DS210 Final Project Report

Computing Jaccard similarities to find relationships in a research institution graph network.

The Dataset

The dataset I did my data analysis on for this project is on email data for a European research institution. The dataset has animalized information of all of the emails sent out and received in the graph network and doesn’t show contents of the emails. Hence each node u that is connected to v represents an email being sent out from person u to v. It is an undirected graph where it just states that an email was sent between the 2 parties.

The Problem

The main problem the data analysis aims to tackle is “Are the friends of friends, my friends”. As a result, I aimed to find how similar are the sets of neighboring vertices for 2 nodes that are connected to each other. The importance according to the dataset arises when you think about how connected this European research institution is. It answers questions like if 2 members of the research institution have a relationship, are they able to connect with the other contacts of the contact for the research institution. I also took it further to see how far the connection would take for the contacts of contacts have relation to each other; how many nodes separation it takes between 2 vertices.

The Proposed solution

The way I computed the solution to my problem was through using an amalgamation of Breadth-first search (BFS) algorithm and Jaccard Similarity score. I used the BFS algorithm to find the 2 vertices that have connections. So, the main one I looked at for the problem of this dataset is computing a jaccard similarity score for 2 vertices with a BFS path of 2. This allows the “friend of friend” to occur, and then see how connected the neighboring vertices are.

Jaccard similarity is a measure used to compare the similarity and dissimilarity between two sets. It's particularly useful when you want to compare how similar two sets are, regardless of their order or repetition.

The Jaccard similarity coefficient J(A,B) between 2 sets A and B is defined as the ratio of the size of the intersection of the sets to the size of their union:

In other words, it's the number of elements shared by both sets divided by the total number of distinct elements in both sets.

The Jaccard similarity coefficient ranges from 0 to 1. A value of 0 indicates no similarity (i.e., the sets have no elements in common), while a value of 1 indicates complete similarity (i.e., the sets are identical).

After computing Jaccard similarities for all connecting vertices with a BFS shortest path of 2, I computed the Mean, Max, Vertices associated with the Max similarity score. The point of doing this was to find interesting discoveries and get a better sense of the connections between 2 connecting vertices.

The Code

The code I have has 3 main modules: jaccard\_similarity.rs, jaccard\_stats.rs, and graph.rs

**Graph.rs**

This module reads an adjacency list from the txt file and import it into a Graph that is stored into a HashMap. The graph is created using a struct with adjacency list being a value, functions besides reading the graph are done as methods in the impl block. Using the HashMap and adjacency list, it computes the BFS shortest path from a source vertex to a specified vertex.

**Jaccard\_similarity.rs**

This module is where the main heavy computation occurs for the project. Using the keys from the Graph HashMap, the function computes the Jaccard similarity score of 2 vertices with the shortest path of 2. Later in the project, I will see how similarity score changes when increasing the shortest path. This is done by changing the distance variable from 2 to other values. Outputting this function in the main will give the Jaccard similarity score for all connecting vertices with the shortest path of the specified n value. It stores the output in a HashMap for other computational purposes

**Jaccard\_stats.rs**

Finally, this module has 2 functions. The first function computes the mean, max and associated max vertices of all the jaccard similarity computed in Jaccard\_similarity.rs storing it into a tuple. The other function computes what percentage of the Jaccard similarity scores are above thresholds 0.1 to 1.0 in 0.1 increments. The importance and findings of this will be discussed later.

**Main.rs**



The main.rs file is how the code is run. As you can see, all the modules discussed have been imported in order to output each important aspect of this project. You can select which aspect you want outputted by commenting out the print statements of everything that isn’t wanted using /\* ~ \*/. This has been done for the similarity scores as it would fill up the terminal since it displays a lot of output. Finally, simply doing cargo run will give any desired output, after putting the dataset in the filename variable (“final\_data.txt” was used for mine).

The Output

The output for the function in the Jaccard\_similarity module, I expect it to give me 2 vertices and the Jaccard similarity function for it. I also designed it so it would loop over the hash map to print all the values associated with the specifiers. Because the output is too long, I am showing a snippet of what it looks like in the terminal for one line of output:

Jaccard Similarity between 210 and 127: 0.0769

Here it says that the Jaccard similarity for Node 210 and 127 is 0.0769. This means that the member associated with these nodes have a very low similarity score. That means the sets of neighboring nodes aren’t very connected with each other, hence yielding the low similarity score. This can be useful as it shows how these 2 members aren’t utilizing their connections to the fullest, and especially for a research institute where collaboration is necessary, this can be used as a tool to promote more collaboration between colleagues of colleagues.

The output for the Jaccard\_stats module has 2 outputs I would like to delve into. The first one is giving the output of the mean, max, and the vertices associated with the max value. I expect the mean to read all the similarity scored computed by the jaccard similarity module, and same with the max. Also, I would cross reference the max vertices with the holistic output to see if it is true. The result I got for shortest path of 2 is:

Mean Jaccard Similarity: 0.0393

Max Jaccard Similarity: 1.0000

Vertices achieving the maximum Jaccard Similarity:

Vertex 1: 839, Vertex 2: 824

The mean Jaccard similarity being so low is a very good indicating data value for the research institute in my opinion. It shows how there could be improvement in collaboration between members, especially when it is a colleague of a colleague, as they can give insight into each other’s research.

