Project 2 Report

Team 1

William Dahl, 001273655

Jerrison Chang-Sundin, 001357402

Tianhao Wang 001441757

ICSI 532 - Network Science

Dr. Chelmis

The University at Albany, SUNY

Abstract

In this paper, we analyze a real-world communication network of email correspondence. We constructed a weighted directed graph out of the email correspondence from the given dataset. In this dataset, nodes represent email addresses and the directed edge depicts a sent/received relationship. Each edge e_{uv} denotes the number of emails sent from node u to node v over the entire dataset. Next, we extracted the 1.5 ego network for each email within the network, where each egonet was considered as an undirected graph. Lastly, we considered the network to be a temporal network in which we constructed temporal graphs out of the data set so that the union of the temporal graphs would be the graph as a whole. Each temporal graph was directed, weighted, and represents a daily sample of the network. Each edge e_{uv} in the temporal graph denotes the number of emails sent from node u to node v within the same day.

Introduction

The Initial Analysis

In this paper, we analyze a real-world communication network of email correspondence. We will construct a weighted directed graph out of the email correspondence from the given dataset. In this dataset, nodes represent email addresses and the directed edge depicts a sent/received relationship. Each edge e_{uv} denotes the number of emails sent from node u to node v over the entire dataset.

We will start our analysis by calculating the following summary statistics of the network:

1. Number of Nodes
2. Number of Edges
3. Number of bidirectional edges
4. The min, max, and average in-degree for each node
5. The min, max, and average out-degree for each node
6. The min, max, and average total degree for each node
7. The diameter of the network

Using the degree distributions calculated earlier we will Plot the in, out, and total degree distributions for each node in the network. All three distributions will be plotted on the same plot on a log-log scale along with each distribution's corresponding least-squares regression line.

Using the power-law calculated earlier for the least-square regression of each degree distribution, we will further analyze the degree distributions by comparing them to an exponential and log-normal distribution and seeing which fits better to the degree distributions.

Extracting 1.5 Ego Networks

Next, we will extract the 1.5 ego network for each email within the network, where each egonet will be considered to be an undirected graph.

To analyze each ego net we will compute the following features for each ego net within the network:

1. Number of nodes within the ego net
2. Number of Edges within the ego net
3. The total weight of the ego net
4. The principal eigenvalue for the weighted adjacency matrix of each ego net

To help the analysis of the ego networks we will plot the number of edges versus the number of nodes within each ego network on a log-log scale along with the least-squares fit on the median values for each ego net after logarithmic binning is applied to the x-axis. We will highlight the ego nets that correspond to the stars and cliques by plotting lines of the same power-law with slopes of 1 (for stars) and 2 (for cliques). In a separate figure, we will analyze the principal component eigenvalue, λ, by the total weights, W, of each ego net. We will then check if we can fit a power law to either distribution.

To further our analysis, we will find the most "out-of-the-norm" ego nets. To do this, let x_u and y_u denote the observed x and y value for a node u for a particular feature pair $f(x, y)$. Using the power law equation $y = Cx^a$ for $f(x, y)$ we can compute the "out-of-the-norm" score o(u) for of the egonet using the following equation:

$$o(u) = (max\{y_u, Cx^a\} / min\{y_u, Cx^a\}) \log(|y_u - Cx^a| + 1)$$

This metric will penalize each node for both the number of times it deviates from its expected value (Cx^a) and its amount of deviation. Thus, when o(u) = 0 $y_u = Cx^a$. We will denote the top 20 scorings "out-of-the-norm" ego nets for both cases of $f(V, E)$ and $f(\lambda, W)$.

Temporal Representation

Lastly, we considered the network to be a temporal network in which we constructed temporal graphs out of the data set so that the union of the temporal graphs would be the graph as a whole. Each temporal graph was directed, weighted, and represents a daily sample of the network. Each edge e_{uv} in the temporal graph denotes the number of emails sent from node u to node v within the same day.

We will analyze the evolution of the network over time by plotting the sizes of both the strongest and weakest largest connected components for each temporal graph. We will also analyze the density of each temporal network over the entire set to see if there is densification or undensification within the network over time. We will also inspect the average clustering coefficient for each temporal network and analyze what the implications of an increasing or decreasing clustering coefficient are for the network.

Methodology

Our network analysis was done using python along with `networkx` to create and analyze our networks. We also used `numpy` to perform mathematical calculations not readily available in base python, and we used `matplotlib` to render our plots and figures for our analysis.

