

Analysis of Facebook Pages: TV Shows Dataset

A Project Report

Presented to Dr. Potika
San José State University

In Partial Fulfillment
Of the Requirements of the Class
Spring 2024: CS 176

By Charlene Khun and Tiantong Li

May 12, 2024

Table of Contents

I .Introduction.....1

II . Algorithms.....2

III. Experiments and Results.....6

IV. Conclusion.....9

V. References.....10

I . Introduction

In our project, we analyze the relationships of TV shows collected from Facebook pages. We got our dataset from networkrepository.com under the social network category. The data was collected in November 2017. There are a total of 3892 nodes and 17262 edges. Each node represents an individual Facebook TV show page while the undirected edges represent the mutual likes among the TV show Facebook pages.

17245	3748,3837
17246	3868,2298
17247	3762,3827
17248	3767,3816
17249	3775,3823
17250	3777,3777
17251	3783,3788
17252	3786,3867
17253	891,3863
17254	3798,3806
17255	3741,3808
17256	3813,3854
17257	486,3867
17258	3826,3844
17259	3830,3843
17260	1240,1240
17261	3876,3885
17262	3879,3886
17263	

Fig1. A Sample of the Dataset

The problem we want to solve is that social media platforms like Facebook have become significant sources of data for understanding user preferences and behavior. Facebook pages dedicated to TV shows accumulate vast amounts of data regarding user interactions. In our dataset, this would be mutual likes. First, analyzing these interactions can provide insights into audience preferences and interests. For example, we would like to know which TV shows are more popular. Secondly, we also want to study the connections to find the communities in the dataset.

II . Algorithms

For analysis, we used built-in algorithms from NetworkX. For community detection specifically, we chose to use the Girvan Newman and Louvain algorithms. This section will introduce the algorithms we used and explain why we chose them.

1. Parsing the dataset

The dataset we used represents the edges as two columns. To parse our dataset, we used the `nx.read_edgelist()` which is a built-in algorithm in NetworkX. The algorithm can take in several arguments, however, we just used the file name, delimiter, and `create_using` as our arguments. `Create_using` is the graph type we want to create. We used this function because we want NetworkX to properly read our dataset and get it prepared for visualization.

2. Basic graph information function

We created a function called `get_graph_info(G)`. This function takes in a graph as an argument. Inside this function, we use the built-in functions, `graph.number_of_nodes()`, `graph.number_of_edges`, `list(graph.nodes)`, and `list(graph.edges)`. The first two functions display the total number of nodes and edges in the graph. The last two functions display the nodes and edges as a list. We used this function to display the basic information of our graph.

3. Connected components

To find the connected components in the graph, we used the algorithm `number_connected_components(G)`, a built-in algorithm in NetworkX. The algorithm takes in an undirected graph `G` and returns the number of connected components in graph `G`. We used this algorithm because we wanted to find how

many connected components there are in our graph. A connected component is a group of nodes in a graph where each node is reachable from every other node in that group, but no node in that group is connected to any node outside of the group. This can help reveal the structure and connectivity of a graph.

4. Diameter

The diameter function, `nx.diameter(G)`, is a built-in algorithm in NetworkX. It takes in a graph as an argument and a weight if there's any. The function returns the diameter of the graph. The diameter of a graph is a measure of the longest shortest path between any pair of vertices in that graph. In other words, it represents the maximum distance between any two nodes in the graph. We used this function to find out how far apart the nodes in our network are from each other and we wanted to know how efficient the flow of the network is.

5. Clustering coefficient

We used `nx.clustering.(G)`, a built-in algorithm in NetworkX. It takes in a graph and weight if any. The function returns a dictionary of the clustering coefficients and their associated nodes. To find the top 5 nodes, we sorted nodes and kept only the top 5. A clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. We want to analyze the structure of our network. Higher clustering coefficients often indicate a more modular or clustered network structure, while lower clustering coefficients suggest a more random or decentralized structure.

6. Betweenness Centrality

We used `nx.betweenness centrality(G)`, a built-in algorithm in NetworkX. It takes

in a graph and weights if any. The function returns a dictionary of the nodes (keys) and their corresponding betweenness centralities (values). Similar to the clustering coefficient algorithm, we sorted the nodes and only kept the top 5. We want to analyze betweenness centrality because it can help identify critical nodes that are strategically important for maintaining network connectivity and communication.

