CSC 599: Social Networking Analysis

Spring 2015

By: Michael Katan & Christopher Lyons

Background of Social Network Analysis

Overview of discipline

- Roots in sociology, psychology, anthropology
- Applications in economics, communications, linguistics, information science, organizational studies
- Social networks already existed, but increase in number of social media users in past 10 years makes these networks more noticeable/more data is available

Attribute data vs. Relational data

- Attribute data: “attitudes and behaviors of agents” (content)
- Relational data: “contacts, ties” (connections)
- Currently more focus on attribute data than relational data

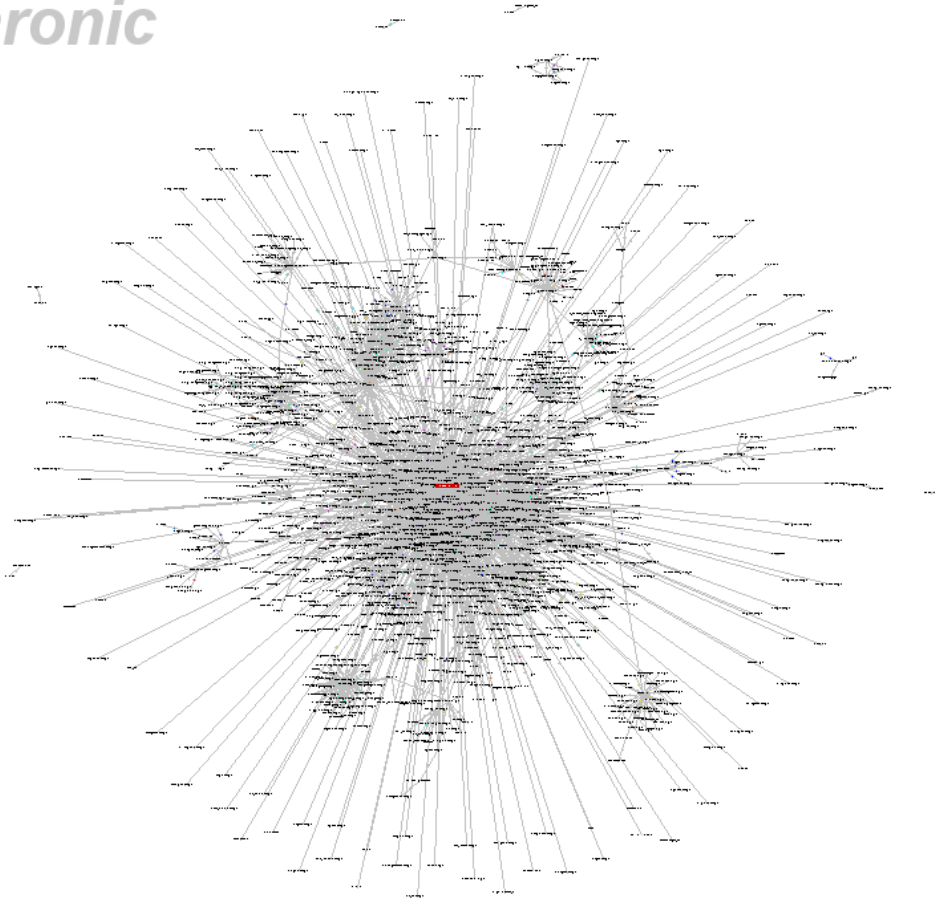
Examples

- Marketing strategies based on friends’ purchases (attribute data)
- NSA mapping terrorist organization members (relational data)
- Behavior analysis, text analysis (attribute data)

Background of Social Network Analysis cont.

Enron scandal (relational data)

- Using relational data based on emails - blobs indicate possible conspirator groups



Source: Enron, Jeffrey Heer. Figure 3 from http://homes.cs.washington.edu/~jheer/projects/enron/v1/enron_1_all.png

Locating Online Communities

Creighton Data Science group: organization analysis project

- Finding communities based on initial "seed" of individuals
- Project focuses on attribute data rather than relational data
- Can identify a target community, but approach cannot easily be extrapolated to other communities (cannot find sub-communities)

Use of self-identification vs. connections of users

- Self-identification: base communities on users' listed groups (e.g. Creighton, Omaha, baseball player, psychology major)
- Connections: base communities on users' connections (e.g. Facebook friend, Twitter follower, connected on LinkedIn)

Purpose & Overview

Locating online communities based on users' connections

- Objective: creating a framework for identifying sub-communities in a network of users
- Reliability: connection-based community helps ensures actual membership
- Using connections, especially "mutual" connections (i.e. both users follow one another, are friends on Facebook, etc.)

