Link Analysis Report – HITS, PageRank, SimRank

I. Environment

OS: Ubuntu 16.04 KVM
Python version: 2.7.12

II. Results of Datasets

針對 hw3dataset 中的資料執行 HITS, PageRank & SimRank 演算法結果儲存於 results.txt 檔案中，Experiment2 為 HITS 與 PageRank 部分，包含助教提供的 Graph1~6.txt 與 project1 所使用的交易記錄 test_10000.data，Experiments 為 SimRank 部分。

III. Increase hub, authority & PageRank

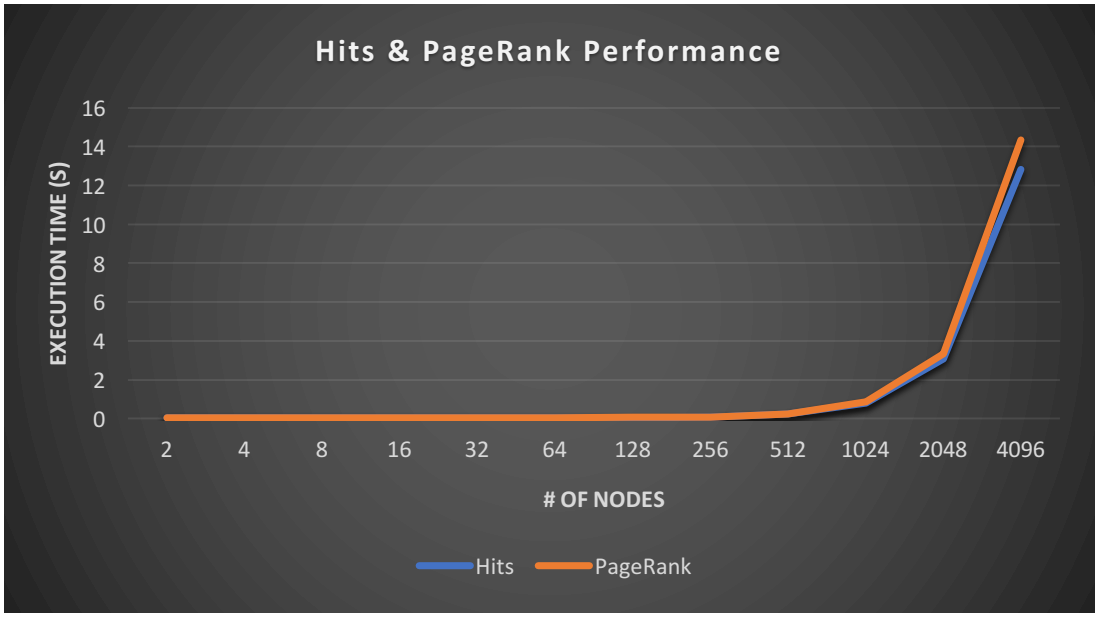
- 提升 Hub

Hub 值高代表可以指向許多高 Authority 的節點，假設我們可以控制整張圖的 Links，我們可以將欲提升節點指向圖中除自身之外擁有最高 Authority 的節點，使欲提升節點的 Hub 值增加。以 hw3dataset/graph_3.txt 為例，原始 Authority & Hub 值皆為 [0.19090909 0.30909091 0.30909091 0.19090909]，為了提升節點 1 的 Hub 值，添加 1->3 的 Links 後 Hub 值變成 [0.33831221 0.27948842 0.17330394 0.20889543]，可見此方法能有效提升某節點的 Hub 值。

- 提升 Authority

Authority 值高代表被許多高 Hub 值的節點所指向，我們可以將圖中除欲提升節點外擁有最高 Hub 值的節點指向欲提升節點，使欲提升節點的 Authority 增加。以 hw3dataset/graph_3.txt 為例，為了提升節點 1 的 Authority 值，添加 3->1 的 Links 後 Authority 值變成 [0.33831221 0.27948842 0.17330394 0.20889543]，節點 1 的 Authority 值由 0.19 變成 0.33 ，可見此方法能使節點的 Authority 得到顯著地增加。

IV. Performance: HITS vs. PageRank



圖一、Hits 與 PageRank 效能分析圖

為比較 Hits 與 PageRank 的效能，分別生成 2, 4, 8…, 4096 個節點的 full mesh 圖形，如圖一所示，可見 Hits 隨著節點數目的成長有相對好的效能。此外，本次作業中的 simrank.py 使用迭代式的方法，最初從 {S(x,x) | x 為任意節點} 開始傳遞相似度，相較於 simrank_recursive.py 的遞迴方法快上許多，雖然 simrank_recursive.py 使用 Dynamic Programming 方法記錄曾經算過的相似度，但在查表跟 Function call 仍然消耗大量的時間，且 simrank_recusive.py 為了避免無窮迴圈，遞迴深度必須被限制，然而 simrank_recursive.py 在查詢少數兩兩配對時效能比迭代版本好，因為迭代版本必須跑過整張圖才能得到結果。

