Analyzing Communities and Influence in the YouTube Social Network

Aaliyah Rachel Aman

December 8, 2024

DS 210 Final Project

Context

YouTube is currently one of the largest and most influential social media platforms in the world, with over 2 billion monthly active users as of 2023. Beyond entertainment, YouTube serves as a critical hub for information dissemination, cultural influence, and community building. Understanding how communities form and how influence spreads within YouTube's social network would provide valuable insights to mapping community structures, identifying key influencers, and analyzing how information, trends, or even misinformation propagate.

Problem Statement

This project aims to analyze the YouTube social network using graph algorithms to:

- Detect major community clusters in the network
- Identify influential nodes within and across these communities
- Investigate ways in which influence spreads
- Find bridge users connecting different communities

Goal: To make use of graph theory and algorithms to reveal insights into the structure of YouTube's social network and demonstrate patterns through these algorithms.

Dataset

The dataset used in this project is the "Youtube social network and ground-truth communities" dataset from the Stanford Network Analysis Program (SNAP)

Link: https://snap.stanford.edu/data/com-Youtube.html

The dataset consists of:

- Nodes representing YouTube users
- Edges representing connections between users
- Community File depicting ground-truth communities of users

The specific files used in this project are:

- com-youtube.ungraph.txt which represents the undirected user-to-user graph, contains 1134890 nodes and 2987624 edges
- com-youtube.all.cmty.txt which represents the ground-truth communities, where communities with fewer than 3 nodes were excluded

The significance of the dataset provides a large-scale social network that is suitable for the analysis and allows the exploration of community structures.

Libraries Used in the Project

1. indicatif - provides progress bars for long-running computations
2. rand - supports random sampling for optimization
3. Standard Rust collections such as HashMap, HashSet, and VecDeque for graph algorithms

How to Run the Project

1. Prerequisites
    - Ensure that Rust is installed
      To install Rust, run the following line:
      curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
2. Clone the Repository
    - The dataset and project code are included in this repository, clone the repository using:
      git clone https://github.com/aaliyahamann/youtube-community-analysis.git
      cd youtube-community-analysis

3. Build the Project
    - Ensure that all dependencies are installed and the code is compiled
    - cargo build
4. Run the Code
    - Run the program to perform the analysis
    - cargo run
5. Run the Tests
    - Validate that each function of the program is working as intended
    - cargo test

Sample Expected Output

Graph loaded successfully with 1134890 nodes.
The graph has 2987624 edges.
Degree distribution calculated:
Minimum degree: 1
Maximum degree: 28754
Average degree: 5.27
The node with the highest degree is 1072 with 28754 connections.

Communities loaded successfully.

Top Communities (size >= 3):
Community 267: 1460 members
Community 821: 141 members
...

The largest community is Community 267 with 1460 members.

Running BFS from node 1 to calculate shortest paths:
Node 1 can reach 1134890 nodes. Average shortest path length: 4.16
Node 1 has 7151 neighbors at distance 2.

Jaccard similarity between nodes 1 and 2: 0.004 (Low)

Calculating Betweenness Centrality for top 50 highest degree nodes...
[--] 50/50
Top 5 Nodes by Betweenness Centrality:
Node 363: Betweenness Centrality = 731888.5341
Node 1072: Betweenness Centrality = 665940.0458
Node 1034018: Betweenness Centrality = 895436.4483
Node 663560: Betweenness Centrality = 854459.6228
Node 106: Betweenness Centrality = 81856.4557

Output from the Program

```
Graph loaded successfully with 1134890 nodes.
The graph has 2987624 edges.
Degree distribution calculated:
Minimum degree: 1
Maximum degree: 28754
Average degree: 5.27
The node with the highest degree is 1072 with 28754 connections.

Communities loaded successfully.

Top Communities (size >= 3):
Community 267: 1460 members
Community 3180: 597 members
Community 175: 583 members
Community 719: 567 members
Community 943: 531 members
Community 1167: 372 members
Community 976: 343 members
Community 4928: 336 members
Community 3707: 263 members
Community 3129: 244 members
Community 731: 241 members
Community 8341: 222 members
Community 1080: 221 members
Community 905: 188 members
Community 5083: 171 members
Community 5307: 164 members
Community 12245: 153 members
Community 150: 147 members
Community 862: 146 members
Community 821: 141 members

The largest community is Community 267 with 1460 members.

Running BFS from node 1 to calculate shortest paths:
Node 1 can reach 1134890 nodes.
Average shortest path length: 4.16

Node 1 has 7147 neighbors at distance 2.

Jaccard similarity between nodes 1 and 2: 0.004 (Low)

Calculating Betweenness Centrality for top 50 highest degree nodes...
                                                                       50/50
Top 5 Nodes by Betweenness Centrality:
Node 363: Betweenness Centrality = 7314888.5341
Node 1072: Betweenness Centrality = 6659409.0458
Node 1034018: Betweenness Centrality = 895436.4483
Node 663560: Betweenness Centrality = 854459.6228
Node 106: Betweenness Centralitv = 818556.4557
```

Explanation of Code and Output

The Youtube Social Network Analysis project focuses on detecting community structures and identifying influential nodes within a large-scale social network. In order to achieve this, I implemented several graph algorithms, each carefully chosen to provide meaningful insights into the dataset. The dataset used consists of 1134890 nodes and 2987624 edges which represent the YouTube users and the connections between users respectively. I stored these connections in an adjacency list which is a HashMap data structure that efficiently represents graphs of this scale. This adjacency list is represented as a HashMap<u32, Vec<u32>> where each key is a node and the value is a list of connected nodes. The graph was loaded using the load_graph function, which parses the node IDs in the com-youtube.graph.txt file, while skipping any possible comment lines, and updates the adjacency list. This step confirms the graph size and integrity, where the output “Graph loaded successfully with 1134890 nodes. The graph has 2987624 edges.” verifies that the file was read correctly.

