DataMining Report3

Dataset

從graph1~6讀txt檔，轉換成dictionary

例如 1,2 1,3 整理成dict{1:{2,3}}

Graph 7則是從project1抓出的data，並轉成相同格式

Graph 8 從bi-directed改成directed

Implementation detail

HITS算法

1.對data進行matrix的建立

2.計算各節點的hub與authority

3.賦予每個節點的hub值與authority值為1

4.接著運行authority的更新規則，然後是hub的更新規則，當authority值>0.1則持續更新

5.最後normalize數值，每個節點的hub及authority分別除以hub之和及authority之和，這樣做是確保最大值為1

def normalize(vector):

max = 0

for component in vector:

if vector[component] > max:

max = vector[component]

if max == 0:

return vector

for component in vector:

vector[component] = float(vector[component]) / max

return vector

Result相關討論

在Graph1中，node1的authority為0，因為沒被其他值指到，最後一個node沒有指向任何node，故hub值為0

Authority:{1: 1.0, 2: 1.0, 3: 1.0, 4: 1.0, 5: 1.0}

Hubbiness: {1: 1.0, 2: 1.0, 3: 1.0, 4: 1.0, 5: 1.0}

在graph2中，因為為循環，所以node和authority值皆同

Authority:{1: 1.0, 2: 1.0, 3: 1.0, 4: 1.0, 5: 1.0}

Hubbiness: {1: 1.0, 2: 1.0, 3: 1.0, 4: 1.0, 5: 1.0}

Graph3中，邊緣兩值的hub和authority較低，因連接的點較少

Authority:{1: 0.625, 2: 1.0, 3: 1.0, 4: 0.625}

Hubbiness: {1: 0.6000000000000001, 2: 1.0, 3: 1.0, 4: 0.6000000000000001}

PageRank算法

EPS=1e-6

Rate=0.15

1.把data存入matrix

2.R:pagerank，初始為1/node數

3.進行公式運算，再normalize，直到兩次的rank差距小於eps即終止

最後算出PageRank，迭代次數

SimRank算法

C=0.8 最大迭代次數=1

1.同樣將data轉成matrix

2.建立各節點的adjacency matrix，初始化matrix為單位矩陣

3.選定兩個node，相同則跳過

4.尋找兩個node的prefix和postfix，相乘回傳

5.重複數次結束，算出simrank 的matrix

Result 相關討論

Graph1、2中，node之間沒有交集，因此單位矩陣=SimRank矩陣

[[1. 0. 0. 0. 0.]

[0. 1. 0. 0. 0.]

[0. 0. 1. 0. 0.]

[0. 0. 0. 1. 0.]

[0. 0. 0. 0. 1.]]

Graph3則會是一個對稱矩陣

[[1. 0.4 0. 0. ]

[0.4 1. 0. 0. ]

[0. 0. 1. 0.4]

[0. 0. 0.4 1. ]]

Find a way (e.g., add/delete some links) to increase hub, authority, and PageRank of Node 1 in first 3 graphs respectively.

要增加hub和authority的一個簡單方法就是增加節點，從graph1下手

因為node1沒有任何節點指向他，所以新增一個node 4,1來增加他的authority值

這是執行後的結果

Authority:{1: 1.0, 2: 0.03125, 3: 0.03125, 4: 0.03125, 5: 1.0}

Hubbiness: {1: 0.03125, 2: 0.03125, 3: 0.03125, 4: 1.0, 5: 0.0}

很明顯的node1的authority值增加了，再來處理增加hub值的方法

同樣增加節點，加入一個6,2的節點讓6可以指向其他節點

結果如下

Authority:{1: 1.0, 2: 1.0, 3: 0.03125, 4: 0.03125, 5: 1.0, 6: 0.03125}

Hubbiness: {1: 0.5, 2: 0.03125, 3: 0.03125, 4: 1.0, 5: 0.03125, 6: 0.5}

6的hub值增加了

剩下的graph同樣也是可以用增加節點的方式讓hub值&authority值增加

Computation performance analysis

很明顯的當節點越多，程式執行的會越久，這是演算法導致的結果

舉PageRank為例

這是graph1的執行結果

0.000499831367626479

total iteration is 4

total time is 0.004912

再來是graph6的結果

0.0005165394904965082

total iteration is 3

total time is 0.006842

很明顯的雖然迭代次數減少了，但跑的時間明顯變長，由於這只是千筆內的小資料差距沒那麼明顯，但資料量越大時間差就越大

Discussion

從這次作業有稍微了解到HITS、PageRank、SimRank的部分實作內容，因為只有implement出部分核心的部分，所以可能沒辦法了解到非常透徹。

另外也了解的這些演算法的問題，據資料顯示GOOGLE已在好幾年前就捨棄了這幾個演算法去做排名，HITS演算法有著作弊的問題，無法真實的顯示網站的質量。

但如果要加入質量去做排名，那該如何定義質量好壞，又是一個很大的問題。