7. Community Detection: Girvan Newman Algorithm

We used 3 helper functions `get_community`, `set_community`, and `get_color`. The first function takes in a graph and their communities as an argument. It is used to divide the network into communities. The second function takes in a graph and is used to set the edges. The last function takes in color values as an argument. It is used to color-code the graph. We then used the built-in function `nx_comm.girvan_newman(G)` which takes in a graph. Next, we used the `itertools`'s slicing function and put a very small parameter because the algorithm was taking too long on NetworkX. We also used `len(communities)` to find the total number of communities. Lastly, we used `nx_comm.modularity(G)` to find the modularity score. The Girvan-Newman algorithm is a top-down algorithm that effectively partitions the network into communities by iteratively removing edges that act as bridges between these communities. By identifying edges with high betweenness centrality and removing them, the algorithm uncovers the underlying community structure of the network. Since NetworkX only finds a small subset of communities, we wanted to experiment with Gephi. To do that, we downloaded the Girvan Newman plugin. Then we ran the Girvan Newman Clustering method.

8. Community Detection: Louvain Algorithm

Similar to the Girvan Newman algorithm, we used 3 helper functions `get_community`, `set_community`, and `get_color`. Each function is explained in the Girvan Newman Algorithm section. We then used `nx_comm.louvain_communities(G, seed=123)` which is a built-in function in NetworkX. The seed function initializes the random number generator with a specified value. This function makes sure that the subsequent random numbers generated by the Louvain algorithm are reproducible and consistent across multiple runs of the program. Next, we used the `len(partition)` to find the total number of communities. Lastly, we used `nx_comm.modularity(G)` to find the modularity score. The Louvain algorithm is a bottom-up algorithm that iteratively merges communities to maximize the modularity of the network. It can produce high-quality community partitions that reflect the underlying structure of the network.

9. Visualization

For our visualization, we decided to use both NetworkX and Gephi. Since we are most familiar with the Spring Layout, we decided to use that for two of our visualizations. We also decided to experiment with different layouts on Gephi because NetworkX is not as good at handling large datasets. Some layouts we decided to use from Gephi include the Circular Layout, Yifan Hu layout and the Yifan Hu Proportional Layout.

III. Experiments and Results

Tables of our results are shown below.

Experiments	Results
Number of Nodes	3892
Number of Edges	17262
Number of connected components	1
Diameter	20

Top 5 Nodes	Clustering Coefficient
Living Biblically (Node 2972)	1.0
BOOM (Node 3283)	1.0
Jorge Ramos y su Banda (Node 4)	1.0
Return To Amish (Node 2713)	1.0
The Voice Kids (Node 7)	1.0

Top 5 Nodes	Betweenness Centrality
Queen of the South (Node 3254)	0.10544488181477074
Home & Family (Node 2008)	0.09352541687013526
The Tonight Show Starring Jimmy Fallon (Node 819)	0.0804900367587108
The Voice Global (Node 2170)	0.07471499425323284
The Voice (Node 2751)	0.07465790776474893

Girvan Newman Algorithm	Results
Number of Communities (Network X)	4
Modularity Score (Network X)	0.03602357841173616
Number of Communities (Gephi)	695
Maximum Modularity Score (Gephi)	0.22318083

Louvain Algorithm	Results
Number of Communities	46
Modularity Score	0.8721668853348115

3.1 Analysis of Results

For our total number of components, we got 1. This suggests that the graph is very connected and that we have a single giant component. For our diameter, we got 20, which means that the nodes are at most 20 hops away from each other in terms of the number of edges. It also implies that communication or travel between certain nodes may be more time-consuming.

For the top 5 nodes clustering coefficients, we got a result of 1 for all five. A high clustering coefficient suggests that these 5 nodes are very connected and all their friends know each other. This indicates a strong tendency for the nodes associated with these TV shows to form tightly interconnected clusters within the graph.

Our top 5 nodes betweenness centrality results are listed on page 6 of the report. Nodes with high betweenness centrality suggest that these nodes play critical roles in connecting different parts of the network.

The Girvan Newman Algorithm identifies communities by removing edges with high betweenness centrality, which are typically edges that serve as bridges between communities. To run this algorithm, we used both NetworkX and Gephi. In NetworkX, we ran 2 steps and got 4 communities and a modularity score of 0.036. One community appears notably larger than the others, as shown in Fig.3 on page 9. In Gephi, we got 695 communities and a modularity score of 0.22. Even though we get more communities, the modularity score is still relatively low. The largest community in Gephi takes up to 65% of the nodes in the graph.

The Louvain algorithm optimizes modularity, which measures the density of edges within communities compared to between communities. In the Louvain algorithm, we got 46 communities and a high modularity score. Obtaining these results indicates a well-structured network with clear and densely connected communities. It also indicates similar levels of connectivity between communities so that the algorithm partitions the graph into multiple communities to maximize modularity.

