COMP7630 – Web Intelligence and its Applications

Social Network Analysis

Valentino Santucci

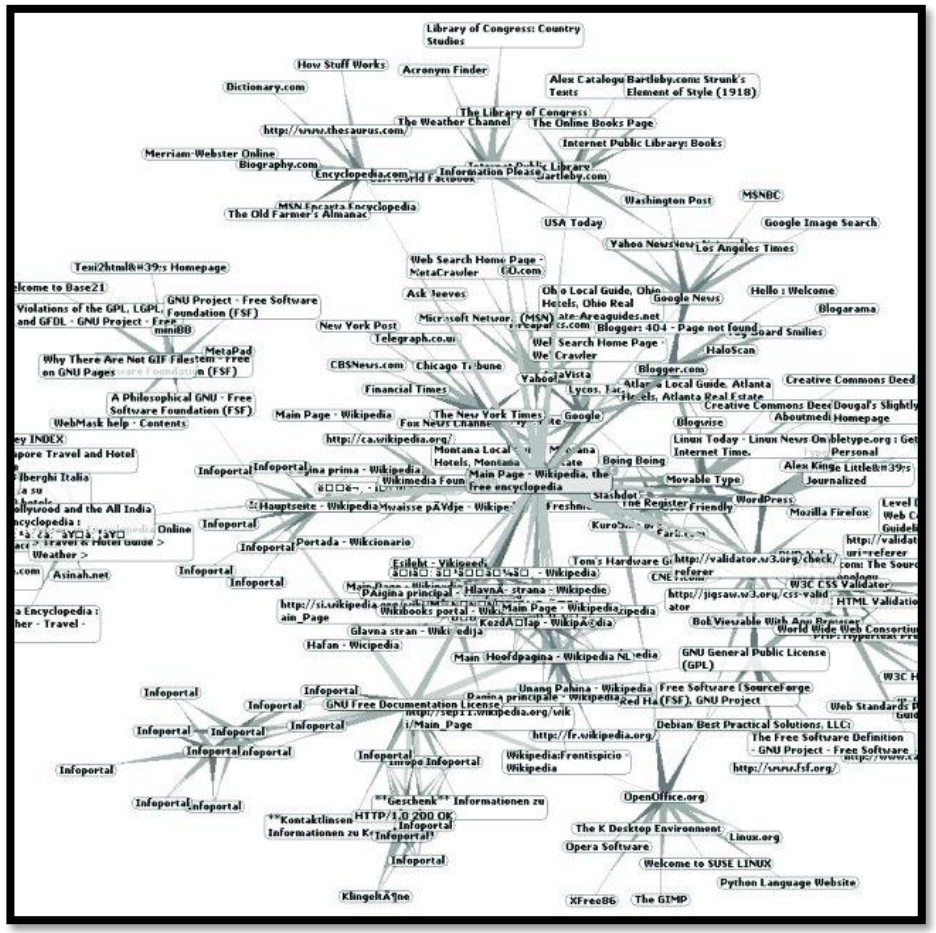
(valentino.santucci@unistrapg.it)

Outline

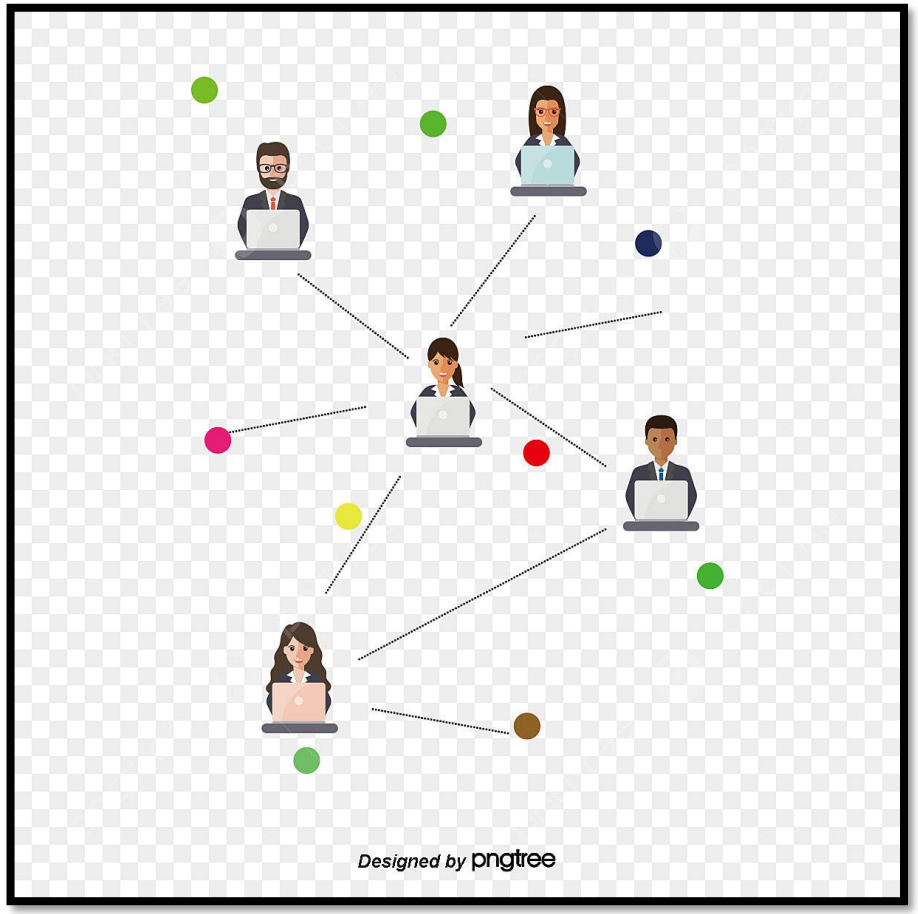
- Web and Social Networks
- Node Influence and Centrality
  - How you are connected to neighbors
  - How you connect others and how you reach out to others
- Similarity among nodes
- Community Analysis
  - Member-based Community Detection
  - Group-based Community Detection

