DS210 Final Project Report

Buning Fan
Collaborators: None

Dataset

Since the dataset I initially selected for my project proposal was too small to generate more than 1000 vertices, I decided to switch to a new dataset on Kaggle about Twitter friends: https://www.kaggle.com/datasets/hwassner/TwitterFriends. This dataset is a subset of a larger database focusing on gathering information about the friends of Twitter users, which are the accounts that a user follows. The primary goal is to analyze users' interests based on the accounts they follow and their connections to the hashtags they use.

Though the topic changed, the project idea I chose is still this one: "Graph clustering and partitioning: Take a network in which the identity of nodes is meaningful. Try to find k best representatives of all the graph vertices. How satisfied are you with the selection, given your previous beliefs about this network?" In my project, I utilized two columns: id (user ID, an unchangeable number), and friends (a list of IDs that the user follows). However, this dataset is too large to run within the desired time frame of under 10 minutes, making it unsuitable for loading in Rust. Moreover, the dataset contains more than 40,000 rows of data, making it difficult to process. Consequently, I cleaned the data and retained only the top 4,000 rows to carry out this project. I uploaded the new dataset on Kaggle: https://www.kaggle.com/datasets/ericafan/twitter-friends-cleaned.

Graph Construction

I utilize the ID and friend relationships to construct an undirected graph, representing each edge in the dataset. To reduce the size of the graph and eliminate isolated points, I retain only those friends from each row that also appear in other rows.

The dataset reveals that:

graph node_count 1237 edge_count 1998

This means that the graph consists of 1,237 nodes and 1,998 edges. On average, each node has approximately 1.57 friends.

Cluster Methodology

I utilize the K-means method to partition the graph into K clusters. I have implemented the distance and center functions within the graphcluster module of the code. Specifically, I utilize the **graph_get_distance** and **graph_get_center** functions. First, I use Dijkstra's algorithm, a graph traversal algorithm that is primarily used to find the shortest path between nodes in a graph, to obtain the node-to-node distance map, which is computed only once. This map is represented as a **HashMap<NodeIndex, HashMap<NodeIndex, i32>>**. I then pass a reference to this map to the distance and center functions to prevent unnecessary recalculations. If there is no connection between node A and node B, return the **MAX_DISTANCE = 100** value. This large value is particularly significant for small clusters.

Results

cluster 0 has size 412
cluster 1 has size 695
cluster 2 has size 133

Analysis

The results reveal interesting insights about the network's structure and user relationships. The graph consists of 1,237 nodes and 1,998 edges, which indicates that the average Twitter user in this subset follows approximately 1.57 friends. This could suggest that users tend to follow a small number of other users, potentially focusing on specific interests or close connections.

By implementing the K-means method for graph clustering, I partitioned the graph into 3 clusters. The size distribution of these clusters (412, 695, and 133) highlights that the network is not evenly divided, with one cluster being significantly larger than the other two. This uneven distribution could be indicative of a core group of users who share more common interests, while the other clusters represent smaller, more niche communities.

I utilized Dijkstra's algorithm to compute the node-to-node distance map, allowing us to calculate distances between nodes efficiently. This approach helped us avoid unnecessary recalculations and kept the computation time within the desired 10-minute time frame. Additionally, the use of MAX_DISTANCE for unconnected nodes ensured that small clusters were not adversely impacted by large distances.

I used the command "cargo run" to run the program and "cargo test" to verify if the tests passed. The results were all good.

Overall, this project demonstrates the effectiveness of graph clustering and partitioning techniques in understanding complex networks such as Twitter. The algorithms used in this project enabled us to analyze the interests of users based on their connections and the accounts they follow, providing valuable insights into the structure and dynamics of the network.

Reference

1. https://doc.rust-lang.org/stable/book/title-page.html
2. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
3. https://docs.rs/petgraph/latest/petgraph/