Project Report 2: The application of graph theory in studies of Social Networks

CS505: Intermediate Topics in Databases Spring 2018

By:

Vijay K. Shah and Sajjad Fouladvand

Date:

April 20, 2018

1. Introduction

In this project, we investigate the emails exchanged between employees of a company named *Barracuda National (BN)* aiming to extract knowledge out of the employee's communications with each other. The data set is a text file including 625 rows each includes an employee's name E_i and names of other employees who received an email from E_i at least one time during December 2017. To process the graph we use Python programming language as well as *NetworkX* as a powerful library facilitating working with graphs.

2. Pre-processing

As we want to construct a graph (adjacency matrix) and ten process the graph, we map the dataset from string to integer numbers. We extract all unique names (1000 names in total) and assign each of them a unique integer code and then replace the names with their relevant code in the original data set. The resulted dataset has the same length and dimension as the original data set except it includes integer codes instead of strings (names). We also create a dictionary file which we use to decode the integers to the names.

3. Overview of the graph

The graph include 1000 nodes and 95,117 edges. Figure 1 shows the graph in which red points are nodes and black parts show the edges. Since the graph contains many nodes and edges there are many overlapped areas in the figure. As the graph in Figure 1 includes many nodes and edges, we plot the in-degree and out-degree histogram to give more intuition about the characteristics of the graph. Figure 2 and 3 show histogram of in-degree and out-degree of the nodes. In Figure 2 (or 3) x-axis is the in-degree (or out-degree) number and y-axis is the number of nodes with relevant in-degree (or out degree). For example, point like *h(100, 39)* on these histograms means there are 39 nodes in the graph with in-degree 100. Note, Figures 2.a and 2.b are almost the same, but 2.b is more focused on nodes with in-degree less than 200. Generally speaking, Figures 2. B and 3.b show that most of the nodes in the graph have in-degree of 100±25 and out-degree of 90±25.

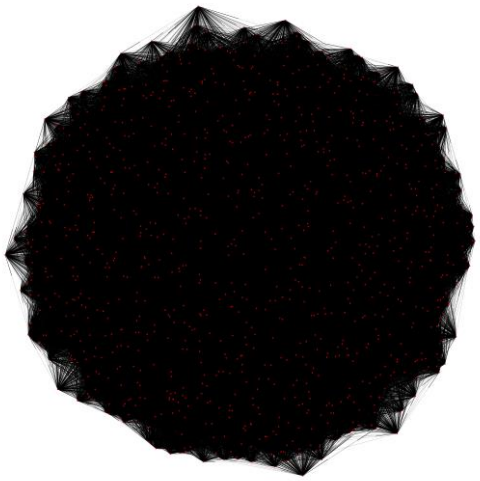

Figure 1. **The social network of BN Company.** Red points are nodes which indicate employees of BN and black part is edges which shows who emailed who.

