 9000 movies while the large one contains around 69,878 users and 10,677 movies. 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>

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 are separated by a space):

A: D B
B: C D
C: B D
D: A B

We are going to solve the above similar users detection problem using MapReduce. 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 languages (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]). Again, the design of your program should be scalable enough to handle both datasets [1][2].

To facilitate the code development/design, we 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 watched.

   For your homework submission, you only need to submit i) your MapReduce codes and ii) The list of the 10 pairs of users having the largest number of movies watched

by both users in the pair within the corresponding dataset. The format of your answer should be as follows:

<userID A>, <userID B>: <the number of movie both A and B have watched> // top1

…

<userID X>, <userID Y>: <the number of movies both X and Y have watched> // top10

b. **[40 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]. If multiple users have the same similarity, you can just pick any three of them.

Hint:

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

2. **You are allowed to use the "sort" command on Linux to get the top K similar users after you have computed similarity scores for all pairs.**

In your submission, you only need to submit your MapReduce codes together with the following output:

i) **[25 marks]** 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 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) **[15 marks]** 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: To reduce the memory consumption of your program, you may consider using the composite key design pattern and secondary sorting techniques as discussed in [7] and [8]).

c. **[30 marks]** Run part (a) for the **large** dataset [2] 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 the 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

Submission requirement:

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

Hints:

1. For students who cannot set up 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. Once the IE DIC cluster is ready, TAs will inform you of the account information.
2. 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).`
3. 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).
4. 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://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/movielens_small.csv
[2] Large scale dataset
https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/movielens_large_updated.csv
[3] How many Mappers and Reducers?
https://cwiki.apache.org/confluence/display/HADOOP2/HowManyMapsAndReduces
[4] Write a Hadoop program in Java
https://hadoop.apache.org/docs/r2.9.2/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/r2.9.2/hadoop-streaming/HadoopStreaming.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/