The Initial Analysis Methodology

To begin constructing the network we created a directed graph using `networkx`. We then read from a text file containing our data set. The data within the text file was of the form `<UnixEpoch sender receiver>` on each line. Each line is read from the file, stripped of white space, and split into spaces. The sender and receiver were added to the graph as an edge. To calculate the weight of each edge, when an edge is added for the first time, the weight of the edge is set to 1. When an edge appears again within the data set the weight of the edges is simply incremented by 1. This is done by using a try/except to try and increment the weight of the edge and if an exception is thrown, because the edge is not already in the graph, then the edge is added to the graph with a weight of 1.

To compute the summary statistics, networkx' s functions were used. The number of nodes and edges were simply computed by using the `number_of_nodes()` and `number_of_edges()` function. To compute the bidirectional edges, each edge e_{uv} within the graph was gone over and the edge e_{vu} was checked to be in the graph. After all the bidirectional edges were counted the total must be divided by two because each bidirectional edge was counted twice. The in, out, and total degrees for each node in the network were computed using the `networkx` functions `in_degree(), out_degree(),` and `degree()`. min and max values were then extracted from the degree distributions using the `min()` and `max()` functions. The average degrees were computed by summing the degrees of each node in the distributions and divided by the number of nodes computed earlier. For the diameter, networkx `diameter()` function takes a long time to compute the diameter without giving information during the computation, instead, we iterate through each node and compute its eccentric value

and log it in a txt file so that we have a point to pick up and resume in case computer crashes. Moreover, we pick the largest weakly connected components to compute the diameter.

The in, out, and total degree distribution is plotted by using the degrees calculated earlier and using the `Counter()` function from the `collections` modules then the degree and count are separated by using the `zip()` function on the items in the degree distribution. The least-square regression coefficients *m* (the slope) and *c* (the y-intercept) is then calculated by using numpy's `polyfit()` function on the log of the degrees and the log of the counts. The fitted *y* value is then calculated by using the equation:

$$e^{mlog(d) + c}$$

Where d is the degree of the node. The in, out, and total degree distributions are then plotted on a log-log plot using matplotlib `loglog()` function along with each respective least squares regression.

Extracting 1.5 Ego Networks Methodology

To begin extracting the 1.5 ego networks we go through each node u in the graph and extract the 1 ego network for node u. The resulting ego network is then converted to an undirected graph by using the networkx function `to_undirected()`. Next, in order to calculate the 1.5 ego network, we check if an edge exists between node u's neighbors within the original network. If the edge exists we add it to the ego network making it a 1.5 ego network. The weighted total for each ego net is calculated by summing all the weights together in the ego net. The eigenvalues are computed by first computing the Laplacian matrix of the ego net using nextworks `normalized_laplacian_matrix()` functions. The eigenvalues are then

computed by using numpys `eigvals()` function and the principal component eigenvalue is extracted by summing all of the eigenvalues together.

The number of nodes are plotted against the number of edges within each ego net along with the best fit least-squares regression line. The line representing the ego nets that are a star shape is plotted using the y-intercept *c* computed from the linear regression line with a slope of 1:

$$e^{log(d) + c}$$

Similarly, the line representing the ego networks that are cliques is plotted with a slope of 2:

$$e^{2log(d) + c}$$

The "out-of-the-norm" score for each ego net is computed by calculating the expected value of y using the coefficients, m as the slope, and c as the y-intercept, computed earlier when getting the least-squares regression line. The expected value, y', is:

$$y' = cx^m$$

We then compute the "out-of-the-norm" score using the equation:

$$o(u) = (max\{y, y'\} / min\{y, y'\}) \, log(|y - y'| + 1)$$

The "out-of-the-norm" scores for each egonet are then sorted and the top 20 scoring ego nets are labeled on the plot for the function *f(V, E)* and *f(λ, W)*.

Temporal Representation Methodology

The temporal representations of the network are created by using the UNIX epoch to distinguish from different days when the emails were sent. Each section represents a different

day of correspondence between email addresses. Each temporal graph is built in the same manner as before for the entire network. The strong and weak components were computed using the `networkx` functions `strongly_connected_components()` and `weakly_connect_componets()` respectively. The largest component from each was then selected by getting the component with the largest length. The density of each temporal graph is calculated using the `networkx` function `density()` and the average clustering coefficient is computed using the `average_clustering()` function. The evolution of the strong and weakly connected component over the temporal graphs is plotted by each day. The densities of each and the average clustering coefficient are also plotted in separate figures.

