Seoul National University

M1522.001400 Introduction to Data Mining

Spring 2019, Kang

Homework 1: MapReduce (Chapter 2)

Due: Mar 28, 23:59 AM

Reminders

- The points of this homework add up to 100.
- Like all homeworks, this has to be done individually.
- Lead T.A.: Bonhun Koo (darkgs@snu.ac.kr)
- Please submit one zip file to eTL. The zip file should include a report (about 2 pages in pdf), scripts (*.sh), and source files (*.py). Also, the zip file's name should be formed in "HW1_StudentID.zip" (e.g., HW1_2015-12345.zip).
- If you want to use slipdays or consider late submission with penalties, please note that you are allowed one week to submit your assignment after the due date. That is, after **April 4**, we will NOT receive your submission for this assignment.
- You cannot use any third-party libraries except Apache Hadoop for this assignment.

1 Introduction

In this assignment, you will implement 2 MapReduce programs to simply analyze the social network on Apache Hadoop, most famous open-source implementation of MapReduce. **We strongly recommend using Linux (Ubuntu 16.04) system for this assignment because it is difficult to run Hadoop program on Windows.** To complete this assignment, you need to understand how a social network is analyzed, program specification and how to test and submit your assignment. Please read the following sections carefully.

2 Homework archive structure

The homework archive 'hw1.zip' contains of the following files:

- `run_docker.sh` – script to run docker image
- `run_*.sh` – scripts to run each task
- `wordcount/*` - wordcount example
- results/* - sample outputs to check your codes
- `tf/* - skeleton codes for Top-50 follower counting`
- `nsf/* - skeleton codes for non-symmetric fellowship`
- You need to fill your code in Python source code files in the `{problem}/bin` directory. There is an example MapReduce program in the "wordcount" directory. The example shows how to implement MapReduce program.

3 How to make pydoop environment

In this assignment, you need to set-up your own pydoop environment with Hadoop. For the convenience, we recommend you run your program on docker image with ubuntu 16.04 (on the virtual box or anything is fine).

- How to install docker

  https://docs.docker.com/install/linux/docker-ce/ubuntu/

  follow the instructions until "$ sudo docker run hello-world"

- How to use pydoop docker image

  https://crs4.github.io/pydoop/installation.html

  just "docker pull crs4/pydoop" is okay

Then, download and extract our homework skeleton file from the eTL (hw1.zip). Try "./run_wordcount.sh" to check your environment.

4 How to grade your assignment

1. Grade of your assignment consists of four parts: 1) 'source code' (10 pts), 2) 'program execution' (60 pts), 3) 'report' (20 pts), and 4) 'submission' (10 pts)

2. We made a grading machine for the 'program execution' part. The machine will run your MapReduce programs, and verify your outputs that your programs generate for given inputs. If your program cannot generate correct answers for an input file, it will not give you're the point corresponding to the input. Our machine will consider the following scenarios:

   - (**Accept**) When your MapReduce program generates exact outputs for an input file, the machine will give you the point of the input.
   - (**Wrong Answer**) When your MapReduce program runs normally, but generates incorrect outputs for an input file, including typos, the machine will not give you the point of the input.
   - (**Run Error**) When your MapReduce program does not run, or is terminated suddenly for some reasons, the machine will not give you the point of an input file because it cannot generate any outputs.
   - (**Time Limit**) When your MapReduce program runs over a predefined execution time for an input file, our machine will stop your program, and it will not give you the point of the input. The time limit of each execution is ***5 minutes***.

3. We will generate 5 input files for each program (totally 10 input files) and assign 6 points for each input file. For example, if your programs get 5 accepts, 3 wrong answer, and 2 run errors, the points for 'program execution' part will be 30 points. Hence, before submitting your assignment, please be sure that your program makes correct answers in reasonable time for any input case.

5 Simple social network analysis

In a social network, we can represent a relationship between people on a directed graph. On a given graph, A node represents a person, and an edge represents a fellowship between two people.

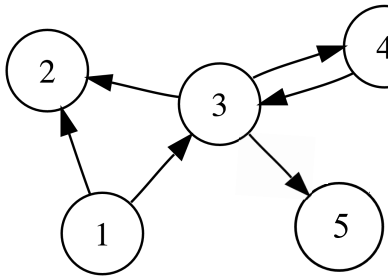

Figure 1. Example directed graph

5.1 Top-50 Follower counting problem

The top-50 follower counting problem is to count the number of followers of each node in a given graph, and then find the top 50 people having the largest number of followers. The number of followers of a person (node) is the number of edges incident to the node. For example, consider person (node) 3 of the directed graph in Figure 1. Since the person 3 has two incoming edges, the number of followers of the person 3 is 2. Note that the number of followers can be zero. For instance, node 1 has no followers.

