Social Network Analysis Homework 3

Team: 1
Student 1: r03922096 洪立遠
Student 2: b00902057 陳煥元
Student 3: b00902104 楊勛

How to run your program

Just type make to run our python code 我們有用到 pickle、sys、http.client、operator、numpy、scipy 等 librarys

Experiments for training data

我們只有紀錄最後兩次的 public，edge 沒有記錄下來；Forest Fire 因為結果不佳沒有記錄完。

Model	Dataset	Degree distribution KL	Closeness centrality	Edge attribute 1
Hybrid	Public data (node:100)	0.2046	590.110000	
Hybrid	Public data (node:300)	0.2064	379.900000	

Model	Dataset	Node attribute 1	Node attribute 2	Node attribute 3	Node attribute 4	Node attribute 5
Hybrid	Public data (node:100)	0.0002416840553	0.07622502658	0.06744258391	0.01478105399	0.0376256893
Hybrid	Public data (node:300)	0.004757778633	0.04924035683	0.07505610773	0.007756301002	0.218599239

Model description

Model	Description
Forest Fire	1. Forest Fire 就是先選擇一個點當作種子，再利用 Geometric Distribution 產生一個變數 W， Geometric Distribution 的 mean 為 1-pfpf，而 Pf 就是這個 function 所需要輸入的參數。 2. 產生 W 以後再從 Seed 點中選 W 個 neighbors 去 visit，被 visit 的點對他再產生一個 W 去選擇，一直重複下

	去直到找不到點去選為止。
Hybrid	1. 前一半的點選擇 max{(node degree – node graph degree) * node attribute ΔKL Divergence} 2. 後一半的點選擇 max{(node degree – node graph degree)}

Others (optional)

一、 Query model

我們首先嘗試運用現有的採樣模型，在閱讀了 Jure Leskovec 等人所著 Sampling in large graph [1] 這篇文章後，決定利用 Forest fire model。

Forest Fire 就是先選擇一個點當作種子，再利用 Geometric Distribution 產生一個變數 W， Geometric Distribution 的 mean 為 1-pfpf，而 Pf 就是這個 function 所需要輸入的參數。

產生 W 以後再從 Seed 點中選 W 個 neighbors 去 visit，被 visit 的點對他再產生一個 W 去選擇，一直重複下去直到找不到點去選為止。

論文建議參數 P 在應在 0 到 0.4 之間去做會有比較好的效果，應此我們採取 pf = 0.2 去做，但是他的 Degree distribution 之 KL Divergence 落於 0.8432。

另外有考慮由於 Numpy 所產生的 Geometric Distribution 之 mean 為 1pf 所以得方程式 1pf=1pf1-pf=10.2 得到 Pf 為 112 下去測試後其 KL Divergence 並沒有下降太多，仍然在 0.73 多

最後也有考慮過把第一次 uniform Select 的點 改成實際 Degree 最高的點下去跑，也沒有看到進步的效果。

評估 Forest Fire Model 效果不佳原因為，Query 得到的點多為 Degree 較多的點，所以需要想辦法 Scale Off。而參考的文章[1] 也有提到， Sampling Algorithm 在 Sample 超過 原圖 25% 的點時，才會有較佳的效果。

因為 Forest fire model 的效果不佳，所以轉念一想，乾脆每次 Query 都 Query degree 最大的 node，這樣可以得到最多的 information，就會更加接近原圖的 distribution。

不過單純從 nod 的 degree 最大去選，可能會有誤判的情況是，其實 query node 的一大部分 neighbor 已經在自己的 sub graph 中了，所以有效的 node

information 其實沒有 degree 那麼多了。所以改良後，我們選擇 (graph.node[n]['degree']-graph.degree(n))最大的點，作為 query node，而整體的 performance，也確實變得較好。

再來，我們考慮全域與區域的差別。全域即指每一次 query 都去找自己的 graph 中，node information 會最大的點來 query，而區域則是每一次的 query candidate 只限定在上一次 query node 的 neighbor 中。

我們發現使用區域的方式，效果較好，因為可能其物理意義是，先把這一個區域的 graph 作更深的了解，然後再 walk 到下一個點，而不像是全域中的做法，每次都在這個 graph 中跳來跳去。

完成了 degree 部分，也就是選區域中最大 information 的 node 來 query，我們打算來把 node attribute distribution 的部分加入演算法中。

我們用了ΔKL Divergence 作為 measure 的依據，也就是要選擇 node attribute ΔKL Divergence 最大的點來 query，為了要有 2 個 distibution 可以做 KL Divergence，我們把那個 node 拿掉，與不拿掉來做為比較。這樣的物理意義是，選擇 graph 中最奇怪的那個人，作為 query 的對象。

經 public graph 的測試，發現 node attribute range 愈小的 attribute，ΔKL Divergence 的效果愈差，也就是說單純最大 information 去 query，KL Divergence 也就會很小了。依據此結論，我們只有考慮了 node attribute 0 與 attribute 2 這兩個 node attribute 的ΔKL Divergence。

最終，與最大 information 這樣的概念融合，我們選擇 (graph.node[n]['degree']-graph.degree(n))*ΔKLDivergencey 最大的點來 query，其中ΔKLDivergencey 只有考慮 node attribute 0 與 node attribute 2。

二、Output strategy

1. degree distribution

考慮到由於我們在選點時都是挑 Degree 大的點所選擇，決定將我們產生好的 list 直接 reverse，把第一項跟最後一 Swap, 第二項, 跟倒數第二項 Swap…. 依此類堆。這樣做的目的是考量到我們每次都選到 Degree 高的點，所以 Degree 低的點反而比較少。

這樣做的好處是比較簡單，容易實作。壞處則是當 Query 次數提升時，KL Divergence 並沒有下降很多。

2. top 100 Closeness

作法本來是使用 Sample 好的圖經過 Networkx 本身的函示去計算得到前 10 最大的點當作答案，後來發現 Closeness 的公式為 $CC(v) = \frac{n-1}{\sum_{u \in V, u \neq v} d(v,u)}$ 。根據論文[2]，Degree 較大的點，分母會比較小，比較容易成為 Closeness 大的點，所以就將 Degree Sort 後輸出當作答案。效果不錯，在 public Graph 中，ATR 從 6869 降到 590.11，且 Query 越多次效果越好。

3. node attribute distribution

由於 sample 出的點並不能涵蓋到所有的 attribute，我們使用 add-1 smoothing 來估計沒觀察到的 attribute 之出現機率。

三、Reference

[1] Jure Leskove,and Christos Faloutsos, “Sampling in large graph”, Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, Pages 631-636

[2] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li, "Ranking of Closeness Centrality for Large-Scale Social Networks", FAW '08 Proceedings of the 2nd annual international workshop on Frontiers in Algorithmics