Web Retrieval and Mining Assignment6

資工三 江東峻B01902032

---

1. How to implement the algorithm on sparse graph

    1. Sparse Graph

        使用 Compressed Column Storage 方法。因為輸入的時候是逐行輸入。而最後要使用的是adjacency matrix的transpose，所以輸入的時候要逐欄儲存，剛好就是Compressed Column Storage。（存成長度為link個數的array)

        若是zero out degree的node，用一個null_col的陣列存起來：若第i欄是zero out degree，則null_col[i] = 1。

        reference: http://www.cs.colostate.edu/~mroberts/toolbox/c++/sparseMatrix/sparse_matrix_compression.html

    2. Efficiency improving

        因為處理null column時，必須視為連到所有node，若是正常的算法會是O(node^2)，跑一次就得跑很久。而不是null column的部分，因為使用Compressed Column Storage ，只需要跑有link的部分，所以最多是O(link數)。

        為了減少null column的計算時間，把PageRank的公式拆成三項：

        P' = (1-d) + d(sum of not-null columns)+ d(sum of null columns)

        其中第一項在初始化的時候就設定好，第二項是使用Compressed Column Storage，最多是O(link)，第三項因為每個null column加在P'的值都一樣，所以可以事先算好，最後每一項P'要加上：

        d　　DAMPING_FACTOR

        * (1/(node_num-1))　　視為連到除了該column以外所有人

        *{(nP_sum-P[i]) if i is no-out-link node, nP_sum otherwise}

        nP_sum為所有null column的P值加總

        nP_sum的計算量最多為O(node)，P'更新也為O(node)

        所以原本算null column的O(node^2)可加速為O(node)。

2. What I find in this task

1. 不能直接把adjacency matrix存起來，必須使用一些壓縮的方法。例如: Compressed Column Storage。可以同時減少計算時的time、space complexity。
2. 計算時，有些重複計算的部分可以合併計算，以減少計算時間。
3. 對stanford-08-03改epsilon，epsilon越小，收斂的時間越長，但是越精確。（對助教給的答案測）
4. 對stanford-08-03改damping值，似乎助教給的答案是d=0.85左右的版本(L1-norm: 0.000153, Spearman's rho: 1)，因為不管怎麼改L1-norm都會變高，而d越高，收斂的時間越長
5. Experiment results:

fixed epsilon = 10^(-6):

Damping factor	L1-norm	Spearmen’s rho
0.2	235440	0.986859
0.3	216057	0.990969
0.4	193680	0.993947
0.5	167268	0.996231
0.6	135257	0.997905
0.7	94573.1	0.999156
0.85	0.000153	1
0.9	51813.4	0.999842
0.95	134796	0.999191

fixed damping factor=0.85:

Epsilon: 10^(-n)	L1-norm	Spearmen’s rho
n=1	22.7414	1
n=2	2.27754	1
n=3	0.223418	1
n=4	0.020597	1
n=5	0.001725	1
n=6	0.000153	1
n=7	1.9E-05	1
n=8	1E-06	1
n=9	0	1