HW3 report F44054045 賴廷瑋

Implementation detail

(i) PageRank

為了實踐 PageRank，定義一個轉移矩陣 $\mathrm{M}=\left[m_{ij}\right]_{n\,x\,n}$，其中 n 為總結點數，i 表示第 i 個被指向的節點，j 為第 j 個指向其他節點的節點。在此舉陣中，每個 column 的總和為 1。

定義 V 為所有節點的集合(set)，R 為全部節點各自的 PageRank，e 為 [1/n, 1/n, ⋯ , 1/n]。為解決 Spider Trap 的問題，定義 d 為阻尼係數，表示每節點都有(1-d)的機率會隨機跳到其他節點(damping factor = 1-d)。

實踐過程中首先將檔案讀取資料後生成轉移矩陣(M)及總節點數量(n)，再以一下過程迭代得到節點的 PageRank：

(1) 初始令 R0 = (1-d) e

(2) 計算 $R = dMR + \frac{1-d}{n}1$

(3) 若對於每個節點，新的 R 和舊的 R 相差皆小於 ε = 1e-4，則停止迭代。

(4) 否則繼續執行 (2)

(ii) HITS

因為 HITS 演算法涵蓋計算 Hubs & Authority 的概念，故定義 L 為連結矩陣(Link Matrix)，即 $\mathrm{L}=\left[l_{ij}\right]_{nxn}$，若$l_{ij}=1$，則表示節點 i 連結到 j 節點，若無則 0，其中 n 為總共的節點數。另外設 a 為各節點的 authority，h 為各節點的 hub (皆為一維向量)，λ 及 μ 各為 a 與 h 的 scale 係數，使得 a 與 h 內最大值為 1

迭代過程

(1) 初始設定 h0 = [1, 1, … , 1]

(2) 計算 $a = L^T h\lambda$

(3) 計算 $h = La\,\mu$

(4) 對於每個節點，若新的 a, h 皆與舊的 a, h 相差小於 ε = 1e-4，則停止迭代

(5) 否則繼續 (2), (3)

(iii) SimRank

定義：

1. 若節點 i 指向節點 j，則稱 i 節點爲 j 節點的 in neighbor，j 節點爲 i 節點的 out neighbor，並定義二部圖爲 G(V,E)，其中 V 爲所有 in neighbors 的集合，E 爲所有 out neighbors 的集合，若 G[v, e] = 1 表示節點 v 指向節點 e，否則爲 0。

2. 因 in neighbors 之間的 SimRank 取決於 out neighbors 的 SimRank，反之亦然，所以定義 S_v 爲 in neighbors 之間的 SimRank；S_e 爲 out_neighbors 之間的 SimRank，兩者互相迭代收斂於定值。

迭代過程

設總共迭代次數爲 n_epochs, 阻尼係數爲 C

(1) 將 S_v, S_e 初始皆設爲 Identity Matrix, t= 1

(2) 計算

$$s(A,B) = \frac{C_1}{|O(A)||O(B)|} \sum_{i-1}^{|O(A)||O(B)|} \sum_{j-1} s(O_i(A), O_j(B)) \quad for\ A \neq B$$

其中 s(A, B)即爲 S_v，O(A)表 A 節點的 out neighbor，C1 = C = 阻尼係數。

(3) 計算

$$s(a,b) = \frac{C_2}{|I(a)||I(b)|} \sum_{i-1}^{|I(a)||I(b)|} \sum_{j-1} s(I_i(a), I_j(b)) \quad for\ a \neq b$$

其中 s(a, b)即爲 S_e，I(a)表 a 節點的 in neighbor，C2 = C = 阻尼係數。

(4) t = t+1

(5) 若 t > n_epochs, 停止迭代。

(6) 否則繼續 (2) & (3) & (4)