Results

The Initial Analysis Results

Starting with the calculation of the summary statistics of the network we found the network to have (Weekend data excluded):

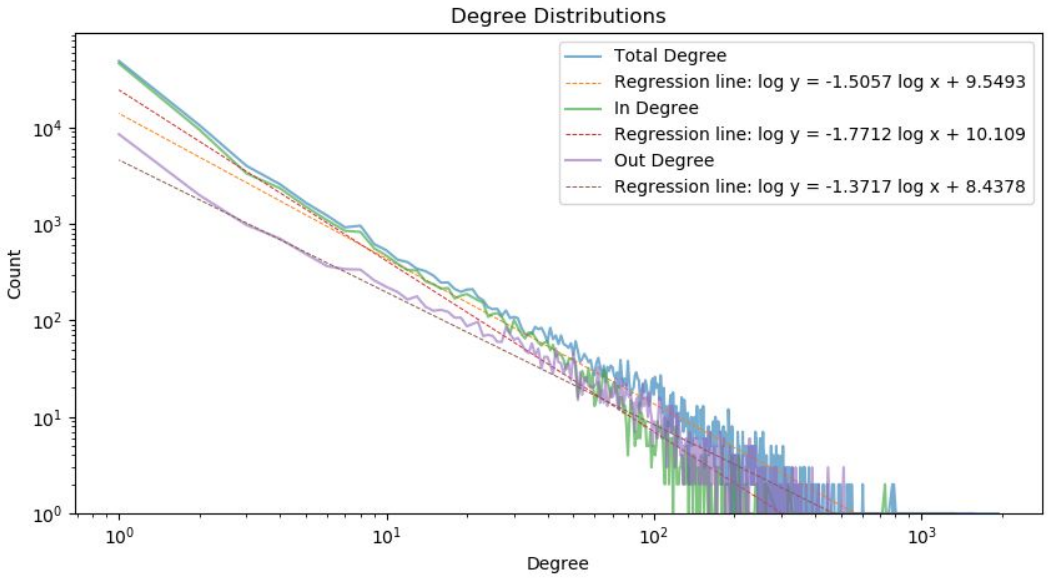
- Number of nodes: 80691
- Number of edges: 347589
- Number of Bidirectional edges: 29684
- Degree information:

	In degree	Out degree	Total degree
min	0	0	1
average	4.30766	4.30766	8.61531
max	1333	1566	1940

- Diameter: 11

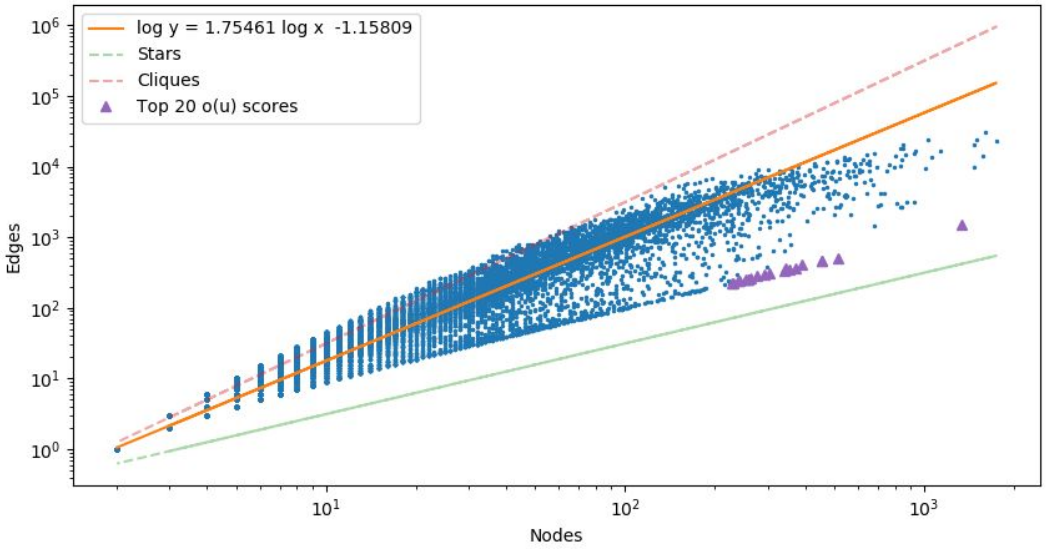
As discussed in the class, when calculating the diameter of a graph, we only consider the connected component (else the diameter will be infinite and is not useful for analysis).