Overview of project's algorithmic approach

- Using a discrete network of connected users:
- (1) Calculate "degree of closeness" between all users in network (add up values based on lengths of paths between any two users)
- (2) Select "key individuals" who have high degree of closeness (group leaders)
- (3) Add remaining users to group with highest degree of closeness to self

Purpose & Overview cont.

Parameters

- Note: arbitrary parameters used throughout process (since project is meant to be a general framework, values can be modified for context)
- What counts as a path between users (limiting length, only unique paths)
- Calculating closeness
- Users' membership in group

Input/Output

Input

- Our main goal for input was a simple and precise format.
- so we chose to do .txt files with no header.
- each line was 1 person
  - each person was identified by unique name, then a colon, then a list of connections.

ex: `Alice: Bob Eve Carol Chuck Dave Matthew`

Output

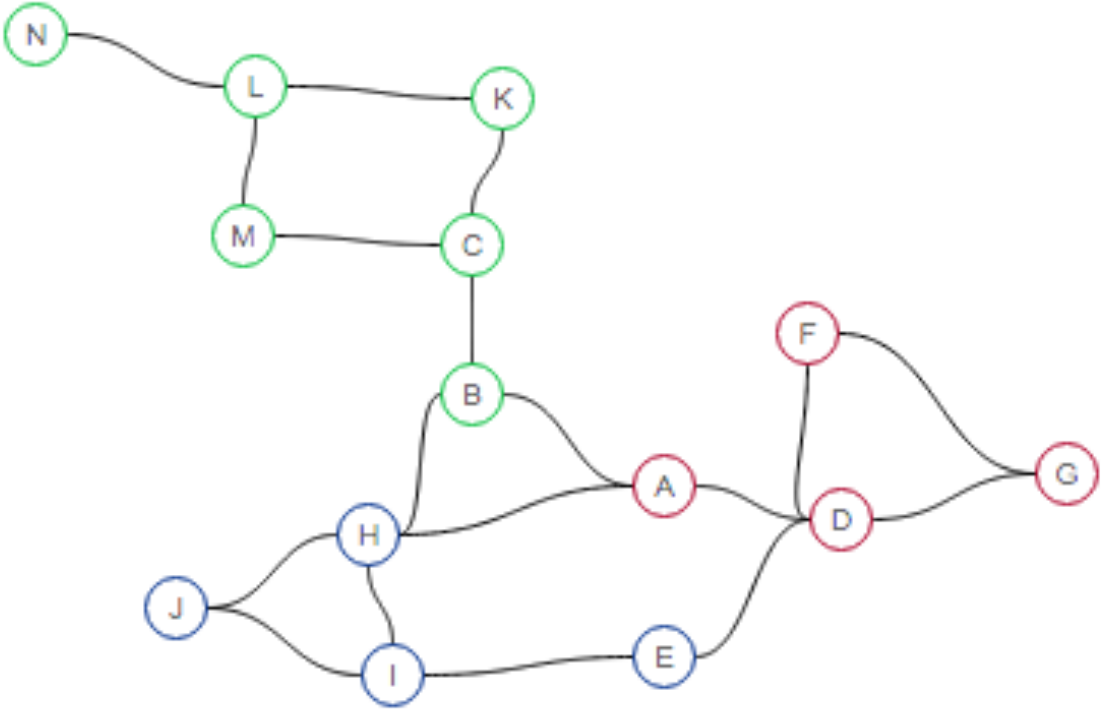
- Many different options for output
  - All paths connecting two individuals
  - How one individual is connected to the rest of the network
  - How each individual is ranked in the network based on their ability to influence the network
  - The graphical view of the network

I/O cont.

Input example

```
A: B D H
B: H A C
C: B K M
D: A F G E
E: D I
F: D G
G: F D
H: A I J B
I: E J H
J: I H
K: C L
L: K M N
M: C L
N: L
```

Output example



Program

generating random graphs

- needed large numbers of graphs for testing algorithm
- created class with 2 parameters, a number of people and density of network 1-100
- outputs a file in a specified location formated according to input specifications.
- each person is randomly connected to a mostly random number of people
- average number of connections is number*density/100.

