# DS210

DS210 Final - Sam Offsey

## Purpose

In a large company, it can be difficult to design an organizational structure that effectively represents the way employees collaborate and communicate. Organizational network analysis aims to use graph network analysis to create better teams. In this project, I will use rust to calculate centrality metrics in order to determine which employees are most central to the organization and whether the existing company structure reflects these interactions.

## Dataset

The dataset used in this project was provided by Harvard Dataverse. The graph dataset represents approximately 83,000 email communications sent between 167 accounts for a manufacturing company over a 9 month period. The csv file “communication.csv” represents this graph as a list of directional edges. The csv file “reportsto.csv” contains the official organization structure as a list of directional edges.

## Implementation

To represent the graph, I created a struct called “EdgeWeightedDigraph”. This is an array that holds the strength of the connection from node i to node j as entry ij in the array. The adjacency array is an efficient way to hold the graph information.

Next I created an impl for this struct that contains the different functions that will interact with it:

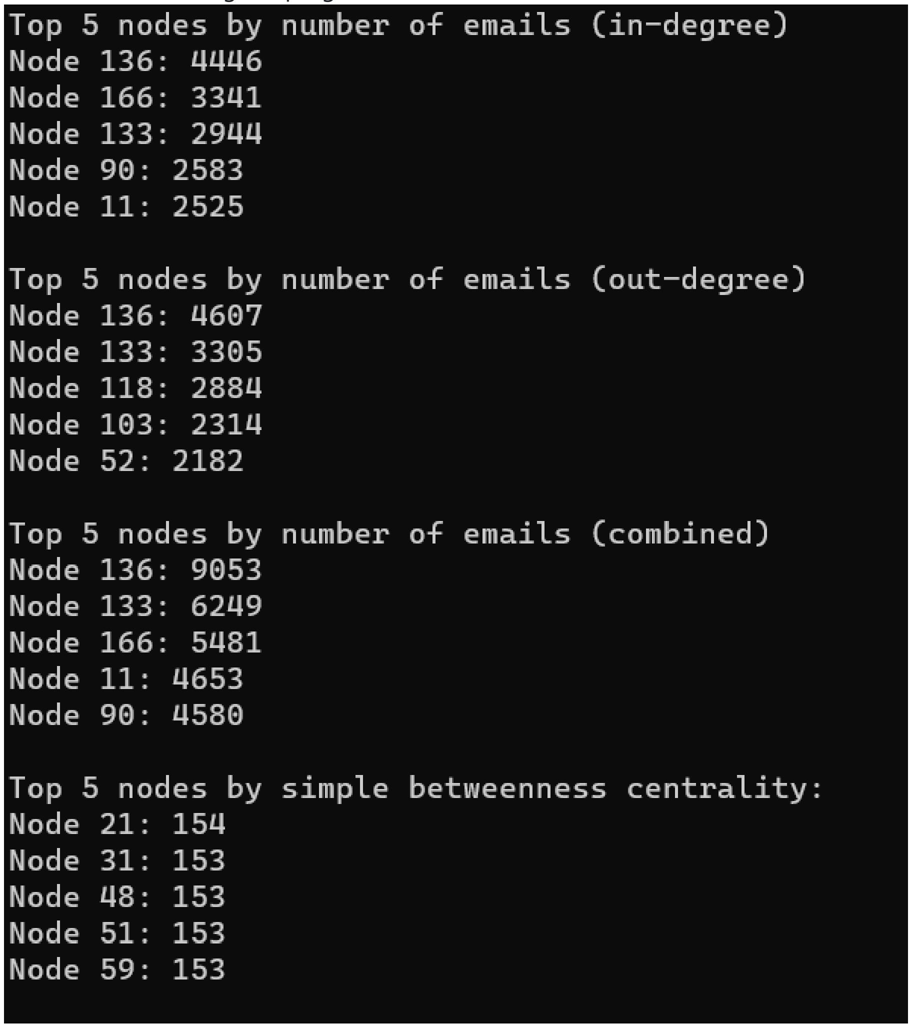
- New, which initializes the array of the given size with zero entries.
- Add_weight, which takes a start and end node and adds one to the weight for that edge.
- Get_weight, which takes a start and end node and returns the existing weight in the graph
- Degree_centrality, which takes a type (either in-degree, out-degree, or combined) and returns a vector with the strength of all ingoing, outgoing, or combined edges for each node.
- Simple_betweenness_centrality takes the graph and returns a vector of tuples representing a Simple_betweenness_centrality for each node. This is the most complicated function so I annotated the code as well. Basically, the function creates a hashmap to store the number of shortest paths through each node. Then a breadth-first search is performed for each node, where the visited nodes are stored in a hash set, and nodes to visit are stored in the queue vector. Finally, a point is added to the betweenness score for each shortest path that passes through the node.

In the main function, I create an EdgeWeightedDigraph datatype, read the CSV file, and use the add_weight function to fill the adjacency matrix with the weights from the dataset. Then I calculate the in-degree, out-degree, combined, and simple betweenness centrality for the graph and print out the top 5 nodes by centrality score.

Finally, I created a mod for tests which ensure that each function within the impl works as expected.

## Results

The results of running the program are below:



```
Top 5 nodes by number of emails (in-degree)
Node 136: 4446
Node 166: 3341
Node 133: 2944
Node 90: 2583
Node 11: 2525

Top 5 nodes by number of emails (out-degree)
Node 136: 4607
Node 133: 3305
Node 118: 2884
Node 103: 2314
Node 52: 2182

Top 5 nodes by number of emails (combined)
Node 136: 9053
Node 133: 6249
Node 166: 5481
Node 11: 4653
Node 90: 4580

Top 5 nodes by simple betweenness centrality:
Node 21: 154
Node 31: 153
Node 48: 153
Node 51: 153
Node 59: 153
```

The description file tells us that the CEO is represented by node 86. Nodes 7, 27, 36, 69, 70, 85, 104, 121, 148, 156, and 163 represent the executives who report directly to the CEO. What is interesting about the results is that the most central employees from the analysis are not in this group. This suggests that some of the lower level employees are very central to the network and might benefit from more recognition. One problem with this hypothesis is that since this is a manufacturing company and the network data came from emails, some employees might be central through verbal communication but not send many emails from the type of job they have. Managers should use this information in combination with their own understanding of employee relationships and structures.

## Dataset

Michalski, Radosław, 2020, "Manufacturing company email metadata and corporate hierarchy", https://doi.org/10.7910/DVN/6Z3CGX, Harvard Dataverse, V2, UNF:6:/mJDDYEUXZ0pb1bvKiKttA== [fileUNF]

## Releases

No releases published
Create a new release

## Packages

No packages published
Publish your first package

## Languages

● Rust 100.0%

## Suggested Workflows

Based on your tech stack

**Actions Importer** Set up

Automatically convert CI/CD files to YAML for GitHub Actions.

**SLSA Generic generator** Configure

Generate SLSA3 provenance for your existing release workflows

**Rust** Configure

Build and test a Rust project with Cargo.

More workflows Dismiss suggestions