When then plotted the distribution of degrees, in-degrees, and out-degrees of the nodes within the network on a log-log scale:

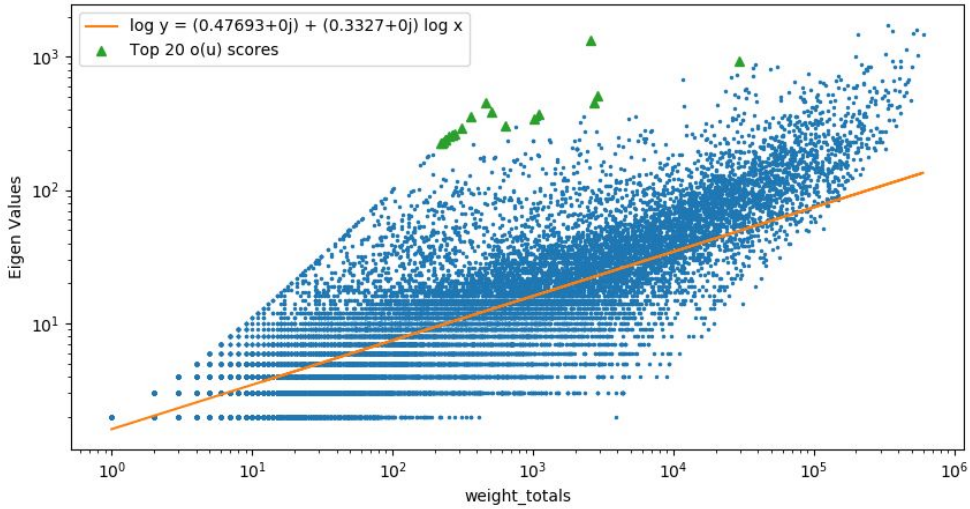


Here, we can see that the distribution of degrees follows a power-law distribution. The fitted power law is similar to that of a lognormal distribution as the distribution of the degrees of the network range from 1 to 1960 with an average degree of 8.432433. Resulting in a distribution similar to that of a lognormal distribution.

Extracting 1.5 Ego Networks Result

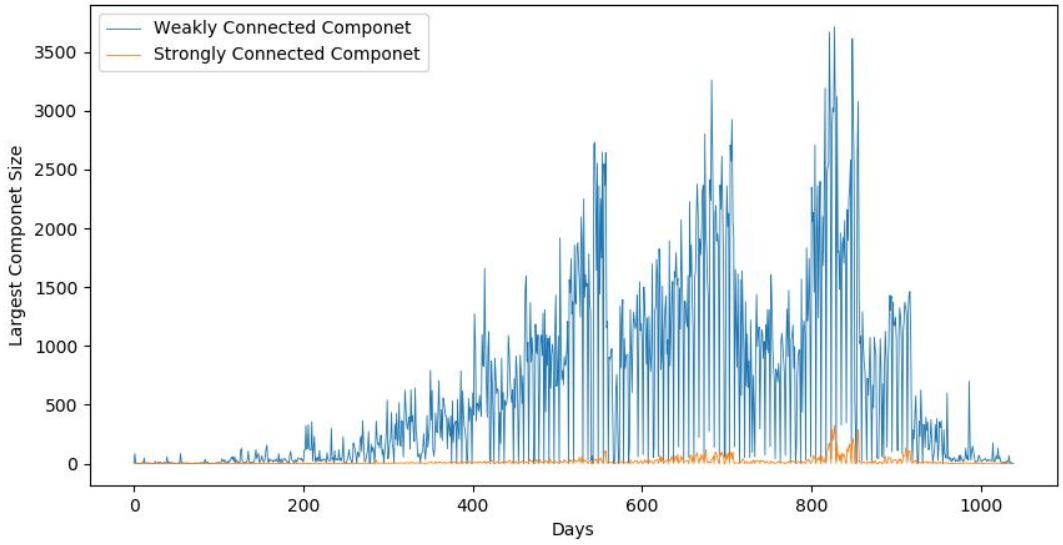


The figure shows the number of nodes vs the number of edges for each ego net. This graph shows that the average number of edges in an ego net is about 1.5 times the number of nodes in the ego net. The ego nets that are cliques are seen when the number of nodes in the ego net is low and the number of edges is twice that of the number of nodes. There also appears to be only one ego net that is a star shape where the number of nodes is low and the number of edges is equivalent to the number of nodes in the ego net. The exponent of the power-law fit to the figure is equal to the slope of the regression line and thus is approximately 1.5. The top 20 ego nets with the highest "out-of-the-norm" score can be when the number of nodes in the ego net is high but the number of edges is relatively low.

