Katherine Wu
DS210 Project Writeup
Collaborators: None

A. Project Overview

- Goal: Friends-of-friends analysis. How often are a node's connections connected to each other? Additionally, which two vertices who are friends have the most similar or most dissimilar sets of connections?
- Dataset: The dataset I used (too large for GitHub) came from the Stanford SNAP website, and includes data from the Amazon website and product listings under the "Customers who purchased this item also bought" section under every product. Thus, it represents a directed graph with products as nodes and edges represent the item appearing under the other listing.
    - If the hyperlink doesn't work: https://snap.stanford.edu/data/amazon0601.html

B. Data Processing

- I loaded my txt file into Rust by using it as the path in my main function. I had a function to create edges out of the graph using bufread for files.

C. Code Structure

- Module 1: graphs.rs
    - This module contains the Graph struct and additional methods to create directed edges given an edges list.
- Module 2: similarities.rs
    - This module contains functions for different measures of similarity and functions that use the similarity measurements to analyze the graphs.
    - Cluster coefficient tells how likely it is that friends of friends are friends with each other, using this formula:

$$C(v) = \frac{2 \times \text{number of connections between neighbors of } v}{\text{degree}(v) \times (\text{degree}(v) - 1)}$$

    - Jaccard similarity calculates the overlap between friends of nodes, using this formula:

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- In the main code, the graphs.rs module is used to create the directed graph using the Amazon data. The functions in the similarities.rs module are then called onto the Graph to produce the final results.

D. Tests

Cargo test output:

```
warning: `project` (bin "project" test) generated 7 warnings (run `cargo fix --bin "project" --tests
    Finished `test` profile [unoptimized + debuginfo] target(s) in 1.08s
     Running unittests src/main.rs (target/debug/deps/project-36aaaf3f197e8e37)

running 2 tests
test similarities::test::test_clustering ... ok
test similarities::test::test_jaccard ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

- test_clustering checks the clustering coefficient, which calculates how likely that friends of friends are also friends. In the test, I made a sample graph, ran my clustering_coefficient function on the graph and node 0, and got the correct clustering coefficient, ⅔, as 2/3 of node 0's friends were also linked - there are edges between (1, 2) and (2, 3), but not (1, 3). This is important because clustering_coefficient is used in full_similarity and no_similarity to get the nodes where each friend is friends with each other and where none are friends with each other, respectively.
- test_jaccard checks the calculation for jaccard similarity, which measures the overlap of friends for two nodes. I created a sample graph for this test, and then used my jaccard_similarity function to check the overlap between friends for nodes 0 in 1 in my sample graph. This is important because jaccard_similarity is used in greatest_jaccard and least_jaccard to get the friends that have the greatest overlap and the friends that have the least overlap in friends, respectively.

E. Results