Nodes

- store three sets of data on their particular person
  - A unique string to identify the person
  - a HashSet<String> containing all identifiers of people directly connected
  - a ArrayList<ArrayList<String>> that shows all paths to all other persons

    The outer dimension is each other person

    The inner dimension is each path to that person

Node Example

ID = A

Direct connections = (B, D, H)

Paths:

B((Direct))
C((B, C))
D((Direct))
E((D, E),(H, I, E))
F((D, F))
G((D, G))
H((Direct))
I((H, I),(D, E, I))
J((H, J),(D, E, I, J))
K((B, C, K))
L((B, C, M, L))
M((B, C, M))
N((B, C, M, L, N))

Algorithms

Initial pathfinding algorithm

- Breadth-first search, starting at one person, going till it reached the specified other person or ran out of things to attempt.
- had a Queue<String> of things to be checked that had the form Alice.David.Bob
  - this meant that Bob would be checked next, and the way to get to Bob was through Alice then David.
- Had a list of already used people that would be ignored in the Queue.
- Problems with algorithm
  - it was O(nodes*connections^{depth})
  - depth was not limited. had some go over 10 deep.
  - these two combined to make algorithm unusable for large test groups
  - we wanted to limit depth to 4, but even that wasnt enough to make this algorithm fast enough for large scale

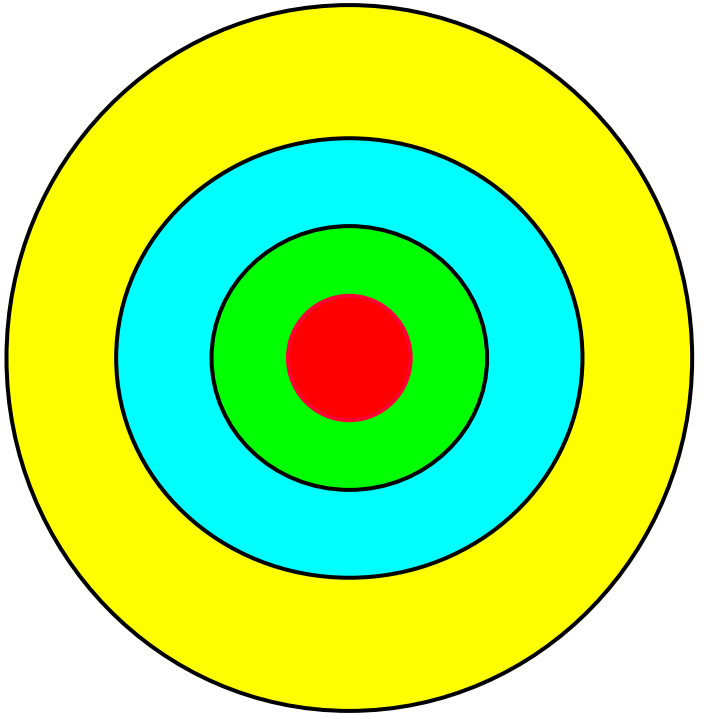
Algorithms cont.

Updated pathfinding algorithm

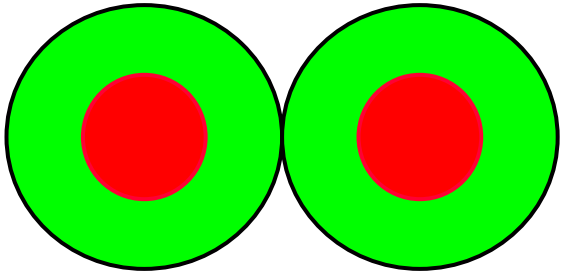
- Breadth-first search, starting at both ends
- had three HashSet<String>. One for each side and one for the middle.
- limited each sides depth to 2. (changeable if needed)
- algorithm checked one level of side A. then one level of side B.
  - each were added to the side HashSets and the middle
  - if conflict was found only in middle, that was the center node. otherwise repeated with next level
- This only found the middle node, however we used it recursively to find the whole path
- This algorithm has O(# of nodes*connections^2)

Algorithms cont.

Original search area



new search area



Algorithms cont.

Closeness algorithm

- Has a value of 1.0 if its a direct connection. Otherwise:
- Starts at 0.0 then for each possible path, takes the remaining amount to get from the current value to 1 divided by the length of the path, plus the old value
  - newValue = (1-current)/(length + current)

    this produces values <= 0.0 to <1.0

Key Individual ranking algorithm

- Key individuals are needed for many reasons:
  - Identifying communities and subcommunities
  - reaching the entire network as efficiently as possible
- Key individuals are selected by totaling their values from the closeness algorithm for every person in the network
- generally sqrt(# of nodes) individuals are selected

Algorithms cont.

