IERG4300 / ESTR4300 Fall 2024 Homework #1

Release date: Sept 30, 2024

Due date: Oct 19, 2024 (Saturday) 11:59pm

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/.

Name_____ SID_____

Date_____ Signature _____

Submission notice:

- Submit your homework via the elearning system (CUHK Blackboard).

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. Moreover, you are **NOT** allowed to use any LLMs (e.g., ChatGPT, Claude etc.) in this course. Anyone who uses LLMs for completing the homework will be treated as cheating.

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]: Similarity Detection for Websites

Problem Description:

In recent years, detecting similarity among websites has become one of the most prevalent computational tasks, useful for community detection or recommending websites. Nowadays, a wealth of valuable datasets can be found online, which can be leveraged to develop algorithms for similarity detection. For this assignment, you will implement a similarity detection algorithm across three datasets focusing on websites. Here, similarity between websites is measured based on the number of common hyperlinks they share on their pages.

Within a website network, there are two types of relationships. One is `symmetric`, i.e., website X has embedded a hyperlink to website Y, and website Y also provides a hyperlink directing back to website X; the other is `asymmetric`, which means website Y may not link to website X even though website X has a hyperlink to website Y. In the latter case, there are two roles involved in this relationship: `referrer` and `referree` (akin to relationships seen on Wikipedia). When website X has a hyperlink to Y, X is considered as the referrer and Y is the referree.

The similarity between a given pair of websites is calculated by the number of ***common referrees*** divided by the ***total*** (deduplicated) number of the two websites' ***referrees***. Specifically, the formal definition of `similarity` is as follows:

If $|\mathbf{out}(A) \cup \mathbf{out}(B)|$ >0, we define

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

where `out(A)` is the set of all referrees of website A, and |A| is the cardinality of A.

If $|\mathbf{out}(A) \cup \mathbf{out}(B)| = 0$, we set the similarity to 0.

Motivating Example:

The following example demonstrates the process of calculating pairwise similarity among five websites: A, B, C, D, and E. A matrix can be used to illustrate the relationships between these websites. For instance, an element with a value of 1 in the cell corresponding to (A, B) indicates that website A contains a hyperlink to website B. In this case, A is the referrer, and B is the referree. Conversely, an element with a value of 0 signifies that no hyperlink exists from A to B.

Referer \ Referee	A	B	C	D	E
A	0	1	1	0	1
B	1	0	1	0	1
C	0	0	0	0	1
D	1	0	0	0	0
E	0	0	1	0	0

According to the matrix above, the set of referrees of A is {B, C, E} and the set of referrees of B is {A, C, E}. There are 2 common referrees between A and B (i.e., C and E), and the number of the union of their referrees is 4 (A, B, C, E). The similarity between A and B is therefore 2/4 = 0.5.

Dataset:

We provide three datasets with different sizes. The small dataset contains around 4K websites; the medium one contains around 80K websites, and the large one with over 100K websites. Each website is represented by its unique ID number (integer). The download links of all datasets are listed in reference [1]-[3]. The small dataset is provided to facilitate your initial debugging and testing. The design of your program should be *scalable* enough to handle all the datasets.

Sample Input:

The format of the data file is arranged as a referrer-referree pair. As in the above example, it is as follows:

A B
A C
A E
B A
B C
…

Sample Output:

We anticipate the similarity detection results to identify the TOP K most similar websites for EACH individual website. An example of the output format (K=3) should be as follows:

A: B C E
B: A D E
…

Note: For each individual website,

1. different similar websites are separated by space;
2. retain all website ids with identical similarities, but in the event of a tie, pick those websites with *smaller* ids to ensure the total count does not exceed K;
3. websites with 0 similarity are omitted.

Objective:

Solve the above similarity detection problem using **`MapReduce`**. Four tasks (plus one bonus task) with instructions can be found below. Note that the output format of each specific task may vary.

[Task Requirements]

1. You can either use the IE DIC or the Hadoop cluster you built for HW#0 to run your MapReduce program(s).
2. You are free to use any programming languages (e.g., Java [4], Python [5] or C/C++) to implement the required MapReduce components, including mapper(s), reducer(s), etc. (e.g. by leveraging the "Hadoop Streaming" capability [6]).
3. Again, the design of your program should be scalable enough to handle all three datasets.