3.2 Visualizations

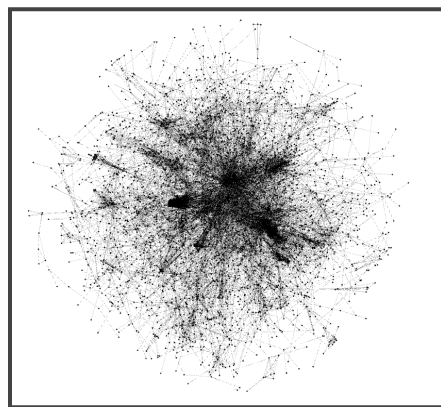


Fig2. Gephi Yifan Hu Proportional Visualization of the Original Dataset

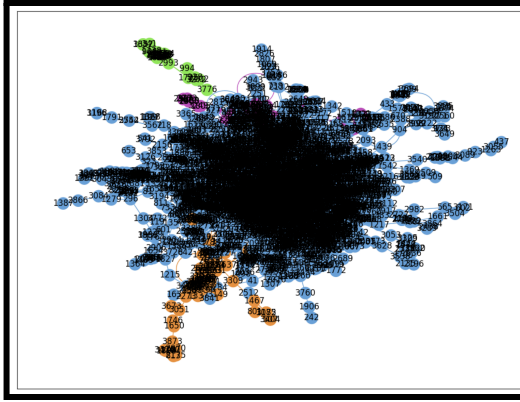


Fig3. Girvan Newman algorithm in NetworkX

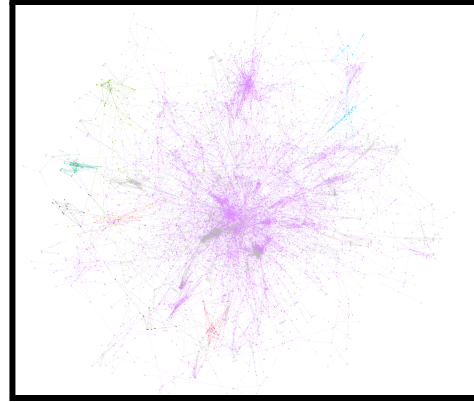


Fig4. Girvan Newman algorithm using Gephi

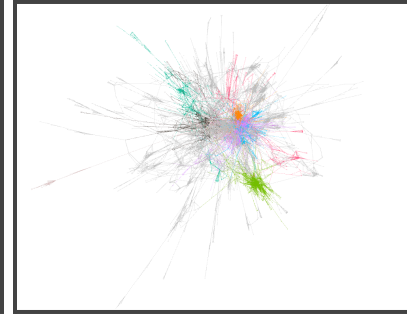
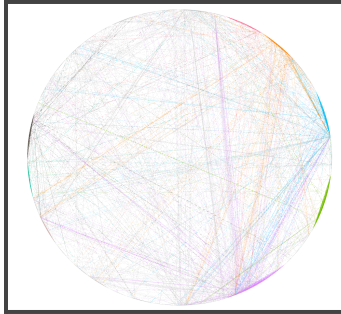
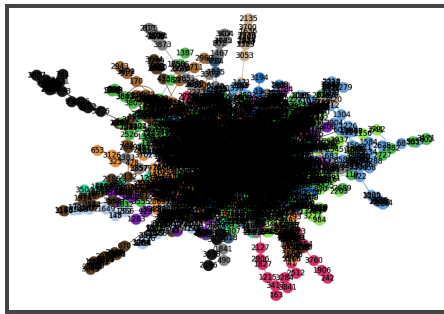


Fig5. Louvain Algorithm in NetworkX Fig6. Louvain in Gephi Circular layout Fig7. Louvain in Gephi Yifan Hu Layout


IV. Conclusion

We have found several interesting facts from the dataset. First, a high clustering coefficient doesn't mean the node is popular in the graph because TV shows with high clustering coefficients are unpopular. Rather, a high clustering coefficient simply means that the nodes are very connected and all their friends know each other. Second, high betweenness centrality means you're important in the graph. These nodes play critical roles in connecting different parts of the network. TV shows with a high betweenness centrality are considered to be well-known. For example, The Voice and The Tonight Show Starring Jimmy Fallon. For community detection, Girvan Newman only finds 4

communities in NetworkX and 695 communities in Gephi. Both results have one big community that takes more than half of the nodes. Louvain finds 46 communities and the sizes of the communities are more even. When we looked into the big difference between the results, we found several possible reasons which may help explain this difference. The first reason is that the graph may have distinct characteristics that favor one algorithm over the other. Another reason is that the Louvain algorithm might be more sensitive to fine-grained community structure, detecting smaller, more internally cohesive groups. Girvan-Newman, on the other hand, may identify larger, more interconnected communities. However, both results have shown that the graph has a homogeneous structure and the graph represents a single, large community without clear subdivisions. These results can also be deduced from the visualization of the dataset.

Throughout the project, we have learned how to analyze a dataset to find what we're interested in using graph theories and graph-related tools, such as NetworkX and Gephi. We have also found beauty in community detection which is powerful in revealing the hidden structure of a graph.

V . References

- [1] Tools: Python, Google Colab, Gephi
- [2] NetworkX <https://networkx.org/>
- [3] Network Repository <https://networkrepository.com/fb-pages-tvshow.php>
- [4] Link to the Code  CS176Project.ipynb