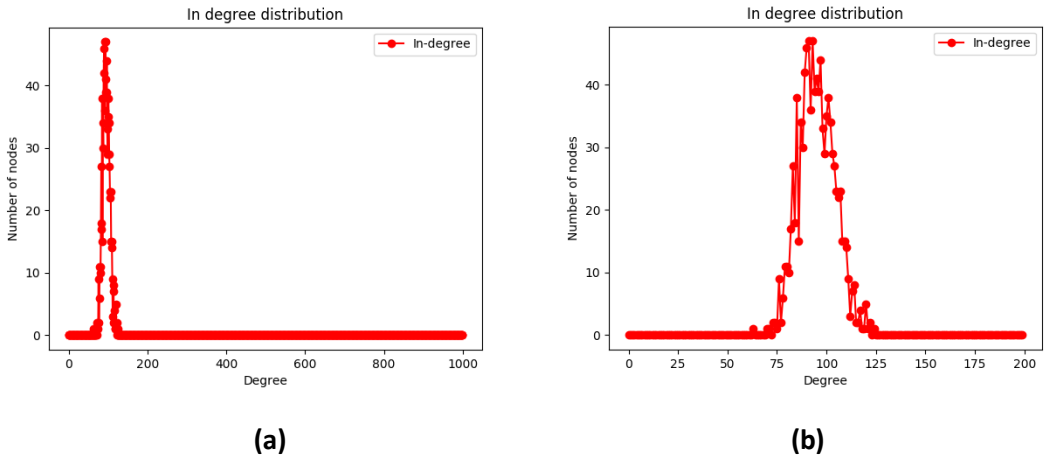

Figure 2. **Histogram of in-degree.** (a) In-degree distribution considering all nodes in the graph. (b) In-degree distribution considering only nodes with in-degree < 200.

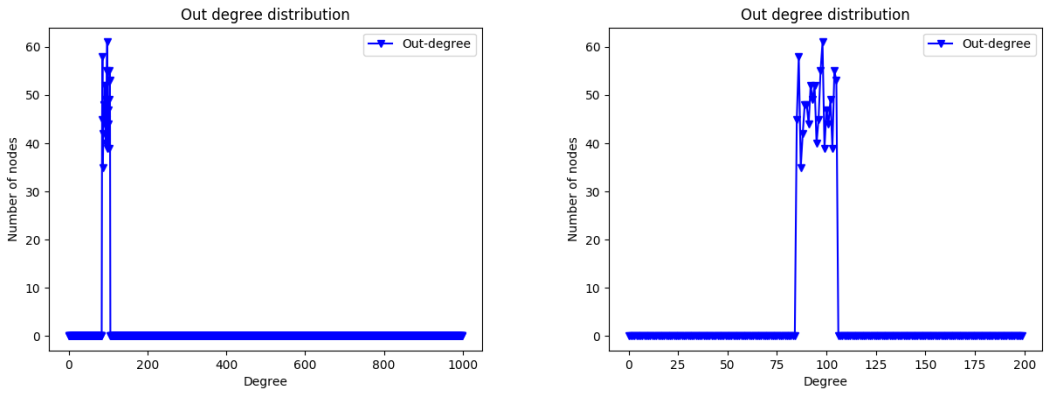

Figure 3. **Histogram of out-degree.** (a) Out-degree distribution considering all nodes in the graph. (b) Out-degree distribution considering only nodes with Out-degree < 200.

Answer to question 1: *List of possible leakers*

As it is describes in the question 1, a leaker is someone who probably negotiated with *Wood* which means the leaker, whoever he/she is, signaled *Wood* by email during December 2017 at least once. Therefore, we process the graph to find nodes which has an edge toward *Wood* (emailed *Wood* at least once) and also an edge back from *Wood* (wood also emailed them). In fact, if an employee E_i emailed *Wood* and *Wood* also emailed E_i at least once we consider E_i as a leaker. The output of our implementation in python is provided in appendix A. Figure 4 shows a sub-graph of the entire graph in which only leakers and *Wood* are shown. In this subgraph you can clearly see how leakers are connected to wood. As it can be seen from this figure, the list of possible leakers is L={MORTON, AYERS, BANKS, MORENO, RICHARDSON, BRANCH, WEST, MCDANIEL, HUDSON, HOWE}

