Northeastern University
CS6200 - Fall 2016

# Assignment 2:

This is an individual assignment. If you get 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.

Date posted: October 10th, 2016

Date Due: October 19th, 2016

## Goal: Link Analysis and PageRank Implementation

## Task1: Obtaining directed web graphs (10 pts)

### *A. Build your own*

Build a graph for the 1000 URLs you crawled in HW1-Task1 by following their links. You may modify and re-run your crawler OR extract the links from the articles you downloaded. Your graph should look like follows.

D1 D2 D3 D4

D2 D5 D6

D3 D7 D8

….

Where, D1 is the webpage ***docID*** which is the article title directly extracted from the URL (e.g., Renewable_energy is the docID for https://en.wikipedia.org/wiki/Renewable_energy). Each line indicates the **in-link relationship**, which means that D1 has three in-coming links from D2, D3, and D4 respectively. 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 you might come across in the re-crawling process. Suppose we name this graph as G1.

### ***B.*** **Use existing ones**

Download the *in-links file* for the *WT2g* collection. *WT2g* is a 2GB crawl of a subset of the web, containing **183,811** web documents. This in-links file is in the format described above, with the destination followed by a list of source documents. This graph is already built and cleaned. We will refer to this graph as G2.

## Task 2: Implementing and running PageRank (65 pts)

A. Implement the PageRank algorithm. PageRank can be computed iteratively by following the pseudocode as 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. Run your iterative version of PageRank algorithm on G1 and G2 respectively until their PageRank values "converge". To test for convergence, calculate the *perplexity* of the PageRank distribution, where perplexity is simply 2 raised to the (Shannon) *entropy* of the PageRank distribution, i.e., 2^H (PR).

$$Perplexity = 2^{H(PR)}$$

$$H(PR) = -\sum_{i=1}^{N} P(x_i)\log_2 P(x_i)$$

Where, $x_i$ denotes one web page, $N$ is the total number of pages in the corpus, and $P(x_i)$ is the PageRank score for page $x_i$.

Perplexity is a measure of how "skewed" a distribution is: the more "skewed" (i.e., less uniform) a distribution is, the lower its perplexity. Informally, you can think of perplexity as measuring the number of elements that have a "reasonably large" probability weight; technically, the perplexity of a distribution with entropy h is the number of elements n such that a uniform distribution over n elements would also have entropy h. (Hence, both distributions would be equally "unpredictable").

PageRank can be considered as converged if the change in perplexity is less than 1 for at least four consecutive iterations.

C*: You can firstly 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
```

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

## Task 3: Qualitative Analysis (25 pts)

Examine the *Top 5* pages by PageRank and the *Top 5* by in-link counts for G1 and G2. For G1, you can check the original web page using its URL, for G2, in order to check the original web page, you can get access via the Lemur web interface to the collection by using the "e=docID" option with database "d=0", which is the index of the WT2g collection. For example, the link

http://fiji4.ccs.neu.edu/~zerg/lemurcgi_IRclass/lemur.cgi?d=0&e=WT04-B22-268

or

http://karachi.ccs.neu.edu/~zerg/lemurcgi_IRclass/lemur.cgi?d=0&e=WT04-B22-268

will bring up document WT04-B22-268, which is an article on the Comprehensive Test Ban Treaty.

Why do you think these web pages have high PageRank values? (Provide your own speculation).

## ***What to hand in:***

1) The source code of your PageRank algorithm implementation.
2) A README.txt file for instructions on how to compile and run your code.
3) In Task 1, the graph file you generated for G1 (in text file)
4) In Task 1, a brief report on simple statistics over G1 and G2 including for both graphs: (a) the proportion of: (a) pages with no in-links (sources), (b) pages with no out-links (sinks)
5) In Task 2, a file listing perplexity values you obtain in each round until convergence for G1
6) In Task 2, a file listing perplexity values you obtain in each round until convergence for G2
7) In Task 2, sort the pages in G1 in terms of the final PageRank score. Report the Top 50 pages by their docID and PageRank score
8) In Task 2, sort the pages in G2 in terms of the final PageRank score. Report the Top 50 pages by their docID and PageRank score
9) Task 3 speculation