[Submission Requirements]

1. Submit the source codes and outputs of your programs in one single PDF report. Besides, for each key step, you should also present the commands used (if applicable), the descriptions (in words) and illustrations (in figures/ screenshots) that help convey and clarify your ideas in solving the problems. In particular, for each Hadoop job (running a map-reduce function pair), please give a brief description in at most three sentences of its functionality.
2. Package all the source codes (as you included in step 1) into a zip file. Please submit both the PDF report and the zip file to CUHK Blackboard.
3. As for task (a), (b) and (e) below, submit the similarity detection results of all those websites whose IDs share the same last 4 digits with your CUHK student ID. For example, if your student ID is 1155004321, then you need to submit the results for websites with ID = 4321, 14321, 24321, 34321, …, 114321, …

   Note that this requirement only serves as a *filter* for the original outputs. In other words, you still need to run your MapReduce jobs on every website and harvest all outputs. This extra step is to trim the raw outputs (only picking certain rows according to your CUHK ID) and paste into your report.

Tasks.

a. **[25 marks]** For EVERY website, recommend the website with the maximal number of common referrees in the **`medium`**-sized dataset [2]. If multiple websites share the same number, pick the one with the *smallest* ID. Your output should consist of m lines, where m is the total number of websites. Each line follows the format below:

A:B, {C,E}, 2

where "A:B" is the website pair, "{C,E}" is the set of their common referrees, "2" is the count of common referrees.

Note:

1. For the set of common referrees, there is **no** special requirement for the elements' sequence, i.e., both {C,E} and {E,C} are acceptable. The same applies to Q1(b) and Q1(e).

2. To determine the “smallest” ID when there is a tie, IDs are compared as integers rather than strings. E.g., in most programming languages, 987 < 1987 but “987” > “1987”. We assume 987 is the smaller ID.

b. **[30 marks]** Find the TOP K (K=3) most similar websites of EVERY website as well as their common referrees for the **`medium`**-sized dataset [2]. If multiple websites have the same similarity with a particular website, they should all be included in your results. (Still, the total number of records for each website should not exceed K, pick the ones with *smaller* IDs when there’s a tie). For each pair of websites, output a line with the following format:

A:B, {C,E}, simscore

where “simscore” is the similarity score between A and B.

Hints:

1. To facilitate the computation of the similarity as defined in Formula (*) in *Problem Description*, you can use the inclusion-exclusion principle $|S \cup T| = |S| + |T| - |S \cap T|$.
2. You are allowed to use the “sort” command on Linux to get the top K similar websites after you have computed similarity scores for all pairs.

c. **[25 marks]** In fact, each website is annotated with a label indicating its community. In each dataset, a label file is provided, with the first column indicating the website ID and the second column indicating the label value. For example, the small dataset has seven different labels (the value ranges from 0 to 6), which means that each website is from one of the seven communities.

For each community in the **`medium`** dataset, please figure out how many (unique) members act as the common referrees of other websites. (For example, suppose that A, B, C, D, E are labeled with community 0, 1, 2, 1, 2, respectively. Then, for community 0, one of its members (website A) acts as the common referree of others (website B and D). As for community 1, none of its members is the common referree of others.) Your reported results should be formatted like the following example:

Community 0: 1
Community 1: 0
Community 2: 2

d. **[20 marks]** Run part (a) for the **`medium`** 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 4 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); (ii) the maximum, minimum and average time consumed by mapper and reducer tasks; (iii) tabulate the time consumption for each MapReduce job and its tasks. (One example is given in the following table.) Moreover, describe (and explain, if possible) your observations.

