

CES Data Scientist - PageRank Exam

Youcef KACER (youcef.kacer@telecom-paristech.fr)

3 octobre 2016

Exercise 1 : We expose here after the system of linear equations that solves PageRank scores associated to graph G (r_i being PageRank score for page i) :

$$\begin{aligned}r_1 &= r_4 \\r_2 &= 1/2 * r_1 \\r_3 &= 1/2 * r_1 + r_2 \\r_4 &= r_3 \\1 &= r_1 + r_2 + r_3 + r_4\end{aligned}$$

We can solve manually the system :

$$\begin{aligned}r_1 &= r_3 \\r_1 &= r_4 \\r_2 &= 1/2 * r_1 \\1 &= 7/2 * r_1\end{aligned}$$

$$\begin{aligned}r_1 &= 2/7 \\r_2 &= 1/7 \\r_3 &= 2/7 \\r_4 &= 2/7\end{aligned}$$

Otherwise, we can solve the system using previous PageRank lab using MapReduce. We just need to adapt *edge_list.txt* to the graph G . In this file, each line represents a page indice and the page indices it links to :

```
1 2 3
2 3
3 4
4 1
```

We have run our python map/reduce implementation of PageRank. We use a teleport coefficient of 0.75, and an error criteria of 0.01. Figure 1 shows corresponding standard output with PageRank scores results at the end.

The screenshot shows a VirtualBox window titled 'Debian Hadoop [En fonction] - Oracle VM VirtualBox'. Inside the window, a terminal window is open with the prompt 'user@hadoop: ~/TP PageRank'. The terminal displays the execution of a Python script 'PageRankDriver.py' using 'W ignore' as an argument. The script is running a map/reduce job to calculate PageRank scores for a graph G . The output shows several iterations of the algorithm, with messages like 'DFSInputStream has been closed already', 'File option is deprecated, please use generic option -files instead.', and 'Jobconf option is deprecated, please use -D instead.'. The final output shows the PageRank scores for the nodes in the graph, such as 'r1: 0.14301882207', 'r2: 0.27828498223', 'r3: 0.27954185989', and 'r4: 0.287047271807'.

FIGURE 1 – Python map/reduce for PageRank scores associated to graph G

Exercise 2 : Suppose we have a file containing integers with their indices as follow :

```
i n
0 256
1 35
2 4122
3 96
...
```

We present here after a pseudo-code for map/reduce function to find maximum value of this list of integers :

```
map(key=i,value=n)
{
    return 0,n
}

reduce(key,values) // values is an array containing all the integers because
{
    // map has emitted each of them with the same key (0).
    MAX = MIN_POSSIBLE_VALUE
    for n in values
    {
        if n>max
        {
            max = n
        }
    }
    return max
}
```

Exercise 3 : The keyword *yield* is very useful to read data that are too big to fit in memory (RAM). Thus, we read those data directly from hard disk piece by piece. For example, one can read a big .csv file line by line (or lines block by lines block). So, not all the data are in memory but only a piece at once. This is well suitable in Map to provide large quantity of pairs (key,value), one by one into main memory of nodes.

Exercise 4 : 1. **True.**

You can make parallel each element computation of vector result. We show here after a Map/Reduce implementation for multiplying matrix $M_{i,j}$ (i being row indice, j column indice) by vector V_j

```
map(key=(i,Vj),value=Mij)
{
    yield i,Mij*Vj
}
reduce(key=k,value=v)
{
    sum = 0
    for p in v
```

```
{
    sum = sum + p
}
yield k,sum
}
```

2. **False.**

At each iteration, communication between nodes takes time, so too much iterations make the whole process slow.

3. **True.**

The number of iterations needed for PageRank computation is 100, even smaller (10 as we experimented in our wikipedia lab).

4. **False.**

5. **False.**

Map/Reduce works efficiently in batch mode, not in streaming mode.