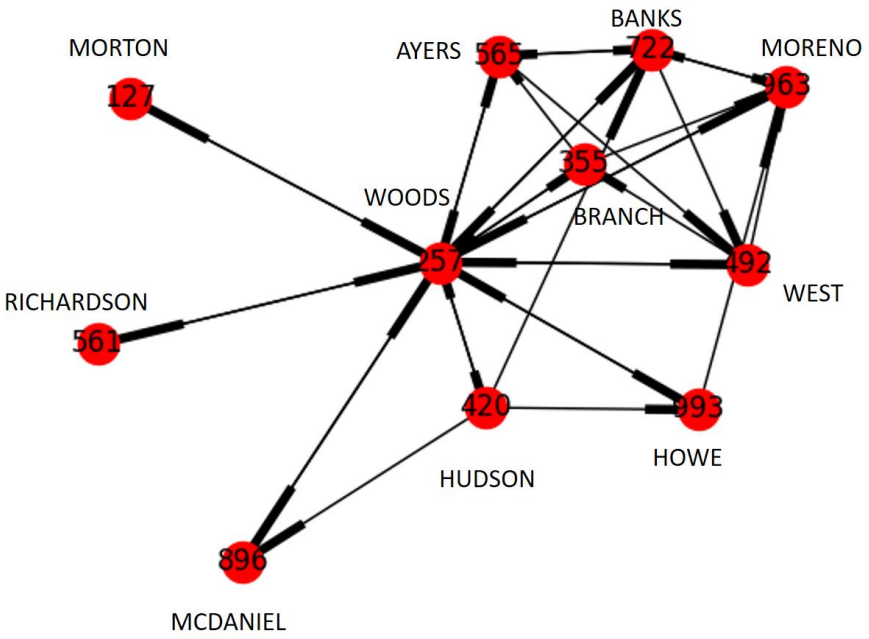

Figure 4. Sub-graph including leakers and Wood. This sub-graph gives an intuition on how possible leakers are connected to Wood and also how they are connected to each other.

Answer to question 2: ***List of all cliques of largest size***

Finding cliques of largest size in an *undirected* graph is simpler to implement compared to finding them in a directed graph. We first describe how to convert the undirected graph *G* to a graph *G'* so that the converted graph satisfies two constrains: 1) *G'* is a directed graph, 2) List of all cliques of largest size in *G'* is equivalent to the list of all cliques of largest size in *G*. To create *G'* , we brows all pairs of nodes in *G* such as *u* and *v* and if there exist both edges *(u, v)* and *(v, u)* in graph *G* then we add an undirected edge (u, v) to *G'*. **Algorithm 1** describes how we implant this idea.

Algorithm 1- Convert a directed graph *G* to an undirected one *G'*
1: G'. nodes = G. nodes
2: For all *u* in *G. nodes*
3: For all *v* in *G. nodes*
4: if *u != v* and *G. has_edge(u, v)* and *G. has_edge(v, u)*
5: add *(u, v)* to *G'. edges*

Graph *G'* connects two nodes u and v if and only if there are two edges (u, v) and (v, u) in *G*. Since all nodes are connected to each other in a clique, the cliques in both *G* and *G'* are the same. Therefore, finding list of all cliques of largest size in *G* is equivalent to finding them in *G'*. Then, we use NetworkX library to recursively find list of all cliques of largest sizes in *G'*. The entire list of all cliques of largest size includes 114 cliques and the list is provided in appendix A. Figure 5, visualize one clique of largest size including {DUKE, OCHOA, GRAHAM} as an example.

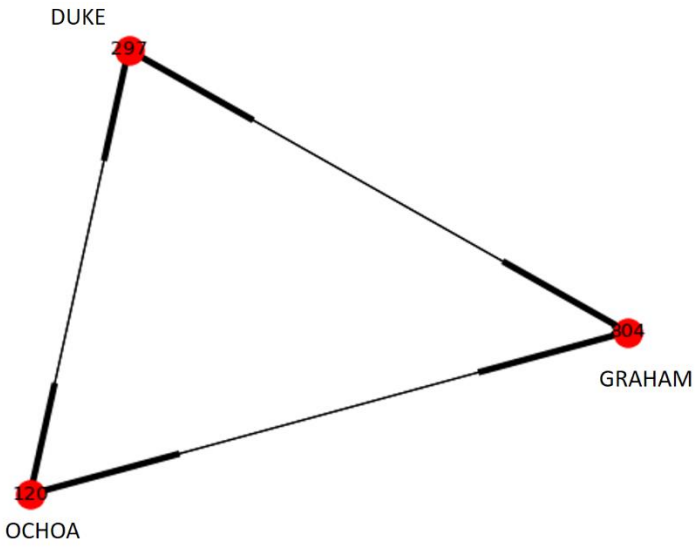

Figure 5. Visualizing one clique of largest sizes.

