Social Network Analysis

William Cullian

Interface: Graph

Purpose and description of class: Abstract graph interface provides the basic methods for creating, printing and exporting a graph

Class: SocialNetworkGraph

Purpose and description of class: Represents the social network with an adjacency list graph object. Stores a list of vertices (each of which store their edges). Standard methods exist for building the graph and querying facets of the graph, but also added the method "evaluateEgonets", "getEgonets" and various helper methods here and in the Egonet subclass, for the "easy" part of this assignment. And for the "hard" part I added "analyzeCommunities" and it calls many methods in the CentralityGraph subclass, as well as reusing some of the helpers from the "easy" part. It also prints Egonets of the top users in the sub-communities.

Class: EgonetGraph

Purpose and description of class: Represents the egonet of a user. Stores it as an adjacency list. Also stores the inner connectivity average and user number for the center of the egonet. Is a subclass that extends SocialNetworkGraph. It contains the methods and data for creating an egonet of varying depth.

Class: CommunityGraph

Purpose and description of class: Represents the graph to Be used for community analysis. Stores it as an adjacency list. Also has fields necessary for the betweenness algorithm. Is a subclass that extends SocialNetworkGraph. It contains the methods and data for analyzing communities in graphs.

Class: Node

Purpose and description of class: Represents a node in a graph. Keeps a list of edges to adjacent nodes in the graph. I also added fields for centrality calculations. I felt it was easier and clearer to keep it all in the Node class without unnecessary sub-classes. Has toString method to facilitate printing graph.

Class: Edge

Purpose and description of class: Represents an edge in a graph. Stores references to each of the vertices the edge connects. Has compareTo method to make comparing edges easier. Has toString method to facilitate printing graph.

Interface: Data

Purpose and description of class: Abstract graph interface provides the basic methods for creating, printing and exporting a graph

Class: EgonetData

Purpose and description of class: A data class for the EgonetGraph sub-class that holds data for creating egonets.

Class: CentralityData

Purpose and description of class: A data class for the CentralityGraph sub-class that holds data for creating egonets.

Class: GraphLoader

Purpose and description of class: Reuse of the graph loader from previous projects. Written by USCD staff. Used to load graphs from csv files.

Overall Design Justification: I chose to use an adjacency list because it lends itself to BFS by having a list in each node. SocialNetworkGraph stores all the users and each user stores its friendships. One design decision was tough. The algorithm for my "getEgonets" method is supposed to return another graph, I decided to create a subclass for egonets to store the extra information stored in them. It makes the code easier to read, instead of adding fields to SocialNetworkGraph and creating new SocialNetworkGraph's every time. That way you know if it's the SocialNetworkGraph or an Egonet. I'm not sure why I am returning egonets, since my "evaluateEgonets" already prints the stats. But I feel like after calculating all that, it should be stored, not just thrown away. Following the EgonetGraph for the easy question, I created a CentralityGraph sub-class for analyzing sub communities, for the hard question. I also created data classes to separate the data for these sub-classes.