Seoul National University

M1522.001400 Introduction to Data Mining

Spring 2016, Kang

Homework 1: MapReduce (Chapter 2)

Due: Mar 23, 09:30 AM

Reminders

- The points of this homework add up to 100.
- Like all homeworks, this has to be done individually.
- Lead T.A.: Jinhong Jung (montecast9@gmail.com)
- Please submit one zip file to eTL. The zip file should include the report (in pdf), Makefile, README, jar files (*.jar), and source files (*.java). Also, the zip file's name should be formed in "HW1_StudentID.zip" (e.g., HW1_2015-12345.zip).
- Please turn in the hardcopy of your report to T.A. at the classroom on Mar 23.
- 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 Mar 30, we will NOT receive your submission for this assignment.

1. Introduction

In this assignment, you need to perform the following tasks:

1) Implement degree counting algorithms on MapReduce framework
2) Write a report with answers for the questions in Section 5

To carry out this assignment, you first need to understand the concept of degree counting and install Apache Hadoop, which is an open-source implementation for MapReduce framework. Please read Sections 3 and 4 to understand what the degree counting is and how to install Hadoop.

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

1) The zip file must include the following files: the report (in pdf), Makefile, README, IndegreeCounter.jar, OutDegreeCounter.jar, DegreeDistribution.jar, and source files (*.java). The information of the 'README' file should include your student id, name, a brief description about how to execute your implementations.

2) The name of the zip file should observe the following form: "HW1_StudentID.zip" (e.g., HW1_2015-12345.zip).

Also, you should hand in the hardcopy of your report to T.A. before starting the lecture on Mar 23. Both copies of your report must be identical.

2. Files included in this homework package

The homework package, 'hw1.zip', contains the following files[1]:

- `example/Makefile:` Makefile for the 'WordCount' example
- `example/WordCount.java:` source code for the 'WordCount' example on MapReduce framework
- `example/input/file01.txt, file02.txt`: input files for the 'WordCount' example
- `plot/Makefile`: Makefile for the Gnuplot script – 'degreedist.plt'
- `plot/degreedist.plt`: Gnuplot script which draw a plot for 'data.txt'
- `plot/data.txt`: example degree distribution data
- README: document about Hadoop installation and how to plot your result
- `problem.edge`: example data used in this assignment
- `[*] Makefile`: Makefile for this assignment
- `[*] IndegreeCounter.java`: source code for computing the in-degree for each node
- `[*] OutDegreeCounter.java`: source code for computing the out-degree for each node
- `[*] DegreeDistribution.java`: source code for computing the degree distribution

[1] `[*]` indicates files you will need to complete

3. Degree counting problem

In a graph, the degree of a node is the number of edges incident to the node. If the graph is directed, meaning that edges point in one direction from one node to another node, then nodes have two different degrees as follows:

- In-degree: the number of incoming edges
- Out-degree: the number of outgoing edges

For example, consider node 3 of the directed graph in Figure 1. Since the node has two incoming edges and three outgoing edges, the in-degree is 2 and the out-degree is 3.

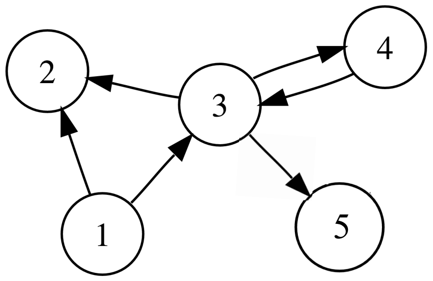

Figure 1. Example directed graph

Note that either an in-degree or an out-degree can be zero. For instance, the in-degree of node 1 is 0 while its out-degree is 2.

The degree counting problem is to count the degree of each node in a given graph. In this assignment, since we handle directed graphs, you need to implement MapReduce algorithms, called ***IndegreeCounter*** and ***OutdegreeCounter***, to count the in-degree and the out-degree for each node.

Among various graph analysis techniques, one way is to draw the degree distribution of graphs. ***The degree distribution*** $P(k)$ is defined by the number of nodes with degree k. In this assignment, you need to implement a MapReduce algorithm, called ***DegreeDistribution***, to compute the degree distribution. See Question 4 in Section 5 for an example and plots of degree distributions.

4. Hadoop installation

To run and test your implementation, you first need to install Hadoop on a single machine (node). Of course, Hadoop is usually performed on distributed systems (or a cluster), but we assume that you do not have a cluster. Fortunately, implementing, running and testing Hadoop programs are also possible on a single machine.

In this assignment, you must use Hadoop 2.7.2, which is the latest stable release. Also, you should exploit the '*org.apache.hadoop.mapreduce*' package, which provides useful APIs such as *Mapper* and *Reducer*, in your source codes. You can download the binary files from the following link:

- http://apache.tt.co.kr/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz

We already described the detailed information about how to install Hadoop in the *'README'* file of the homework package. Please read the file carefully, and we recommend using a Linux system. If you need another information for Hadoop installation on a single machine, please refer to the following link:

- http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/SingleCluster.html

To check if Hadoop works well, you can use the example WordCount code, which counts the frequency of each word in a document. The WordCount code (*WordCount.java*) is located in the *'example'* folder of the homework package. Please read the *'README'* file to know how to compile and execute the example. The code can be used as a skeleton code for implementing the degree counting algorithms.