A has a closeness value of 1.0 for B, D, and H.

it has a value of .5 for C, F, and G (½)

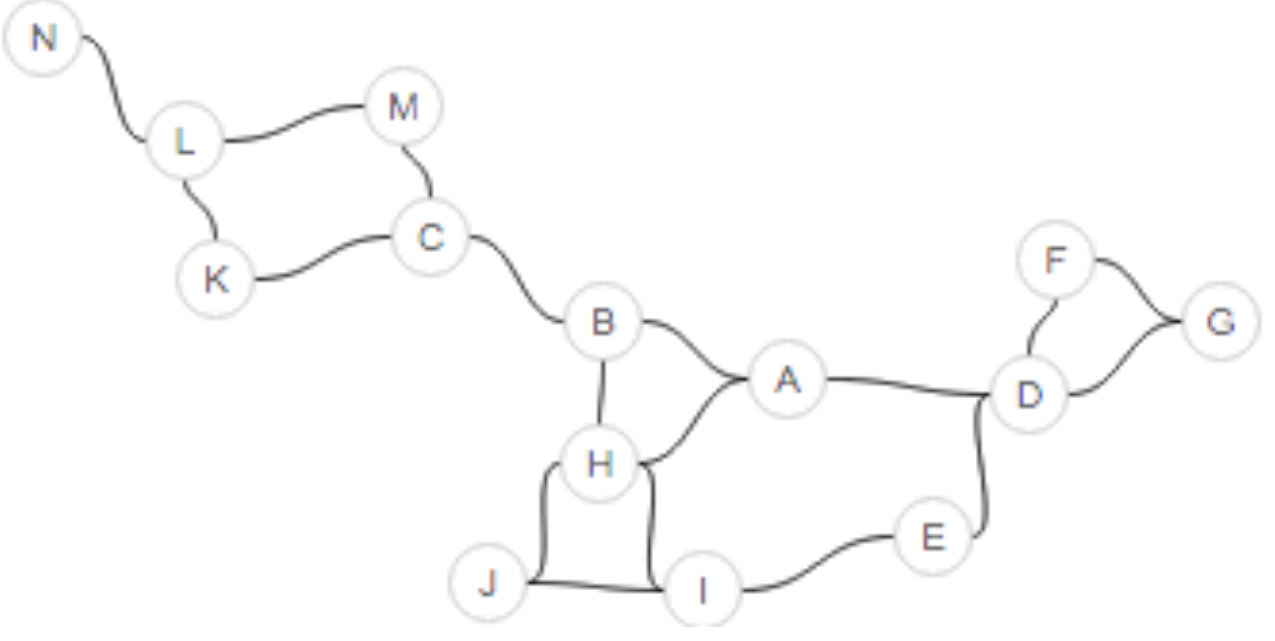
it has a value of .33 repeating for K and M (⅓)

it has a value of .25 for L (¼)

it has a value of .66 repeating for E and I (½ + ½*⅓)

it has a value of .625 for J (½ + ½*¼)

it has a value of 0.0 for N since there is no path 4 or less



Algorithms

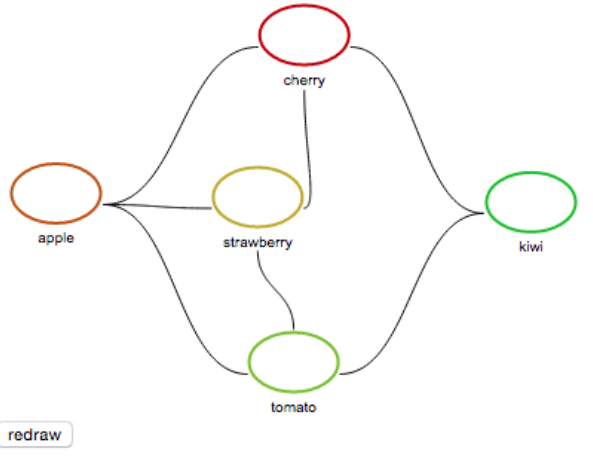
Key individuals as group leaders

- Add remaining users into groups based on closeness to group leader

Visualization: Dracula Graph library

- Display interactive graphs, JavaScript, created by Johann Strathausen, available at github.com/strathausen/dracula

```
var g = new Graph();

g.addEdge("strawberry", "cherry");
g.addEdge("strawberry", "apple");
g.addEdge("strawberry", "tomato");

g.addEdge("tomato", "apple");
g.addEdge("tomato", "kiwi");

g.addEdge("cherry", "apple");
g.addEdge("cherry", "kiwi");

var layouter = new Graph.Layout.Spring(g);
layouter.layout();

var renderer = new Graph.Renderer.Raphael('canvas', g, 400, 300);
renderer.draw();
```



Future work

- Update key individual grouping process
- Testing on “real world” social networks
- User-chosen parameters
- Direct connection to Twitter, Facebook, etc.

Sources

McGrath, Ryan. *Twython*. (2013). GitHub repository, https://github.com/ryanmcgrath/twython

Otte, E. & Rousseau, R. (2002). *Social network analysis: a powerful strategy, also for the information sciences*. *Journal of Information Science*, *28*(6), 441-453.

Scott, John. (2000). *Social Network Analysis: A Handbook*. London: SAGE Publications.

Strathausen, Johann P. *Dracula.* (2010). GitHub repository, https://github.com/strathausen/dracula