That being said the Max Jaccard similarity score being 1.00 does give hope for the research institute. This means that members associated with nodes 839 and 824 have identical sets of neighboring vertices. This could be explained with them being in the same small department or having a very good collaboration network.

The 2nd output for the Jaccard\_stats module is giving what percentage of all the similarity scores are above thresholds from 0.1 to 1.0 in increments of 0.1. I expext a percentage with each increment that goes down as increments go up based on how low the mean is. The output is as follows:

Percentage of Vertices with Jaccard Similarity Scores:

Threshold: 0.00, Percentage: 92.05482400563793%

Threshold: 0.10, Percentage: 6.801273394457192%

Threshold: 0.20, Percentage: 0.9303972587997181%

Threshold: 0.30, Percentage: 0.1725400886654701%

Threshold: 0.40, Percentage: 0.024648584095067157%

Threshold: 0.50, Percentage: 0.014233689407010613%

Threshold: 0.60, Percentage: 0%

Threshold: 0.70, Percentage: 0.0013886526250742061%

Threshold: 0.80, Percentage: 0%

Threshold: 0.90, Percentage: 0.00034716315626855153%

Threshold: 1.00, Percentage: 0.00034716315626855153%

As you can see from this, majority of the similarity scores is below the 0.1 similarity score threshold, which matches with the mean. This data value is useful to use as a metric for the research institute on giving researchers a linear path to improve collaboration between one another. Promoting the goal of trying to get a majority percentage for the 0.8 threshold for example. Moreover, the reason thresholds 0.6 and 0.8 show 0% is that there are no two nodes with similarity scores close to 0.6 and 0.8.

Extra Analysis:

To expand on the problem from earlier I computed a little more data analysis. I wanted to see what would happen if I changed the shortest path metric between 2 vertices from 2. This prompted me to create the following graph of the result I found intriguing.

The graph above examines how changing the shortest path between 2 vertices, affects the mean Jaccard similarities for all vertices. The exponential decrease sort of makes sense when you think about it, because increasing the connections between to members of the institute will make it less likely that they would collaborate with the contact of that contact for the comparing contact. Moreover after 5 connections between 2 members, the jaccard similarity scores would be 0 for all vertices. This could give the institute insight on upping collaborators where the cutoff should be more than 5 people. By increasing the mean similarity score for a shortest path of 2, could increase the cutoff of 5, as more collaboration would be happening overall.

Testing

The tests were computed by hand and a scientific calculator and the answers were put into the assert\_eq!, to ensure that the test has passed.

Reading the Graph: The test\_read\_graph function reads a test dataset into a graph, asserting that it contains 7 nodes. This ensures that the graph reader operates correctly.

Jaccard Similarity: The test\_jaccard\_similarity function calculates the jaccard similarity between 2 nodes with a shortest path of 2 in the test dataset, it is to make sure that the similarity between nodes 1 and 7 is 0.25. This verifies the accuracy of the similarity computation by comparing it to a known value.

Mean and Max Similarity: The test\_jaccard\_stats\_similarity function computes the mean and maximum similarity scores for the test dataset, making sure the values are 0.2044 and 1.00 respectively. This verifies that the statistical metrics are computed accurately and align with manually calculated values.

BFS Shortest Path: In the test\_bfs\_shortest\_path function, I validate the accuracy of the BFS algorithm by computing the shortest path from a source vertex to a destination vertex in our test dataset. I assert that the computed distance is 1, ensuring the algorithm's precision in determining the shortest route.

Similarity Percentages: Within the test\_compute\_similarity\_percentage function, I gauge the distribution of similarity scores across pairs by calculating the percentage of pairs with scores exceeding different thresholds. This comprehensive assessment spans thresholds ranging from 0.0 to 1.0, ensuring thorough coverage. Additionally, I verify specific values at thresholds 0.0 and 0.1, confirming the consistency of our percentage calculations with the anticipated outcomes.

Of course, all the tests have passed ensuring validity to all the functions. The following screenshot provides proof of the successful test runs:

A black background with green text

Description automatically generated

The Conclusion

In our DS210 Final Project, I utilized network analysis techniques to explore relationships within a European research institution's email exchange network. By combining the Breadth-first search (BFS) algorithm with Jaccard similarity scores, I aimed to uncover the depth of connections among institution members.

Through computation and analysis, I discovered insights into collaboration patterns within the institution. The Jaccard similarity scores provided a metric to quantify similarity between sets of neighboring vertices, shedding light on the interconnectedness of individuals.

Our computational framework comprised three main modules: graph creation, Jaccard similarity computation, and statistical analysis. By analyzing mean and maximum similarity scores, as well as distribution across thresholds, I gained actionable insights into collaboration dynamics.

Overall, our project highlights the potential of network analysis in understanding and fostering collaboration within research institutions. Through computation and analysis, I provided valuable insights aimed at enhancing connectivity and innovation within the academic community.