Example:

1st Run:

#Job	Mapper num	Reducer num	Max mapper time	Min mapper time	Avg mapper time	Max reducer time	Min reducer time	Avg reducer time	Total time
1	3	2	60s	40s	50s	60s	40s	50s	2.5 min
2	…	…	…	…	…	…	…	…	…

Note:

1. The number of rows in the table depends on the number of jobs (one distinct map and reduce function for each job) you chain to complete part (a).
2. The elapsed time for each Map/Reduce Task can be found on Job History Service Web UI (port 19888).
   For students using their own VMs, start the service by running `./sbin/mr-jobhistory-daemon.sh start historyserver`. Remember to set up proper firewall rules/ use ssh port forwarding so as to access the Web UI on your local browser.
   For students using IE DIC Cluster, the Web UI is served at http://dicvmc2.ie.cuhk.edu.hk:19888/. Moreover, users can find the details of a particular job via e.g. http://dicvmc2.ie.cuhk.edu.hk:19888/jobhistory/job/job_1694578679658_0003, where job_1694578679658_0003 is the ID of the job you created.

e. **[Bonus 20 marks*]** Find the TOP K (K=4) most similar websites and the list of common referrees for each website in the **`large`** dataset in [3] using the format of Q1(b). (Hints: 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].)

*Part (e) is a bonus question for IERG4300 but is required for ESTR4300.

General Hints:

1. Going through HW#0 Q1(c),(d)'s source codes may help you understand how to construct your mapper & reducer code here. However, unlike `WordCount`, you cannot always expect to finish all your work with only one single mapper and reducer code.
   Chaining multiple (i.e. a series of different) MapReduce jobs to handle complex similarity detection problems is a standard approach. It may be difficult to just use one MapReduce program to get the final results for each problem due to memory exhaustion and the parallel & distributed nature of data in Mappers/Reducers.
2. For each dataset, there are two files included. The one with the suffix '`_relation`' indicates the mutual relations of each pair of websites. The one with the suffix '`_label`' indicates the community label for each website.
3. For students who did not manage to set up their Hadoop cluster in HW#0, please contact the TAs. You can either choose to set up the cluster with TAs' help or you can

run your MapReduce programs on the IE DIC Cluster, where Hadoop has already been installed. Once the IE DIC cluster is ready, TAs will inform you of the account information. More details will be provided in the tutorial/ on CUHK blackboard.

4. If you use Java, you can specify the number of mappers 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`. The number of reducers can be modified similarly.
   If this does not work, you may need to modify the split size in `$hadoop/etc/hadoop/mapred-site.xml`: `mapred.min.split.size=268435456`. Refer to [9] for more information.
5. 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).
6. Tackling the problems may take much longer than you expect. Particularly, the large dataset may take a long time to process even if everything is correct. **Please start doing this assignment as early as possible**.

Q2 [20 marks]: Q&As for HW#0 Review

The questions that follow might not have a definitive answer. They are intended to test your understanding and help you review HW#0 from a system design perspective.

a. [10 marks]

1) Which default ports do machines (VMs) in a multi-node Hadoop cluster use for inter-machine communications (i.e., transmission of network traffic between machines in the cluster)? Name at least 2 ports and describe their roles.
2) Are you using public or private IPs of the VMs to access SSH? Are machines in your Hadoop cluster using public or private IPs to identify and communicate with each other?
3) To ensure proper communication between machines, did you set up extra firewall rules/policies as you did for SSH (port 22) in HW#0? Why or why not?

b. [10 marks] Consider a Hadoop cluster with the following configurations:

1. You have allocated 100GB of disk space for each VM in your Google Cloud/AWS Console.
2. There are at most 4 such VMs that can be utilized by the Hadoop cluster.

Given this setup, please evaluate the feasibility of taking up 150GB of total disk space on the HDFS (Hadoop Distributed File System).

Please list your considerations point by point. You may first give a general answer and then take into account all potential factors that might affect the usage of disk space.

References:

[1] Small-scale dataset
http://mobitec.ie.cuhk.edu.hk/ierg4300Fall2024/static_files/homework/small.tar.gz
[2] Medium-scale dataset
http://mobitec.ie.cuhk.edu.hk/ierg4300Fall2024/static_files/homework/medium.tar.gz
[3] Large-scale dataset
http://mobitec.ie.cuhk.edu.hk/ierg4300Fall2024/static_files/homework/large.tar.gz
[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 Python2
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
https://techmytalk.com/2014/11/14/mapreduce-composite-key-operation-part2/
[8] Secondary Sort
http://codingjunkie.net/secondary-sort/
[9] How many Mappers and Reducers?
https://cwiki.apache.org/confluence/display/HADOOP2/HowManyMapsAndReduces