To analyze the connectivity of the graph, I calculated the degree distribution using the calculate_degrees function, which computes the number of connections for each node. From this, I identified the node with the highest degree using the find_highest_degree_node function. Additionally, degree centrality, which is a measure of influence based on direct connections, revealed that Node 1072 was the most connected user with 28754 connections. This highlights Node 1072 as a key influencer in the network. The average degree of 5.27 indicates that most users are sparsely connected, which accurately represents the reality of social networks as few nodes act as influencers while the majority of users have fewer connections. All these results align with the “small world phenomenon” which states that a small number of highly connected nodes are the ones that help maintain connectivity amongst the network.

In order to identify tightly connected groups of users, I loaded the ground-truth communities from the com-youtube.all.cmty.txt file using the load_communities function. The function reads the file, where each line represents a community and lusts the node IDs that belong to that community, the function then assigns a unique community ID (starting from 0) to each line in the file while parsing the node IDs on each line and mapping each node to its community ID using a HashMap<u32, u32>. The output revealed that Community 267 is the largest with 1460 members followed by a couple of other significant clusters. These results indicate that the YouTube network is composed of natural communities of users, which may represent groups of viewers with shared interests or subscribers of specific content creators.

Next, I analyzed the connectivity of the graph using a Breadth-First Search (BFS). BFS would help show the shortest path distances in unweighted graphs. The bfs_shortest_path function computes the shortest paths from a given source node to all other nodes in the graph. The function initializes a queue with the source node and distance 0, and a visited set tracked visited nodes to avoid revisiting the same node. The BFS explores neighbors level by level, storing the distance to each node in a HashMap<u32, usize>. By running BFS from Node 1, I was able to determine that Node 1 can reach 1134890 nodes, which represents the entire graph. The average shortest path length was calculated to be 4.16, which indicates that users on YouTube are closely connected. This further aligns with the small world phenomenon as it indicates that most nodes can be reached from a single node with just a few iterations.

To measure the reach of users beyond their immediate neighbors, I implemented the degree distribution at distance 2 using the degree_distance_2 function. This function calculates the number of nodes that are two hops away from a given starting node. For each direct neighbor of the starting input node, it collects the neighbor’s neighbors, while avoiding counting the input node itself and counting neighbors that are already directly connected to the input node. The use of a HashSet helps to ensure that only unique nodes are counted. For Node 1, the output shows that it has 7147 neighbors at distance 2, indicating a significant number of users reachable within two hops. This highlights the indirect influence of well-connected nodes, as they can reach a much larger pool of users through their immediate connections.

I also made use of the Jaccard Similarity, calculated by the jaccard_similarity function, which measures the overlap between the neighbor sets of two nodes. This provides a quantitative measure of similarity between the two nodes based on their neighbors to help identify structural similarities. The function uses an enum to classify the similarity score, where it starts by converting the neighbor lists of two nodes into HashSet to ensure unique values. It then computes the intersection of common neighbors and the union of the total unique neighbors, and returns the ratio of intersection to union. The similarity was further categorized into High, Medium, or Low using the categories_similarity function for clearer output. Based on the output, comparing Node 1 and Node 2 returned a Jaccard similarity score of 0.004, which is categorized as Low. This result indicates that the two nodes have very little overlap in their connections, suggesting they belong to separate communities in the social network.

Lastly, I also implemented betweenness centrality using the betweenness_centrality_top_nodes function. Betweenness centrality measures how often a node lies on the shortest paths between pairs of other nodes, making it an effective measure of a node's role as a bridge between communities. Due to the computational complexity of betweenness centrality on large graphs, I decided to focus on the top 50 nodes with the highest degree to approximate their centrality since nodes with high degree are more likely to have high betweenness centrality. The progress of the computation is tracked using an indicatif create, which helps to provide a clear progress bar during the running of the code since it takes a while to run. The betweenness_centrality_top_nodes selects the top N nodes based on their degree where it then makes use of BFS to calculate the shortest paths to other nodes and accumulate dependencies to determine centrality scores. The output revealed that Node 363 has the highest betweenness centrality, with a score of 731888.5341, followed by several nodes like Node 1072 and Node 1034018. These nodes act as critical connectors and bridges in the network, facilitating the flow of influence between different communities. This output essentially helps us understand the importance of nodes in connecting otherwise disconnected communities in the YouTube social network.

The results from this project highlight several key findings about the YouTube social network. Highly connected nodes, such as Node 1072, act as hubs, while nodes like Node 363 play an essential role in the network to connect different communities. The network exhibits the small world phenomenon, with the average shortest path length of 4.16, suggesting that influence can spread rapidly across the platform. Community detection revealed tightly-knit groups of users, and the degree distribution at distance 2 emphasized the extended reach of central nodes. Overall, this project implemented multiple graph algorithms to analyze the YouTube social network. By combining degree distribution, BFS, neighbors and distance 2, Jaccard similarity, and betweenness centrality, I was able to detect communities, identify influential nodes, and understand the structure of connections within the graph more specifically. These insights could be used to model information flow, identify key influencers, and further explore engagement patterns in YouTube.