DS702 Assignment 4

Release Date: 20 March 2022
Due Date: 10 April 2022

- Submit your answers as an electronic copy on Moodle (pdf, jupyter notebook).
- No unapproved extension of deadline is allowed. For emergencies and sickness, extensions must be requested as soon as possible.
- Cite your sources if you are taking help (papers, websites, students etc.).
- Plagiarism is strictly prohibited. Negative mark will be assigned for plagiarism.
- Remember to comment your code. And your answers should be detailed.

1 PageRank

Exercise 1.1: Compute the PageRank of each page in Figure 1, assuming no taxation.

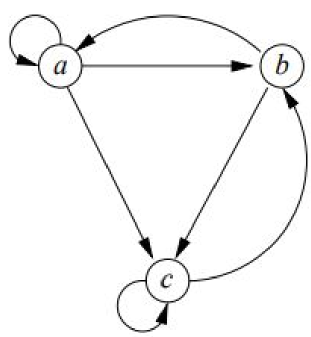

Figure 1: An example graph for exercises

Exercise 1.2: Compute the PageRank of each page in Figure 1, assuming $\beta = 0.8$.

Exercise 1.3: Suppose the Web consists of a clique (set of nodes with all possible arcs from one to another) of n nodes and a single additional node that is the successor of each of the n nodes in the clique. Figure 2 shows this graph for the case n = 4. Determine the PageRank of each page, as a function of n and β.

Exercise 1.4: Suppose we recursively eliminate dead ends from the graph, solve the remaining graph, and estimate the PageRank for the dead-end pages as described in Section 5.1.4 [1]. Suppose the graph is a chain of dead ends, headed by a node with a self-loop, as suggested in Figure 3. What would be the PageRank assigned to each of the nodes?

Exercise 1.5: Repeat Exercise 1.4 for the tree of dead ends suggested by Fig. 5.10. That is, there is a single node with a self-loop, which is also the root of a complete binary tree of n levels.

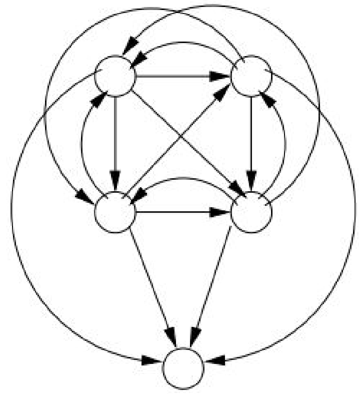

Figure 2: An example graph for exercise 1.3.

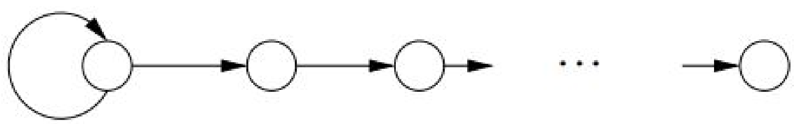

Figure 3: A chain of dead ends

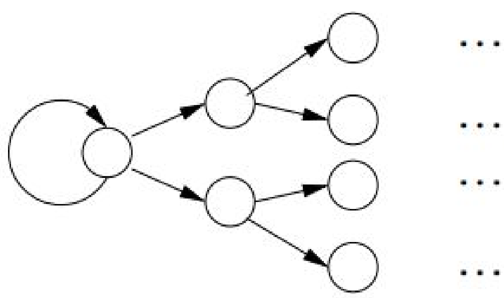

Figure 4: A tree of dead ends

2 Topic-Sensitive PageRank

Compute the topic-sensitive PageRank for the graph of Figure 5, assuming the teleport set is:

(a) A only.

(b) A and C.

3 Link Spam

In Section 5.4.2 [1] we analyzed the spam farm of Figure 6, where every supporting page links back to the target page. Repeat the analysis for a spam farm in which:

(a) Each supporting page links to itself instead of to the target page.

(b) Each supporting page links nowhere.

(c) Each supporting page links both to itself and to the target page.

