CSCI 5702/7702– Fall 2019 – Assignment 3

PageRank

Due: 11/18/2019 – 8:59 pm

- Please note that this assignment must be done individually. Your code will be checked against other submissions and other existing resources (such as websites and books) using automatic tools.
- Review the lecture notes on PageRank before starting with this Assignment. Then, fully read this document before starting with the implementation or thinking on the solution.
- If you have technical issues with Spark, please Google the error messages and share the error message alongside the solution that got it fixed on Microsoft Teams, as your classmates may run into the same issues.
- Check Canvas regularly for possible clarifications and updates.

In this problem, you will learn how to implement the PageRank algorithm **in Spark**. You will be experimenting with a small randomly generated graph (assume graph has no dead-ends) provided in the file graph-full.txt.

There are 100 nodes (n = 100) in the small graph and 1000 nodes (n = 1000) in the full graph, and m = 8192 edges, 1000 of which form a directed cycle (through all the nodes) which ensures that the graph is connected. It is easy to see that the existence of such a cycle ensures that there are no dead ends in the graph. There may be multiple directed edges between a pair of nodes, and your solution should treat them as the same edge. The first column in graph-full.txt refers to the source node, and the second column refers to the destination node.

Implementation

Assume the directed graph $G = (V,E)$ has n nodes (numbered 1,2,…,n) and m edges, all nodes have positive out-degree, and $M = [M_{ji}]_{n\times n}$ is a $n \times n$ matrix as defined in class such that for any $i, j \in [\![1, n]\!]$:

$$M_{ji} = \begin{cases} \frac{1}{\deg(i)} & \text{if } (i \to j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Here, deg(i) is the number of outgoing edges of node i in G. If there are multiple edges in the same direction between two nodes, treat them as a single edge. By the definition of PageRank, assuming $1-\beta$ to be the teleport probability, and denoting the PageRank vector by the column vector r, we have the following equation:

$$r = \frac{1-\beta}{n}\mathbf{1} + \beta M r, \tag{1}$$

where $\mathbf{1}$ is the $n \times 1$ vector with all entries equal to 1.

Based on this equation, the iterative procedure to compute PageRank works as follows:

1. Initialize: $r^{(0)} = \frac{1}{n}\mathbf{1}$
2. For i from 1 to k, iterate: $r^{(i)} = \frac{1-\beta}{n}\mathbf{1} + \beta M r^{(i-1)}$

Run the aforementioned iterative process in Spark for 40 iterations (assuming β = 0.8) and obtain the PageRank vector r. You do not have to implement the *block-based algorithms* mentioned in the lectures to develop the PageRank algorithm; the vanilla implementation would work. Matrix M can be large and should be processed as an RDD in your solution.

Compute the followings for the larger dataset:

- List the top 5 node ids with the highest PageRank scores.
- List the bottom 5 node ids with the lowest PageRank scores.

***Implementation hints**:*

- *You may choose to store the PageRank vector r either in memory or as an RDD. However, matrix of links and M must not be stored in memory.*
- *The page ranks should sum up to ~1 at each iteration.*
- *For a sanity check, we have provided a smaller dataset (graph-small.txt). In that dataset, the top node has id 53 with value ~0.036 (after 40 iterations).*

Your need to declare the following variables on top of your code:

```
dataset_path= '' # path to the dataset
iterations = 40 # number of iterations
beta = 0.8
```

Not that you must not hardcode the number of nodes in your program, as it may be tested with a different dataset.

The driver function (which is the only function directly called for grading) should have the following signature:

```
page_rank(dataset_path, iterations, beta)
```

At the end of each iteration, your code must generate the id and score for each of the top 5 nodes in the following format (below values are synthetic):

Iteration 1:

43: 0.04234
12: 0.04012
10: 0.03900
90: 0.03802
27: 0.01234

Submission

You need to submit a .zip file on Canvas, named as your-lastname_your-first-name.zip containing the following items:

A. The code as a Jupyter Notebook (.ipynb) or a .txt file (**70 points**). If your code is not in Spark or you collect the dataset or build M in memory, you will get 0 for this part. You can use PySpark, Scala-Spark, or Java-Spark, but PySpark is strongly recommended.
B. A pdf file containing (**30 points**):
    - List of the top 5 node ids with the highest PageRank scores. Include the scores alongside the ids (10 points).
    - List of the bottom 5 node ids with the lowest PageRank scores. Include the scores alongside the ids (10 points).
    - Output of your program for all 40 iterations (10 points)

DO NOT INCLUDE EXTRA FILES, SUCH AS THE DATASETS, in your submission.

Please download your assignment after submission and make sure it is not corrupted. We will not be responsible for corrupted submissions and will not be able to take a resubmission after the deadline.

Need Help?

In case you need help with the details of this assignment, please email Shahab at shahab.helmi@ucdenver.edu or go to his office hours (Tuesday/Thursday 9-10 am).

Evan will be able to help only with the general Spark/Python questions for this assignment.

You are highly encouraged to ask your question on the designated channel for Assignment 3 on Microsoft Teams (not monitored by the TA's). Feel free to help other students with general questions. DO NOT share your solution.