In this assignment, you need to implement MapReduce algorithms, called ***TopFollowersCounter***, to count the number of followers for each node and find the 50 people which have the largest number of followers.

5.2 Non-symmetric fellow-ship problem

The non-symmetric fellow-ship problem is to find the non-symmetric fellowship between two people in a social network. Non-symmetric fellowship means that a *person a* follows a *person b*, but the *person b* does not follow the *person a*. For example, consider nodes 1 and 2 of the directed graph in Figure 1. Since the person 1 follows the person 2 but the person 2 does not follow the person 1, the relationship between the nodes 1 and 2 is a non-symmetric.

In this assignment, you need to implement MapReduce algorithms, called ***NonSymmetricFellowship,*** to find the all of the non-symmetric fellowships in a given graph.

6 Program Specification

In this assignment, you will implement 2 MapReduce programs.

- `tf/*` – a MapReduce program to find the top 50 people having the largest number of followers
- `nsf/*` – a MapReduce program to find the all of the non-symmetric fellowships

All the programs above should contain your student ID as the name of MapReduce program. We mark where the student ID should be in the source code. Please put your student ID into the source code.

Input specification: each line represents fellowships. A source person index follows a destination person index. The first integer is a source node index and the second one is a destination node index. A source node index follows a destination node index. One line consists of two integers which are tab-separated.

- Ex) **"**0	1" indicates that the person 0 follows the person 1.

Output specification: each line represents a person (node) and the number of followers of a person which are tab-separated. Note that the program should print the top 50 people in descending order based on the number of followers.

- Ex) "3	4" indicates that person 3 has 4 followers.

Sample Input – (Source, Destination)	
0	3
1	3
1	4
2	1
2	3
2	4
4	3

Sample Output – (person, number of followers)	
3	4
4	2
1	1

Problem 2: Implement a MapReduce program to find the non-symmetric fellowship in '*nonSymmetricFellowship.java*'. You may create a Mapper class and a Reducer class.

Input specification: A source person index follows a destination person index. (This is the same as the input specification of Problem 1)

- Ex) "0	1" indicates that the person 0 follows the person 1.

Output specification: each line represents two people which are tab-separated. Note that the "0	1" output will exist when there is "0	1" but there is no "1	0" in a given data. Also, an index of left node should be smaller than an index of right node regardless of direction of an edge when you print. For example, there is "7	3" but there is no "3	7" in a given data. Then, you should be print "3	7".

- Ex) "1	2" indicates that the *person 1* and the *person 2 have the non-symmetric fellowship*.

Sample Input – (Source, Destination)	
0	3
1	3
1	4
2	1
2	3
2	4
3	0
3	1
3	2
4	3

Sample Output – (person, person)	
1	4
1	2
2	4
3	4

7 How to test your program

We strongly recommend testing and verifying your program before submitting the assignment. You can test your MapReduce programs by using 'run_*.sh' scripts.

We will test your program with following test process

```
unzip {your_submitted_zip}
cd {unzipped folder}
for i in range(5):
    replace input file in “tf/input”
    ./run_tf.sh
    Check result in “results/result_tf.txt”

for i in range(5):
    replace input file in “nsf/input”
    ./run_nsf.sh
    Check result in “results/result_nsf.txt”
```

You can modify skeleton code if you want. Please make sure that after the aforementioned test process, your program generates a results file to the proper path from an input file in the proper path.

8 How to submit your assignment

After finishing your assignment, you must submit an archive file (in zip format) file to eTL (http://etl.snu.ac.kr) with complying the followings:

1. The archive file should contain all of files needed to execute your program.
2. The name of the archive file should observe the following form: “HW1_StudentID.zip”
3. Your programs should contain your student ID as the name of MapReduce program.

If you don’t follow these submission rules, you may lose some points in ‘submission’ part.

Your submitted archive file should contain a 2-page report about your homework in pdf format. Please include a description of your map reduce function implementation and include a description of the execution process on the Hadoop distributed system.