Answer to question 3: ***Find a butterfly in the graph***

To answer this question we benefit from the list of cliques we already found while solving question 2. We brows all possible pairs of cliques such as C_i and C_j and then if they both have length 3 and are connected in one point we consider them as a butterfly. **Algorithm 2** generally describes how we implanted this idea in python.

Algorithm 2- Finding butterflies

```
1: list of cliques= Algorithm 1(G)
2: For all cliques Ci  in list of cliques
3:     For all cliques Cj  in list of cliques
4:             Butterfly_Candidate = Ci . nodes + (Cj .nodes – Ci .nodes)
5:             if Cj != Cj  and length(Butterfly_Candidate) == 5
6:                     butterfly_list.append(Butterfly_Candidate)
```

Complete list of butterflies are provided in appendix A. Figure 6, shows one butterfly including {VARGAS, GARRISON, THOMPSON, GHUNG, DUNCAN}. As it can be seen from the figure, the butterfly includes two cliques with size 3 and the two cliques have only one node in common.

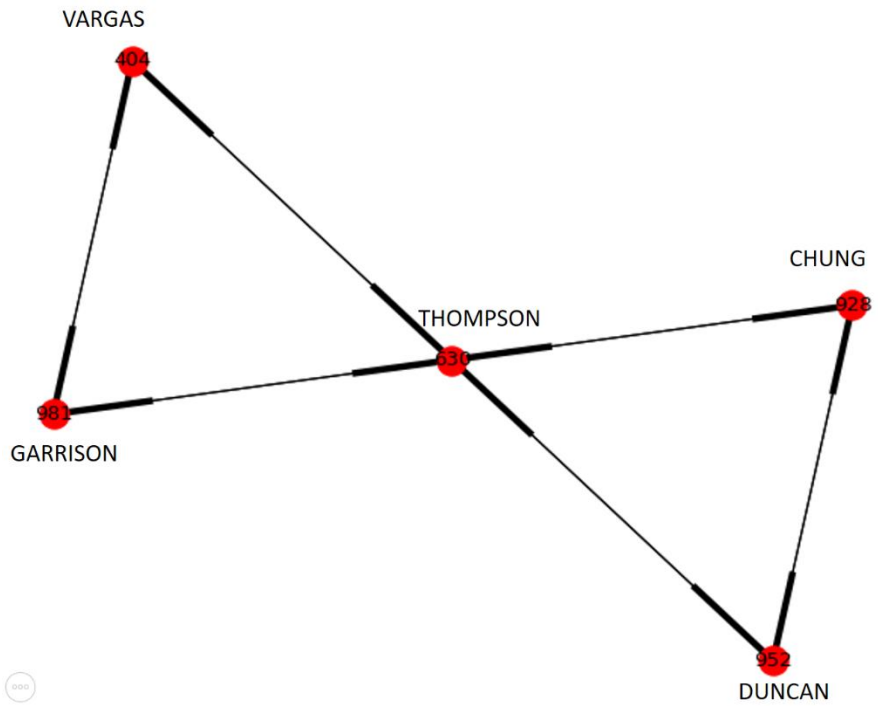

Figure 6. A butterfly.

Answer to question 4: Find the most influential people in the graph

In this question we find the most influential individual in the graph. In fact, we process the edges and find the people who have a centrality roles since it is reasonable to consider a person as an influential individual if he/she has connections to many others which means he/she might has more access to information, or more prestige than those who have fewer connections. To address the mentioned question and find the most influential individuals we utilize *Degree Centrality*. Degree centrality of a node is the number of edges connected to that node (Figure 7).

