College of Computing and Information Science — CS6200
Northeastern University — Information Retrieval
Fall 2018 — October 6th, 2018

Homework Assignment #2
Due Date: October 15th, 2018 @ 11:59pm

Instructions:

- The assignment is due on the time and date specified.
- This is an individual assignment. You may discuss the problems with your friends, but the code, analysis, interpretation and write-up that you submit for evaluation should be entirely your own.
- You are encouraged to use the Piazza discussion board, and seek help from TAs and instructors to get clarifications on the problems posed.
- If you receive help from others you must write their names down on your submission and explain how they helped you.
- If you use external resources you must mention them explicitly. You may use third party libraries but you need to cite them, too.

Goal: Link Analysis and PageRank Implementation

Task 1: Constructing directed web graphs (20 points)

A. Build a graph over the set of the 1000 URLs you collected in HW1-Task1 (unfocused Breadth-First crawling starting with the seed: https://en.wikipedia.org/wiki/Carbon_footprint)

Your graph should have a structure as shown below:

```
D1 D2 D3 D4   // Node D1 has incoming links from nodes D2, D3 and D4.
D2 D5 D6      // Node D2 has incoming links from nodes D5, and D6.
D3 D7 D8      // Node D3 has incoming links from nodes D7, and D8.
….
```

In your web graph, one of the nodes will correspond to the webpage *docID* which is the article title directly extracted from the URL (e.g., Carbon_footprint is the docID for https://en.wikipedia.org/wiki/Carbon_footprint).

Each line indicates the **in-link relationship**, which means that D1 will have in-coming links from all the URLs that link to D1. You only need to build the graph for the 1000 web pages you have crawled, and do not need to consider any other web pages.

We will name this graph **G1**.

B. Build a graph for the 1000 URLs that you obtained from running the focused crawler for HW1-Task3. We will refer to this graph as **G2**.

Task 2: Link analysis: PageRank Implementation (40 points)

A. Implement the PageRank algorithm. PageRank can be computed iteratively using the pseudo-code given below.

```
// P is the set of all pages; |P| = N
// S is the set of sink nodes, i.e., pages that have no out links
// M(p) is the set (without duplicates) of pages that link to page p
// L(q) is the number of out-links (without duplicates) from page q
// d is the PageRank damping/teleportation factor; use d = 0.85 as a fairly
typical value

foreach page p in P
    PR(p) = 1/N                    /* initial value */
while PageRank has not converged do
    sinkPR = 0
    foreach page p in S            /* calculate total sink PR */
        sinkPR += PR(p)
    foreach page p in P
        newPR(p) = (1-d)/N         /* teleportation */
        newPR(p) += d*sinkPR/N     /* spread remaining sink PR evenly */
        foreach page q in M(p)     /* pages pointing to p */
            newPR(p) += d*PR(q)/L(q)        /* add share of PageRank from in-
links */
    foreach page p
        PR(p) = newPR(p)
Return and output final PR score.
```

B. To test for convergence, calculate the L1-norm for the difference in PageRank values from successive iterations.

$$L_1(newPR, PR) = \Sigma_p\,|newPR(p) - PR(p)|,$$

where $|x|$ denotes the absolute value of x, and the summation is over all of the web pages p in our graph.

PageRank can be considered to have converged if $L_1(newPR, PR)$ is less than 0.001 for at least four consecutive iterations.

C. (Ungraded task) You can first test your PageRank algorithm on the following small graph:

```
A D E F
B A F
C A B D
D B C
E B C D F
F A B D
```

The final ranking should be: A>E>(F,C)>B>D (F and C have the same PageRank value)

D. Finally, run your iterative version of PageRank algorithm on G1 and G2 respectively until their PageRank values "converge". Your results should be a list of the URLs sorted by PageRank value in descending order.

Task 3: Experiments with PageRank (40 points)

A. Perform the following runs for **both** graphs G1 and G2.
As a **baseline** for comparison, use the resulting PageRank for G1 (BFS) from Task2-D.

1) Re-run the PageRank algorithm using damping factors of d = 0.5 and d = 0.65. What do you observe in the resulting PageRank values relative to the baseline? Discuss the results.
2) Re-run the PageRank algorithm in Task2-D for exactly 4 iterations. Discuss the results obtained with respect to the baseline.
3) Sort the documents based on their raw in-link count. Compare the top 20 documents in this sorted list to those obtained in Task2-D sorted by PageRank. Discuss the pros and cons of using the in-link count as an alternative to PageRank (address at least 2 pros and 2 cons).

What to hand in:

1) The source code of your PageRank algorithm implementation. This should include code for Tasks 1, 2 and 3.
2) A README.txt file for instructions on how to compile and run your code.

For Task 1:

3) The graph files you generated for G1 and G2 (as text files).
4) A brief report on simple statistics for G1 and G2 including:
    a. The number of pages with no in-links (sources)
    b. The number of pages with no out-links (sinks)
    c. Maximum in-degree
    d. Maximum Out-degree

For Task 2:

5) A file listing the L1-norm values until convergence for G1 and G2, as well as the sum of the computed PageRank values (summation over newPR).
6) A sorted list of the pages in G1 and G2 by the steady-state values of PageRank. Report the Top 50 pages by their docID and score.

For Task 3:

7) A short report on your findings from the suggested experiments in 3.1, 3.2 and 3.1 (one paragraph each).

Compress your all files into one folder and name your folder using the following format: FName_LName_HW2