Anthony Ciccone

CS-581WS

Assignment03

10-28-20

Report for Assignment 05

**Purpose**

The purpose of this program is to find triads of relations in a dataset from epinions.com. It takes a massive dataset on reviewers reviewing other reviewers and can make positive / negative relations between all of them. Within this data there are triads of reviews that shows a relationship between the three reviewers.

**Input**

This program awaits input from the user of a filename with .csv added. It then takes the rest of the necessary information from opening that specific file.

**Output**

The output here is a bit tricky. First it gets a lot of basic information. edgeCount is the total edges, loopCount are the total self-loops, totEdges is edges – selfloops because selfloops can’t be counted in triads, pEdge is the total positive edges, nEdge is the total negative edges, triads is the total triads in the data. Besides that, it goes deeper into the probabilities of getting combinations of edges in triads. Like TTT, TTD, TDD, and DDD are possibilities of positive and negative edges in a found triad. With those probabilities you can compute expected values and compared those with the data that was found from the csv file.

**What the program does**

The computation here is simple for most of the output, it is just simply counting and adding the edges to the network graph. After getting the basic data it begins running through the edges in the graph. From there is checks to see if at every edge, there are two nodes that connect to each other and back to the first one. This signifies a triad and is accidentally found by each node in the triad, making it run that section of the code 3x as much, as necessary. When it finds the triad, it determines what relationship type it is out of TTT, TTD, TDD, and DDD. After all that, it computes the probabilities and expected values of the data and compares it to the real data found in the csv file, as explained above.

**Additional Information**

I could have improved this by not haphazardly dividing all the relevant variables by 3 by keeping track of the triads I have seen so far. Maybe with some sort of HashMap or list that holds the seen ones already. This is also probably much less efficient because it computes 3 times as many triads when it finds the triads from each node in the graph. The output is as close as I could reasonably do to your template.