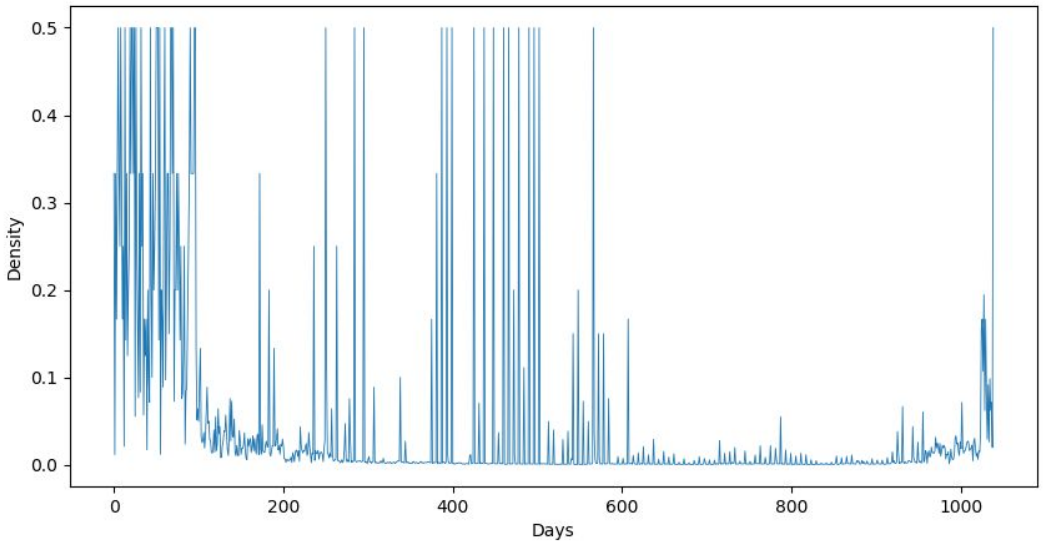


The above figure shows the weighted totals versus the principal eigenvalue for each ego net. This graph shows that the fitted power-law exponent for the figure is 0.3327 and the top 20 scorings “out-of-the-norm” scores can be seen when the principal eigenvalue is much higher than the total weight for the ego net.

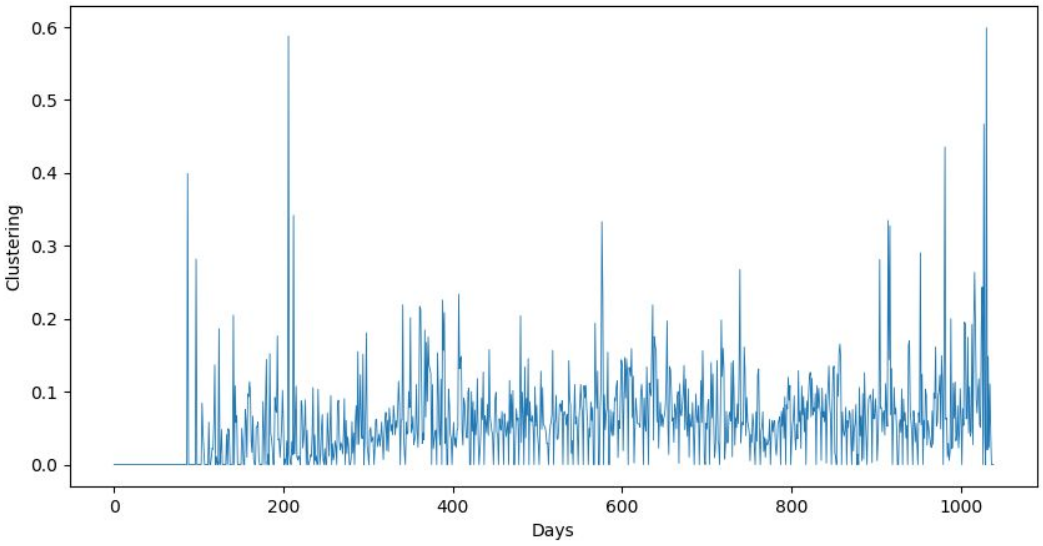
Temporal Representation Result



The figure above shows the change in the largest strongly and weakly connected components by day. We can clearly see that the size of the largest weakly connected component is much larger than the size of the largest strongly connected component. For every day the graph the network does appear to be weakly connected but there are some days when the network is not strongly connected.



There seems to be an un-densification trend in the graph as each day goes by. The density of the network stays high in the earlier days but stays low for longer periods of time as the days go by in the graph.



The average clustering coefficient appears to be increasing slightly as the days go by. This implies that more people are emailing new people who were not emailing before. This could result in fewer cliques forming over time as people begin to email more people than they were original.