Nathan Chang

DS 210 Final Project - Centrality and Trust Analysis in Bitcoin Alpha Trust Weighted Signed Network Dataset

Introduction

This Project revolves around the Bitcoin Alpha trust weighted signed network

Dataset information: This is who-trusts-whom network of people who trade using Bitcoin on a platform called Bitcoin Alpha. Since Bitcoin users are anonymous, there is a need to maintain a record of users' reputation to prevent transactions with fraudulent and risky users. Members of Bitcoin Alpha rate other members in a scale of -10 (total distrust) to +10 (total trust) in steps of 1. This is the first explicit weighted signed directed network available for research.

The more detailed information of this network can be found in this link:

SNAP: Signed network datasets: Bitcoin Alpha web of trust network (stanford.edu)

Objective

The primary objective of this project is to perform an in-depth analysis of the Bitcoin Alpha trust-weighted signed network dataset to understand the dynamics of trust and centrality within this unique digital community.

Key questions include:

Who are the key influencers in the trust communities of the Bitcoin Alpha network, and what are their roles in the network?

Do key influencers have a higher mean trust score?

Answering these questions can help understand the relationship between centrality and trust within the Bitcoin Alpha network.

Implementation

The implementation of this project involves several key steps, leveraging the capabilities of the Rust programming language and various libraries for data processing and network analysis. The following sections outline the main components of the implementation process.

csv_parsing.rs

The first step involves taking in Bitcoin Alpha network dataset csv. We created a struct (Source, target, Rating) to save the imported data.

graph_analysis.rs

The purpose of this module is to analyze graph centrality with different rating system.

Here we had 5 functions: construct_graph, calculate_degree_centrality, calculate_and_print_pagerank, calculate_and_print_betweenness_centrality, and calculate_and_print_closeness_centrality.

construct_graph

This is fairly straightforward. It takes in record (the imported data), returns a Directed Graph <node, edge>. It iterates over the records to create nodes for each unique user and directed edges for their trust relationships, with trust scores as edge weights.

calculate_degree_centrality

This function is to calculate the degree centrality of nodes in a directed graph. It takes in a graph, return the sorted lists of nodes by in-degree and out-degree in the form of (node, unweighted in-degree, unweighted out-degree, mean weighted in-degree, mean weighted out-degree). The reason that I also include mean weighted in-degree, mean weighted out-degree is because this is my very first function, and initially I am interested in seeing is more centralized nodes has and gives a higher rating.

We first collect each detail (node, unweighted in-degree, unweighted out-degree, mean weighted in-degree, mean weighted out-degree) of each node. Next, we sort the nodes twice—first by unweighted in-degree and then by unweighted out-degree—resulting in two lists, then we return them. The more neighbors a node has the more it's central and highly connected, thus has an influence on the graph.

calculate_and_print_pagerank

This function calculates the PageRank of a graph. PageRank is for measuring the importance of nodes in a network, one of the ways of centrality measures.

I followed the instructions on https://crates.io/crates/simple-pagerank.

Note that here I only take the top 38 nodes. In fact, for every way of centrality measures, I also only take the top 38 nodes. The graph contains 3783 unique nodes, and I am interested in the top 1% of each way of centrality measures. Note: I will not be repeating this again.

calculate_and_print_betweenness_centrality, calculate_and_print_closeness_centrality

This function calculates the betweenness centrality, closeness centrality of all nodes in a graph, then prints the top 38 nodes.

Closeness centrality measures the average path length. A high closeness centrality score for a node signifies that it is relatively close to all other nodes in terms of trust relationships. Such nodes can be considered as having quicker and more direct access to the entire network, making them pivotal in spreading or gathering information and trust.

Betweenness centrality measures the percentage of shortest paths where our studied node lies in them. Nodes with high betweenness centrality scores are those that frequently lie on the shortest paths of trust between other users. They hold significant influence over the flow of trust within the network.

I followed the instructions on https://docs.rs/rustworkxcore/latest/rustworkx_core/centrality/fn.betweenness_centrality.html.

https://docs.rs/rustworkx-core/latest/rustworkx_core/centrality/fn.closeness_centrality.html

After calculating both scores, the function sorts these scores in descending order to identify the nodes with the highest betweenness centrality, then prints out the top 38 nodes along with their centrality scores.

trust_analysis.rs

The purpose of this module is to analyze trust score of different groups

We have 2 functions here: calculate_network_average_trust, analyze_trust_scores, and calculate_and_print_group_trust_scores

calculate_network_average_trust

This function computes the average trust score for the network. It sums up all the ratings from records and finds the average rating.

analyze_trust_scores

This function computes the average trust score for a set of nodes. It takes in a slice of record structs, a list of nodes. It iterates over each node, gets it trust rating, calculates the average. Next, it averages the average rating of nodes in the list. This final average represents the mean trust score for the subset of nodes under consideration.

calculate_and_print_group_trust_scores

I noticed that analyze_trust_scores will cause very repetitive code in main.rs. So, for the sake of "DRY", and also testing analyze_trust_scores, this function is designed. This takes in a slice of record structs, a list of nodes, and a string, and prints the calculated average trust score along with the name of the group.

main.rs

Running all the functions mentioned above.

