Practice of Social Media Analytics

HW2 – Community Detection

執行程式方法:

1. 使用 jupyter(此檔案為 ipynb)
2. 安裝套件 leidenalg、igraph、matplotlib、pandas、networkx
3. 執行每個 code block
4. 產生 csv 檔再放到 kaggle 即可

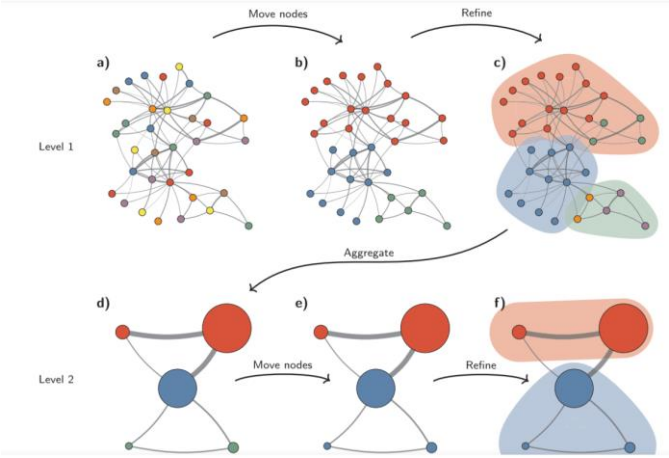
演算法流程:

1. 將訓練資料及測試資料讀進來
2. 對 train_data 建立 graph
3. 使用 leidenalg.find_partition 將 train_data 分成數個 community
4. 分別對測試資料中每筆資料的 Node1、Node2 使用迴圈判斷他們屬於哪個 community 並存至 node1_cluster 以及 node2_cluster
5. 判斷每筆資料 node1_cluster 以及 node2_cluster 結果是否相等，若相等，則他們為同一群
6. 將預測結果與 sample_submit.csv 合併產生完整可交至 Kaggle 的檔案

- 預測結果之 public leaderboard 分數

Leiden algorithm

- 示意圖



- 為 Louvain 演算法之進化版
- 改善了 Louvain 之 bad connected 的情況
- Steps:
  1. 局部移動節點
  2. 改善 community 劃分結果
  3. 基於改善後劃分情況凝聚網路，基於未改善時劃分情況初始化凝聚網路

la.ModularityVertexPartition

- 是一個用於在圖形 G 上執行基 Modularity 的節點分區函數
  1. 需先提供一個 graph
  2. 使用 modularity 計算節點權重
  3. 初始化時讓每個 node 自己形成一個 community
  4. 在更新時，會將每個 node 移動至其他 connunity，以改善 modularity。此演算法會計算 node 移動到其他 community 後之 modularity 增益，並選擇增益最大的移動。此過程會重複進行，直到無法改善 modularity 為止。