V. Implementation detail

- **HITS**

  1.將資料讀入以 csr_matrix 形式保存

  2.csr_matrix 與 auth 向量相乘後可得新的 hub

  3.csr_matrix 轉置後與 hub 向量相乘可得新的 auth

  4.檢查新舊 hub 與 auth 的差異是否小於 EPSILON，若小於則終止

- **PageRank**

  1.將資料讀入以 csr_matrix 形式保存

  2.計算 sink, tm, jump 三個轉移矩陣，sink 矩陣當某點的 out-degree 為零

時，則此點可轉移到任意矩陣且機率相等；tm 矩陣則依照讀入資料的 out link 建立轉移矩陣且每個向外的機率相等；jump 矩陣則代表某點可能轉移到任意一點且機率相等

3.在實驗中，任意跳轉的機率參數 d=0.15

4.因此 PageRank 的更新定義式為 n_pr = d*jump.dot(pr) + (1.0-d)*sink.dot(pr) + (1.0-d)*tm.T.dot(pr)

5.當變動量小於 EPSILON 則可視為馬可夫穩定狀態

- **SimRank**

1.將資料讀入以 csr_matrix 形式保存

2.將 csr_matrix 轉置之後，取出 Query 兩節點的 Row 可得指向兩節點的節點清單

3.首先初始 ranks 為 n*n 的 Identity Matrix

4.相似度的定義 = 共同節點/兩清單節點的排列組合

5.節點相似度的貢獻度會隨著遞迴次數而衰退，其衰退比率為 C 參數

6.相似度會不斷的傳遞，直到整體的差異值小於 EPSILON 才終止

VI. Result analysis and discussion

不論是 HITS 或是 PageRank 演算法，其結果大致上都與 link degree 呈現正相關，雖然 HITS 與 PageRank 最後的結果都會逐漸收斂，但 HITS 每個迭代的數值是震盪的，而 PageRank 則是沒有此種現象，此現象證實了 HITS 是由 auth 與 hub 兩條路徑交錯而成，且最後會收斂至一個定值，而 PageRank 使用馬可夫鍊的概念逐漸收斂，因此相較於 HITS 沒有劇烈震盪的現象發生。

VII. Questions & Discussion

- More limitations about link analysis algorithms

三者演算法各有利弊，對於 HITS 最大的詬病是必須跟隨 Query 進行運算，相較之下 PageRank 可以沒有這種困擾，此外 HITS 必須知道指向特定節點的節點清單有哪些，在現實生活中我們很難知道特定網頁被哪些網頁所指向，PageRank 僅需要透過 out links 使整個圖趨於穩定即可，相對 HITS 而言可行性較高，然而 HITS 與 PageRank 兩者都沒有針對相似度做判斷，因此可能受到不相關的網頁或廣告內容干擾結果。

- Can link analysis algorithms really find the “important” pages from Web?

不一定，如上題所述，可能受到不相關的網頁或廣告內容干擾結果，且惡

意使用者在知道演算法之後，可能使用個人網頁等方法相互強化，使特定網頁的分數得到提升，其結果未必準確。

- What are practical issues when implement these algorithms in a real Web?
  在現實生活中，鏈結的指向是會隨時間所更動的，因此每次都要更新整張圖，那麼花費是相當大的，此外僅依照鏈結來判定一個網站的好壞程度太過果斷。
- Any new idea about the link analysis algorithm?
  如上述問題，應該開發一個演算法可以動態的更動鏈結，並使結果可以重新依當前狀況局部重新計算，而非將針對整張圖重新執行一次演算法，此外應該加上一些使用者操作元素當作特徵進行判斷比較客觀一點，例如，點擊次數等等。
- What is the effect of “C” parameter in SimRank?
  C 在 SimRank 中代表影像力的衰退常數，換句話說，較近的共同父節點有較強的影響力，反之較遠的則因衰退函數的影響而指數遞減。
- Design a new link-based similarity measurement
  我認為 SimRank 在分母部分使用排列組合的方式，當某個網站被大量不相關的網站所指向會快速吸收其相似度，因此我認為分母的部分改成相加，而分子的部分改成 2*S(I(a),I(b))可能會好一些。