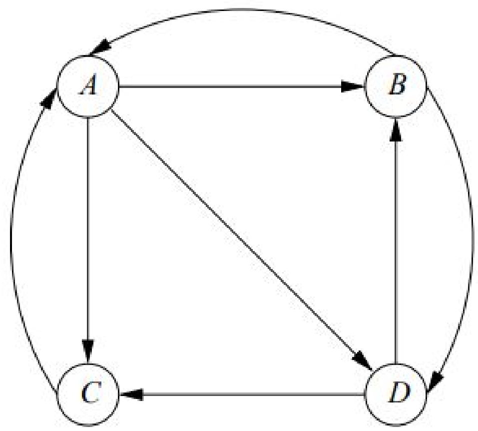

Figure 5: Web graph

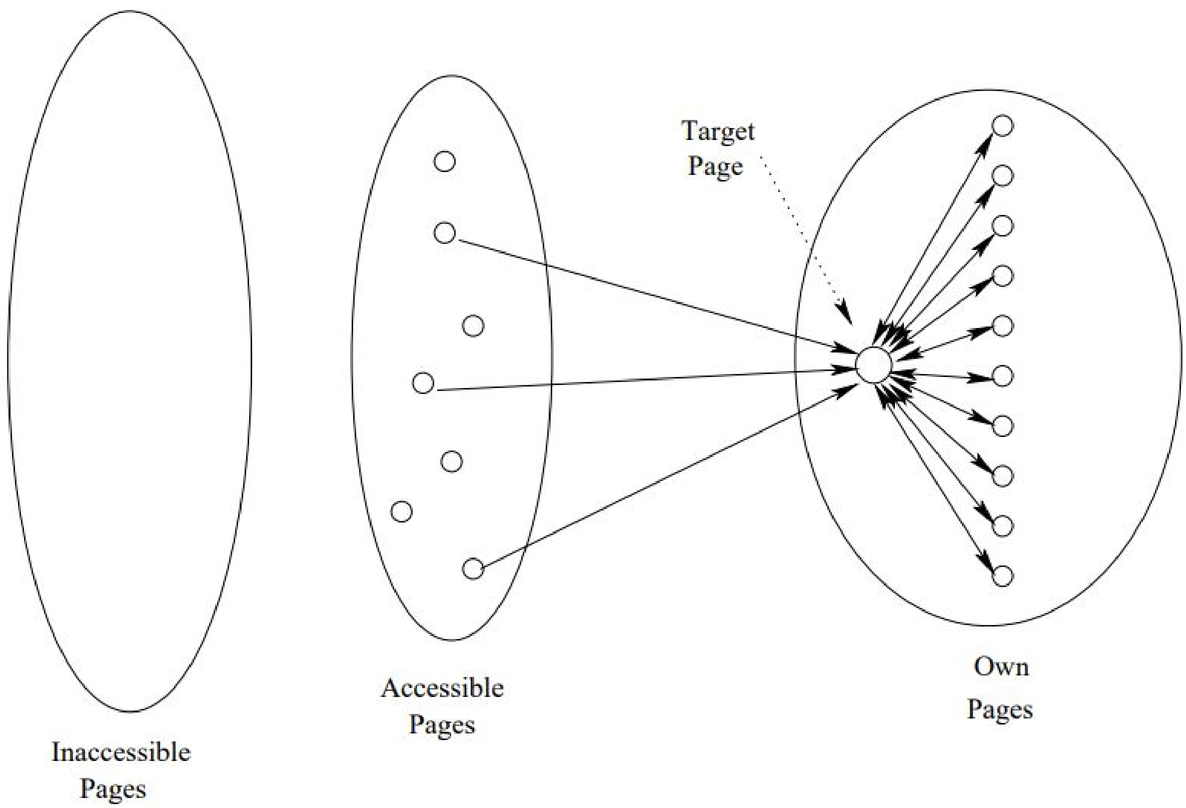

Figure 6: The Web from the point of view of the link spammer

4 Dead ends in PageRank computations

Let the *matrix of the Web* M be an n-by-n matrix, where n is the number of Web pages. The entry m_{ij} in row i and column j is 0, unless there is an arc from node (page) j to node i. In that case, the value of m_{ij} is $1/k$, where k is the number of arcs (links) out of node j. Notice that if node j has k ¿ 0 arcs out, then column j has k values of $1/k$ and the rest 0's. If node j is a *dead end* (i.e., it has zero arcs out), then column j is all 0's.

Let $\mathbf{r} = [r_1, r_2, ..., r_n]^T$ be (an estimate of) the PageRank vector; that is, r_i is the estimate of the

PageRank of node i. Define $w(\mathbf{r})$ to be the sum of the components of $\mathbf{r}$; that is $w(\mathbf{r}) = \sum_{i=1}^{n} r_i$.

In one iteration of the PageRank algorithm, we compute the next estimate **r'** of the PageRank as: $\mathbf{r'} = M\mathbf{r}$. Specifically, for each i we compute $w(\mathbf{r'}) = \sum_{i=1}^{n} M_{ij} r_j$. Define $w(\mathbf{r'})$ to be the sum of components of **r'**; that is $w(\mathbf{r'}) = \sum_{i=1}^{n} r'_i$.

You may use D (the set of dead nodes) in your equation.

(a) Suppose the Web has no dead ends. Prove that $w(r') = w(r)$.

(b) Suppose there are still no dead ends, but we use a teleportation probability of $1 - \beta$, where $0 < \beta < 1$. The expression for the next estimate of r_i becomes $r'_i = \beta \sum_{j=1}^{n} M_{ij} r_j + (1 - \beta)/n$. Under what circumstances will $w(r') = w(r)$? Prove your conclusion.

(c) Now, let us assume a teleportation probability of 1β in addition to the fact that there are one or more dead ends. Call a node `"dead"` if it is a dead end and `"live"` if not. Assume $w(\mathbf{r}) = 1$. At each iteration, each live node j distributes $(1\beta) r_j / n$ PageRank to each of the other nodes, and each dead node j distributes r_j / n PageRank to each of the other nodes. Write the equation for r'_i in terms of β, M, $\mathbf{r}$, n, and D (where D is the set of dead nodes). Then, prove that $w(\mathbf{r'})$ is also 1.

5 Implementing PageRank

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 at `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 hint: You may choose to store the PageRank vector r either in memory or as an RDD. Only the matrix of links is too large to store in memory.

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

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

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:

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

where $\mathbf{1}$ is the n × 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 $\beta = 0.8$) and obtain the PageRank vector r. In particular, you don't have to implement the blocking algorithm from lecture. The matrix M can be large and should be processed as an RDD in your solution. Compute the following:

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

References

[1] Jure Leskovec et al. Mining of Massive Datasets. 2019