Results and Discussion

The output is attached at GitHub. (ds210 final project output.csv and node_details_expanded.csv)

ds210 final project output.csv has each of the top 38 nodes for every metric, and its corresponding score.

The data can be seen as pairs. For example, node_pr (column A) and pagerank_score (Column B) are pairs. node_pr represents top in page rank and pagerank_score is the corresponding page rank score. C,D are pairs, E,F are pairs, G,H,I,J,K are pairs, and L,M,N,O,P are pairs.

node_details_expanded has a record of nodes repeated 3, 4 or 5 times, and which metric they repeated in. For instance, the first five rows have node 1 and all 5 centrality measurements.

Key influencers and their role

I used python to find nodes that are repeated in 1,2,3,4,5 times in those lists. For instance,

Nodes [1,8] are in all 5 of the top 38 nodes. Node [30] is in 4. The rest of them which repeats 1,2,or 3 times are in main.rs. (let groups = vec![….])

Here is my interpretation:

[1,8], Nodes that repeated across all 5 centrality measures:

These nodes appear in the top 1% across all 5 centrality measures. This indicates they are not only influential (high PageRank) but also serve as critical connectors or bridges (high Betweenness) and are close to other nodes in the network (high Closeness). Their high Degree Centrality suggests they have numerous direct connections, further emphasizing their importance in the network. They are likely key players in the network, possibly acting as major trusted entities or hubs in the transaction network.

[30], Nodes that repeated across 4 centrality measures:

The absence of Node 30 from the top tier of betweenness centrality may suggest that while it is a hub for direct interactions and is trusted or holds sway over a large portion of the network, it may not necessarily control the flow of transactions or information between other nodes. This could indicate that Node 30 operates within a densely connected cluster where multiple paths exist, reducing its role as the sole bridge and thus

its betweenness score. Nonetheless, Node 30 is likely to be a significant node for direct interactions and trust within the community.

Nodes Repeated 3 Times:

I have noticed a very interesting pattern here:

- Most of these nodes (24 out of 26) consistently appear in the PageRank ('node_pr') centrality measure
- For a majority of these nodes (23 out of 24), have a combination of PageRank, unweighted in-degree, and unweighted out-degree centrality measures.

The absence of these nodes in the top tier of both betweenness and closeness centrality measures, despite their high rankings in PageRank and both in-degree and out-degree centrality, might interpret:

- Limited Role as Intermediaries (Betweenness Centrality)
- Not the Quickest Path to Others (Closeness Centrality)

These nodes are influential and active within certain areas or communities in the network, but not as crucial connectors or the most efficient paths that link the entire network. They are important pieces of the network's puzzle but not necessarily the central or bridging pieces that connect all parts of it.

Nodes Repeated 1 or 2 Times:

Unfortunately, I do not have enough time to complete such a deep analysis as Nodes repeated 3 times. However, I will give a general idea here for nodes that repeat only once or twice in different centrality measures are likely to:

- Have specific roles within the Bitcoin Alpha network.
- While still important, but in a more limited or targeted way than nodes repeat more than 2 times

This is the end for the graph analysis. Next, we will discuss the trust scores among those nodes.

Correlation Between Centrality and Trustworthiness

Following is the output for trust scores:

Network Average Trust Score: 1.4639

Average Trust Score for top_in_4_or_5 group: 2.1227

Average Trust Score for top_in_3 group: 2.0833

Average Trust Score for top_in_2 group: 1.8016

Average Trust Score for top_in_1 group: 1.7266

Note the trust score scale is ranging from -10 to +10, implies that scores above zero indicate positive trust ratings, while scores below zero would indicate distrust.

The Network Average Trust Score was at 1.4639, which is positively skewed. Implying people in the network generally rate each other more positively than negatively by slightly.

Clearly, there is a positive correlation between centrality in the network and the level of trust obtained. Nodes that rank highly in multiple centrality measures tend to have higher trust scores.

Conclusion

Answering our key questions:

Who are the key influencers in the trust communities of the Bitcoin Alpha network, and what are their roles in the network?

Nodes that appear at the top across all five centrality measures are the key influencers. They do everything.

Do key influencers have a higher mean trust score? YES

Citations

Stanford Network Analysis Project (SNAP). "Bitcoin Alpha trust weighted signed network dataset."

S. Kumar, F. Spezzano, V.S. Subrahmanian, C. Faloutsos. Edge Weight Prediction in Weighted Signed Networks. IEEE International Conference on Data Mining (ICDM), 2016.

S. Kumar, B. Hooi, D. Makhija, M. Kumar, V.S. Subrahmanian, C. Faloutsos. REV2: Fraudulent User Prediction in Rating Platforms. 11th ACM International Conference on Web Searchand Data Mining (WSDM), 2018.

Towards Data Science. "Notes on Graph Theory: Centrality Measurements."

"Simple-pagerank." Crates.io.

"rustworkx-core." Documentation on Docs.rs.

OpenAI. (2023). ChatGPT [Large language model]. https://chat.openai.com