DS210 Final Project Writeup

Report:

This project computes the average vertex degree distributions on an ego-net (social network) graph dataset of Twitch streamers. Specifically, I am finding the average vertex degree of a streamer who plays multiple games versus a streamer who only plays one game. The vertex degree refers to the number of other vertices one is connected to.

The code is split into four modules:

- Files module (I/O processing functions to read the dataset files):
    - pub fn read_edges_file
        - takes in the user's path to the twitch_edges.json file in the twitch_egos dataset
        - reads and iterates through the file contents to store each egonet into an adjacency HashMap
        - returns the data in a Graph vector named edges_file, created with fn create_graph.
    - pub fn read_labels_file
        - takes in the user's path to the twitch_target.csv file in the twitch_egos dataset
        - reads the file contents and iterates through each node to load its id and target label into a new HashMap
        - returns the data into a map named targets_file.
- Tests module:
    - fn test_calculate_average_vertex_degree()
        - this function takes in no parameters, but initializes an example targets HashMap with sample nodes and example edges graph with a sample adjacency list.
        - Uses assert_eq to make sure the calculate_average_vertex_degree function calculates the ego net's averages correctly (should be 2 for both categories)
- Visuals module:
    - Pub fn draw_histogram:
        - Takes in a sorted HashMap containing each node in the targets_file and its calculated respective vertex degree, or the degree distribution of all the nodes in the graph
        - Finds the max vertex degree among the nodes and normalizes the rest of the nodes to the value, and iterates over the data to create a bar for each vertex degree count

        - Prints the final histogram, where the number represents the vertex degree and the line represents how many times a node in the graph had that same number of degrees. This function was created with the help of chat gpt for the purpose of adding a secondary feature to the project.
- Main module:
    - Impl graph
        - fn new()
            - takes in the n number of nodes in the ego net's adjacency list computed later in fn read_edges_file
            - initializes and returns a new graph, which consists of an empty adjacency list of length n
        - fn create_graph()
            - takes in n and a vector of edges from the current ego_net,
            - creates a new graph and pushes the adjacent edges from the list into
            - returns the new graph
        - fn get_degree()
            - takes in the current node being iterated over in fn calculate_average_vertex_degree
            - computes and returns the count of its neighbors found in the adjacency list using len. This was written with the help of chat gpt to fix an error I was encountering while making my test function.
    - pub fn calculate_average_vertex_degree
        - takes in the computed ego-net averages from the edges_file and target_file
        - iterates through each node in the target file, and depending on its label (whether the user plays multiple games or not), the node's degrees are added to the respective counter and the node is counted.
        - returns 2 floats of the averaged degree of those who play multiple games, and those who play one game

Results:

The histogram is heavily right-skewed, meaning that twitch users usually have fewer friend connections instead of many friend connections. Across all ego-nets in the dataset, the mode (value that appears most frequently) is 33 friend connections.

The computed average degrees show that Twitch users who play only one game (317) on average have more friend connections than users who play multiple games (257). This is in

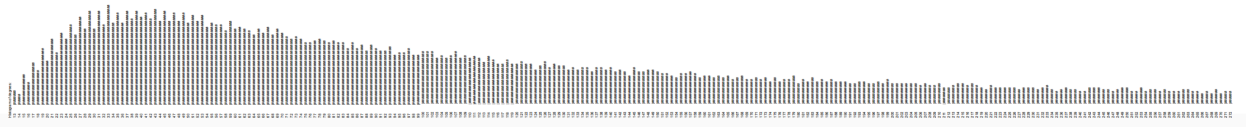
line with results of the initial binary classification task done on this data (https://arxiv.org/pdf/2003.04819.pdf).

The GitHub repository does not contain the original SNAP dataset files because they are too large. It can be downloaded from https://snap.stanford.edu/data/twitch_ego_nets.html to run the program.

Program Output:

```
Average degree for multiple games: 256.7186228961386
Average degree for single game: 316.8169259362241

Histogram of degrees:
```



(terminal output rotated + condensed for readability)

Resources used:

Lectures Jupyter Notebooks
https://doc.rust-lang.org/book/
https://www.altcademy.com/blog/compute-the-degree-distribution-of-a-graph/
https://arxiv.org/pdf/2003.04819.pdf
https://stackoverflow.blog/2022/05/26/the-complete-beginners-guide-to-graph-theory/
https://whoisryosuke.com/blog/2022/parsing-json-with-rust
https://chat.openai.com/share/9daca545-d7f7-4d6c-9b9a-80d6b100dbc3

https://stackoverflow.com/questions/50662744/how-to-parse-int-value-from-string-in-rust
https://bookdown.org/omarlizardo/_main/2-7-average-degree.html