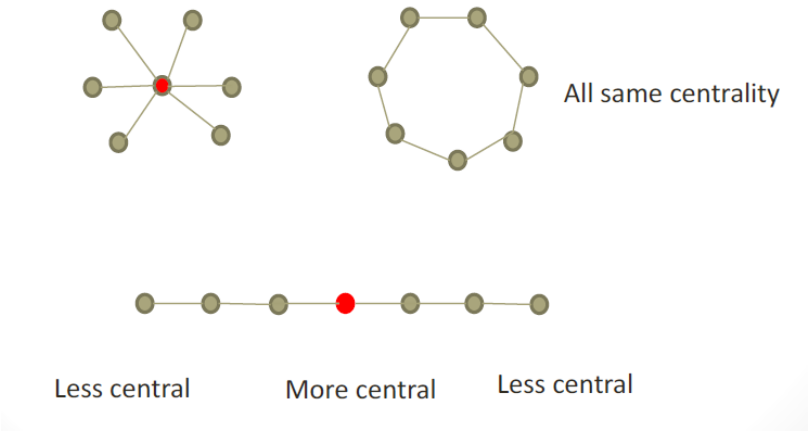

Figure 7. **Degree of centrality.** Degree of centrality shows that how much a node is important in a social network.

The graph in this project is a directed graph and so we calculated both in-degree and out-degree centralities. Figure 8 shows the distribution of in-degree and out-degree centralities for our graph. In figure 8, to make the figure simpler to visualize, we used people's codes instead of their actual names; for example, since the coordinates of *695* in figure 8 is *(0.064, 0.097)* then the person with ID *695* (COOLEY) has in-degree centrality of 0.064 and out-degree centrality of 0.097. As it can be inferred from the figure, majority of people are actively involved in communications of the company. However, we scrutinize the graph more by looking at the top-10 individuals with highest in-degree and out-degree of centrality.

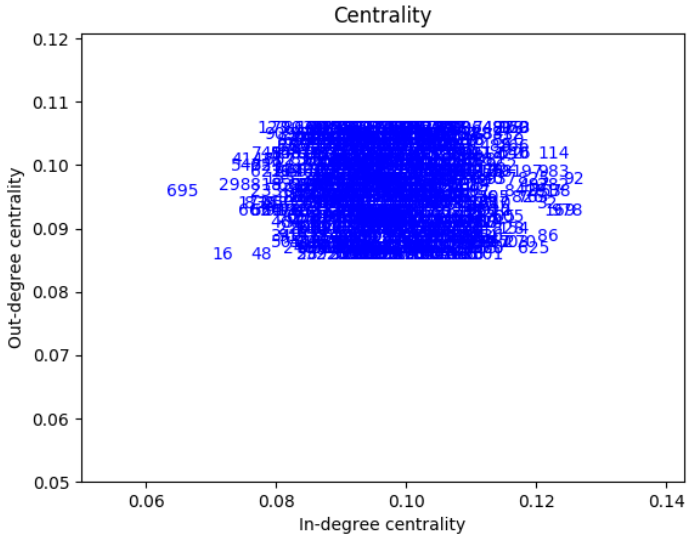

Figure 8. Distribution of in-degree and out-degree centrality for all of the individuals.

Appendix A includes our program's output for top-10 individuals in terms of both in-degree and out-degree centrality. Top-10 people with highest in-degree centrality include *A={COLLIER, WARREN, SAVAGE, COSTA, BATES, SANCHEZ, SNYDER, FRY, FERREL, GALVAN}*. That means people in *A* are top-10 persons who were contacted by other people in the network the most. In fact, they are the most influential persons in the network since larger number of other employees probably need to email them and get in touch with them. On the other hand, top-10 individuals in term of out-degree centrality include *B={WOOD, CHARLES, ARMSTRONG, SEXTON, GARRET, HESS, BRADSHAW, KIRBY, MATHEWS, HUANG}* which means these persons contacted larger number of other people in the network during December 2017. They are also influential since they contact many other people in the network. Another interesting point is that *Wood* is listed as an influential person with a high out-degree centrality which is consistent with what we saw in the first question, because *Wood* is the one who sold a trade secret of HN on dark web and he probably has been contacting many other people to find a potential leaker. We also want to highlight that the intersection of *A* and *B* is empty which means there is no one in the graph who has both high in-degree and high out-degree centralities. Therefore, people are either contacting higher number of other individuals or they are receiving emails from higher number of other individuals but not both.