Megha Mansuria
Prof. Cheryl Dugas
CS 581 WS
28 October 2020

Assignment 5: Graphs

PURPOSE

The program graphs.py considers positive and negative relationships in a network with triads of reviewers, reviewees, and weights of the trust/distrust. The dataset of different epinions csv files calculates specific data regarding the graph that is created. At the completion of the program, there is printed output in the console: information and statistics regarding the graph and the distributions of the different triad types.

INPUT

After running the program in the terminal, the user is first prompted for a csv file.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

meghamansuria@Meghas-MacBook-Pro HW5 % python3 graphs.py
Enter a file: epinions96.csv
```

OUTPUT

The output is printed in the console with the required statistics regarding number of edges, self-loops, edges used to identify triads, trusts, distrusts, and triangles. The output also includes the respective probabilities and expected vs. actual distributions of triad types.

```
meghamansuria@Meghas-MacBook-Pro HW5 % python3 graphs.py
Enter a file: epinions96.csv

************************ Statistics ************************
Edges in network: 96
Self-loops: 0
Edges used - TotEdges: 96
Trust edges:    68        probability p: 0.71
Distrust edges: 28        probability 1-p: 0.29
Triangles: 27

Expected Distribution    Actual Distribution
Type percent number      Type percent number
TTT   35.5     9.6       TTT     74      20
TTD   43.9    11.9       TTD     11       3
TDD   18.1     4.9       TDD     11       3
DDD    2.5     0.7       DDD      4       1
Total  100      27       Total  100      27

meghamansuria@Meghas-MacBook-Pro HW5 %
```

WHAT THE PROGRAM DOES

To begin, the program starts by asking for the user to give a single input: a csv file. As the program processes the given file, the count of edges, self-loops, positive edges (trusts) and negative edges (distrust) increment. These edges between the reviewer and reviewee are added to a graph with the respective weight (the weight is referring to trust/distrust). Next, the program finds all the triads in the csv file and determines the number of triangles. These aspects of the data are printed. Then, the expected distributions of the triad types (TTT, TTD, TDD, DDD) are calculated by using the positive probability and the negative probability to determine the percentages and numbers of the triads of each type to be expected. Similarly, the actual data of the triads from the csv file are determined by counting the total weights of the triangles and assigning the types. The count of actual numbers increments accordingly and the percentages are also found as well by comparing (actual numbers)/(number of triangles). Finally, the expected and actual distributions of the triad types are printed out.

ADDITIONAL INFORMATION

Looking at the printed output, the actual distribution of triad types differs from the expected distribution based on a random assignment. They differ because the breakdowns of the types are not the same values. This is because the expected distribution is calculating expected percentages and numbers based on the probabilities of the edge being positive and negative. It calculates the expected values based on all the combinations of getting TTT, TTD, TDD, and DDD. However, the actual distributions can differ based on how the triangles actually formed and what weights the edges hold. The graph of all the actual triads can end up being different combinations of the triad types than what is expected by the probabilities.

While working on this assignment, I had difficulty understanding how the graph was being made for the triads regarding the nodes (reviewers/reviewees) with their weighted edges. I struggled with going about the code for the actual distribution. It took me some time to figure out that I should iterate through my all_triads list and grab each value of the weights and add them up. The weight attribute came in handy for the triad types, but it took so long for me to figure that out! Then, I was able to compare the sums to what each type was (a total of 3 means TTT, 1 means TTD, -1 means TDD, and -3 means DDD). This was the hardest part for me to get to but once I had a breakthrough for this method, it made the rest of the calculations easier. Overall, I still very much enjoyed using the NetworkX package and it had some useful methods like add_edge (with a weight attribute!) and enumerate_all_cliques (which I used in my code to get all_triads).

Note: I realized some of the values differ if you were to calculate the percentages by using the probabilities (e.g. for the expected distribution) and that is due to the fact that I round all the variables - the probabilities, percentages, numbers, etc. - in the printing but my calculations are completed without the rounded numbers.