5. Questions

Question 1: Implement the in-degree counting algorithm in '*IndegreeCounter.java*'. Make sure that when we type '*make in*', Hadoop should execute your program, and print the in-degree of each node in the graph, given by the '*problem.edge*' file included in the homework package. Report the screenshot after printing the in-degree of each node. (To look the content of files in HDFS, use the *'hadoop fs –cat'* command.) [25 points]

Input specification: each line represents an edge of a directed graph in the file. One line consists of two integers which are tab-separated. The first integer is a source node index and the second one is a destination node index**.**

- Ex) **"**0	1" indicates the edge from node 0 to node 1.

Output specification: each line represents a node and its in-degree which are tab-separated. Note that if a node has zero in-degree, the program should not print the line corresponding to the node.

- Ex) "1	2" indicates that node 1 has 2 in-degrees.

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

Sample Output – (Node, In-Degree)	
1	2
2	2
3	3
4	2

Question 2: Implement the degree counting algorithm for counting out-degree in '*OutdegreeCounter.java*'. Make sure that when we type 'make out', Hadoop should execute your program, and the program should print the out-degree of each node in the graph, which is represented in the '*problem.edge*' file. Report the screenshot after printing the out-degree of each node. [25 points]

Input specification: each line represents an edge of a directed graph in the file. (This is the same as the input specification of Question 1)

- Ex) **"**0   1" indicates the edge from node 0 to node 1.

Output specification: each line represents a node and its out-degree which are tab-separated. Note that if a node has zero out-degree, the program should not print the line corresponding to the node.

- Ex) "1   2" indicates that node 1 has 2 out-degrees.

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

Sample Output – (Node, Out-Degree)	
0	1
1	3
2	3
4	2

Question 3: Implement a MapReduce code for computing the degree distribution in '*DegreeDistribution.java'*. Make sure that the following commands perform the corresponding tasks for the graph in the '*problem.edge*' file:

- '*make in_dist*': compute the in-degree distribution on Hadoop, and print the in-degree distribution.
- '*make out_dist*': compute the out-degree distribution on Hadoop, and print the out-degree distribution.

Report the screenshots after printing the in-degree and the out-degree distributions. [25 points]

Input specification: each line represents a node and its degree which are tab-separated. If you compute the in-degree distribution, the degree should be an in-degree. If you compute the out-degree distribution, the degree should be an out-degree. Note that the input file is generated from either '*IndegreeCounter.java*' (Question 1) or '*OutdegreeCounter.java*' (Question 2).

- Ex) **"**1 2" indicates node 1 has 2 (in or out) degrees.

Output specification: each line represents a degree and its count (frequency) which are tab-separated. As like the input specification, the degree is either an in-degree or an out-degree according to the distribution you should compute.

- Ex) "3 2" indicates the number of nodes with the (in or out) degree 3 is 2.

Sample Input – Degree Information (Node Id, Degree)	
0	1
1	4
2	3
3	3
4	3

Sample Output – Degree Distribution (Degree, Count)	
1	1
3	3
4	1

Question 4: Using your implementations, compute the in-degree and the out-degree distributions of the *LiveJournal* dataset, which is one of real-world graphs. Report the plots of both degree distributions for the graph in log-log scale. [25 points]

Data: you can download the LiveJournal dataset from the following link:

- https://snap.stanford.edu/data/soc-LiveJournal1.txt.gz

Plot: For drawing the distributions, you can use any tools such as Excel, Matlab, or Gnuplot, but we strongly recommend using Gnuplot. We already attached a sample Gnuplot script in the '*plot*' folder of the homework package. Please read the '*README*' file to see the usage. Figure 2 depicts in-degree and out-degree distributions of an example dataset (Twitter).

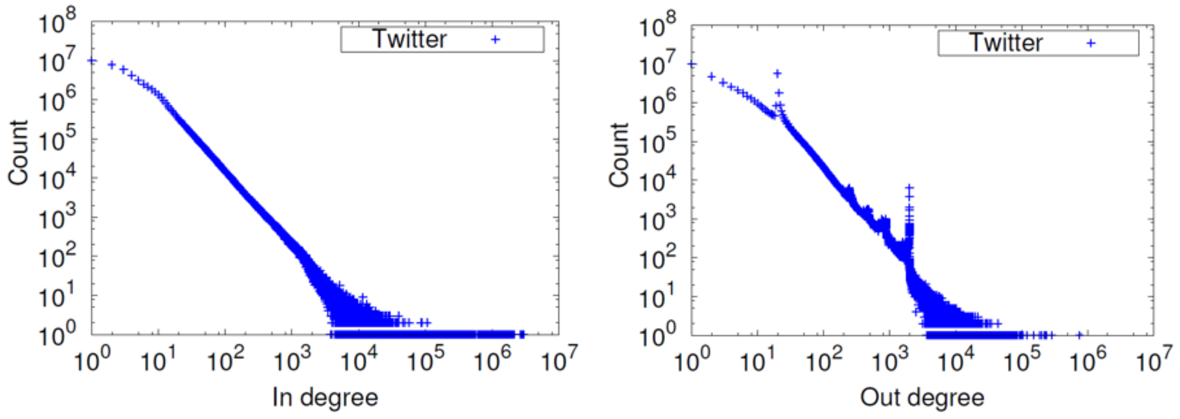

Figure 2. In-degree and out-degree distributions of an example dataset (Twitter). The x-axis indicates $\log(degree)$ *and the y-axis indicates* $\log(the\ number\ of\ nodes\ with\ such\ degree)$*.*