```
katherinewu@crc-dot1x-nat-10-239-15-40 project % cargo run --release
   Compiling project v0.1.0 (/Users/katherinewu/Downloads/rust/project)
    Finished `release` profile [optimized] target(s) in 1.40s
     Running `target/release/project`
greatest jaccard similarity: (34158, 25793)
similarity: 0.9
smallest jaccard similarity: (1, 4)
similarity: 0.0
number of nodes that have a clustering coefficient of 1.0 - all their friends are friends: 22766
number of nodes that have a clustering coefficient of 0.0 - none of their friends are friends: 24989
```

-
  - Nodes 34158 and 25793 were friends that had the greatest Jaccard similarity out of all the pairs of linked nodes in the graph. Their similarity was 0.9, meaning 90% of their linked nodes overlapped.
  - Nodes 1 and 4 were friends which had the smallest overlap between friends, with a Jaccard similarity of 0.0, meaning that none of their other friends overlapped. It is possible that there are other nodes that also had this similarity too, but my function just returns the first instance of a similarity of 0.0, which was found quite early on.
  - I printed out the number of nodes with a clustering coefficient of 1.0, which meant that all their friends were friends, and there were 22,766 nodes for which this applied. This shows that for many nodes, their friends shared a similar network as them, which makes sense since Amazon probably promotes items that are similar to each other under each other.

- I also printed out the number of nodes with a clustering coefficient of 0.0, which meant that none of their friends were friends, and there were 24,989 of them.
    - Another interesting takeaway is that 403,394 - 22,766 - 24,989 = 355,639, which means that a vast majority of the nodes had a clustering coefficient between 0 and 1, which definitely makes sense and shows that only some of a node's friends are linked to each other.

F. Usage Instructions

- Build/run code using cargo run → make sure the 'amazon0601.txt' file is also located in the project folder
- With cargo run --release, the runtime is quick and should take no longer than a few minutes.

G. AI-Assistance Disclosure and Other Citations

- Asked ChatGPT for a general overview for how I could find similarity between friends-of-friends and for overlap between friends. It returned the formulas for clustering coefficient and jaccard similarity, which I implemented functions for in my similarities.rs module.

📌 1. Friends of Friends Being Friends → *Clustering Coefficient*

Concept:

Measures how likely it is that the friends of a node are also friends with each other.

For a node `v`, the *local clustering coefficient* is:

$$C(v) = \frac{2 \times \text{number of connections between neighbors of } v}{\text{degree}(v) \times (\text{degree}(v) - 1)}$$

- If all of a node's friends are also friends with each other, $C(v) = 1$.
- If none of them are, $C(v) = 0$.

📌 2. Similarity Between Friends' Connections → *Jaccard / Cosine Similarity*

A. Jaccard Similarity:

$$J(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

Where $N(u)$ is the set of neighbors of node `u`.

- I also asked it for elaboration on some parts - like how I could find the number of connections between neighbors and to elaborate on the purpose of Jaccard similarity.

for the local clustering coefficient, how can i find the number of connections between neighbors of v?

Great question! To compute the **local clustering coefficient** for a node `v`, we need to find out how many edges exist **between its neighbors** — that is, how interconnected the neighbors are **excluding** `v` **itself**.

📌 Here's how to do it step-by-step:

Suppose:

- `v` has neighbors `N(v) = {n1, n2, ..., nk}`
- We want to count how many of the possible $\binom{k}{2} = \frac{k(k-1)}{2}$ edges **actually exist** among the neighbors.

Could you explain jaccard similarity to me?

Absolutely! The **Jaccard similarity** is a simple and intuitive way to measure how similar two sets are. In the context of graphs, it's often used to compare the **neighbor sets** of two nodes (i.e. their friends).

🧠 Definition of Jaccard Similarity

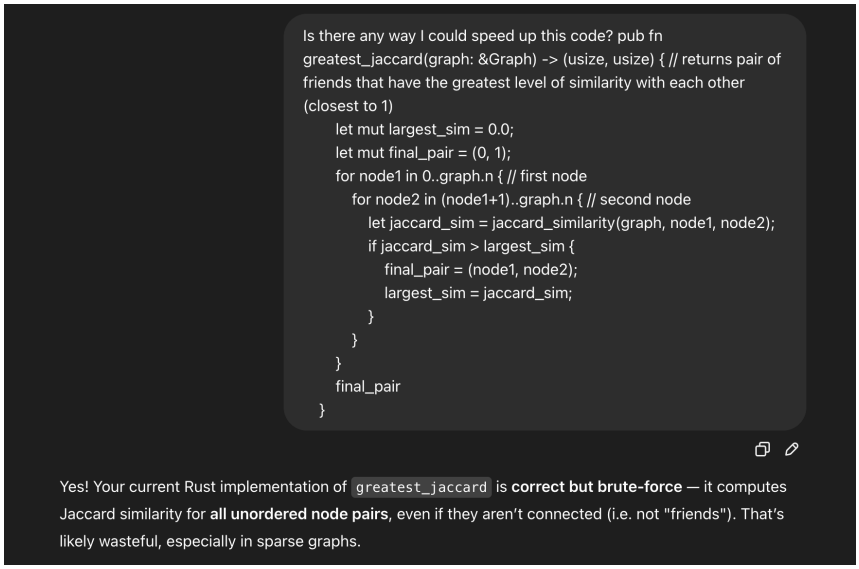
Given two sets A and B:

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- $A \cap B$ is the number of **elements they share** (intersection).
- $A \cup B$ is the **total number of unique elements** in either set (union).

    - These explanations helped me understand how each formula works to calculate the similarity. ChatGPT also gave me sample Python implementations, but I didn't really look at them because I wanted to implement the formulas in Rust myself.
- I also asked ChatGPT for help on how to speed up my runtime, because I initially had the greatest_jaccard function to be a nested for loop with 0..graph.n and (i+1)..graph.n, iterating between every possible combination of two nodes. Since the complexity of that

code was O(n^2), it was taking a long time to run, even using --release, as there are 400,000+ nodes in my data.



- ChatGPT's response helped me realize that I didn't need to look through every combination, because my stated objective was to find a pair of nodes that are already friends and had the most overlap. Thus, I could just iterate through the node's friends in the nested loop, reducing the complexity and the runtime.

Full chat log: https://chatgpt.com/share/68152840-f454-8011-9d74-5d93c07e2698 (While Chat provided some code examples, I mainly used it to ask about the overall ideas, and wrote my own code once I figured out what the issue was).