Result analysis and discussion

所有矩陣 M_{ij} 結果中皆表示節點 i 與 節點 j 的關係，其中節點 I,j 即 data 中的 node i, j

(i) Graph 1

1 → 2 → 3 → 4 → 5 → 6

- PageRank

  觀察演算法結果，PageRank 最大爲節點 6，依序降冪到節點 1。其實我個人覺得 PageRank 對 graph 1 的重要性的解讀要看應用的場景， 因爲一個節點的 PageRank 會取決於指向它的節點的 PageRank，但是因爲 graph 1 是延續連接的 edge，PageRank 會持續增加，這時如果是部落格之間的連結，我們不能說部落格 6 就是比較重要，它可能只是剛好沒有指向任何其他部落格。

- Authorities

  節點二到六的 authority 相同，節點一爲 0，可以很清楚看到 authority 計算的特性，因爲除了節點 1，每個節點都被一個節點指向，所以有相同的值，相對直覺

- Hubs

  節點 6 的 Hub 爲 0，其他節點都有相同的 hub，也很容易解釋，每個節點都指向一個節點，以部落格的例子來想，的確 share 相同的 hub，不會因爲連續的指向關係而有不同的 hub 值

- SimRank

  從結果觀察到相異任兩點之間的 simRank 皆爲 0，以 SimRank 演算法的角度計算的確會得出這樣的結果，但以 SimRank 的算法有一個缺點，就是沒辦法考慮 in neighbor 和 out neighbor 之間的關係，因爲我的直覺是，如果 node i 指向 node j，那這兩個節點之間應該具有一定的相關性。

(ii) Graph 2

- PageRank：針對圓形的指向圖，每個節點的 PageRank 皆相同，這時候就跟直覺一樣，以部落格爲例，每篇都指向不同的另一篇，且被指向的部落格都不同，則無法辨識誰比較重要，可以從 PageRank 看出。
- Hit & authority：

  每個點對於 hits 和 authority 皆相同，也符合直覺。

- SimRank：

  每個節點和其他節點 SimRank 皆爲 0，針對圓形的指向圖，我覺得節點間應該要具有相關性，但 SimRank 把指出和指向的節點當作兩個不同的群，所以之間沒有關係可以計算，這點可能可以加以改

進。

(iii) Graph 3

- PageRank：
  以圖來看節點二三都分別有和兩個節點連接(in & out)，而節點 1,4 都只跟一個節點連接，以部落客來講的確會把節點 2,3 視爲較重要的 page，此時 PageRank 算法可以反應這個重要性。
- Hubs and authorities：
  節點 2, 3 的 hub 及 authority 都相同且都比 1,4 高，這部分也很合理，因爲 2, 3 分別都指向和被指向於兩個節點，相對來講更爲重要(對 hub & authority)
- SimRank：
  SimRank 結果顯示 1, 3 之間有關係，以結論來說可以知道原因是因爲 1, 3 相同都指向 2 也被 2 指向，所以應該有相似性，這部分我覺得以 SimRank 來算很合理

Graph4, 5, 6 因爲圖形複雜度較高，所以重要概念皆在 1,2,3 圖討論了。

Computation performance analysis

一開始使用 SimRank 時，在 graph1, 2, 3, 4 計算上都不會花很多時間，但到了 graph 5，時間就會花非常久，之後發現的原因是在 SimRank 計算時一次僅考慮兩節點的 SimRank，而之後利用了矩陣運算的方式就可以在一秒內將 graph5 的 SimRank 計算出來，相同的，在 PageRank 我利用矩陣運算的方式也讓計算時間有一定的縮短。因爲目前資料量其實算少，眞實世界的資料非常龐大，利用矩陣運算會是在 Link Analysis 中非常重要的技術。

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

- Graph 1
  1. 如果越多重要的節點指向節點 a，則節點 a 的 pagerank 會增加，所以將節點 2 指向節點 1 的話，節點 1 的 pageRank 及 authority 將會增加。
  2. 將節點 1 額外指向節點 6，則節點 1 hub 會增加。
- Graph 2

  同樣的，若將節點 2 指向節點 1，則節點 1 的 PageRank 及 authority 會增加；若將節點一額外指向節點 3，則節點 1 的 hub 也會增加。
- Graph 3

  若將節點 4 指向節點 1，則節點 1 的 PageRank 及 authority 會增加；若將節點一額外指向節點 3，則節點 1 的 hub 也會增加。