Web and Social Networks

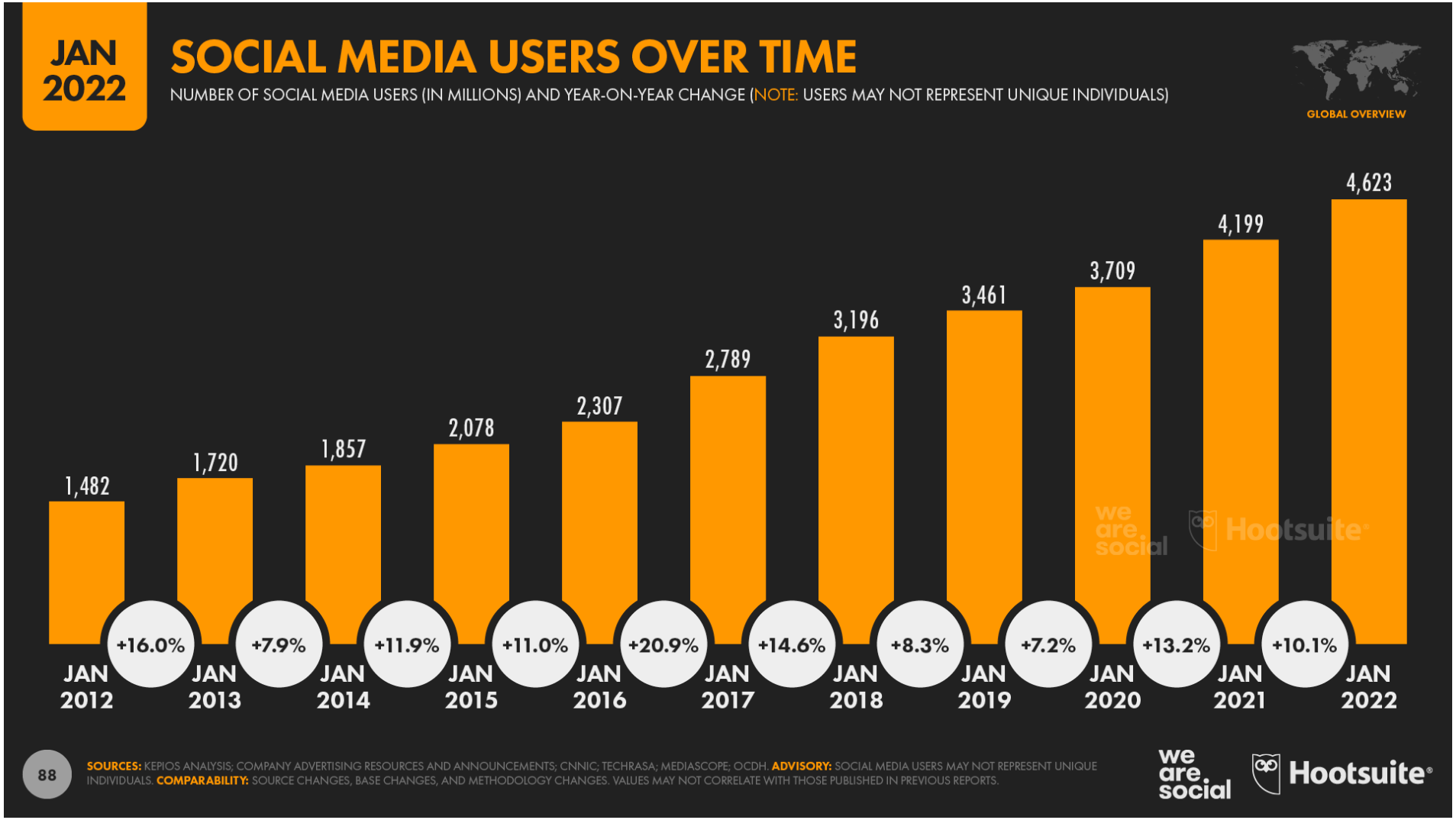
A small fraction of the WWW
(hyperlink structure)



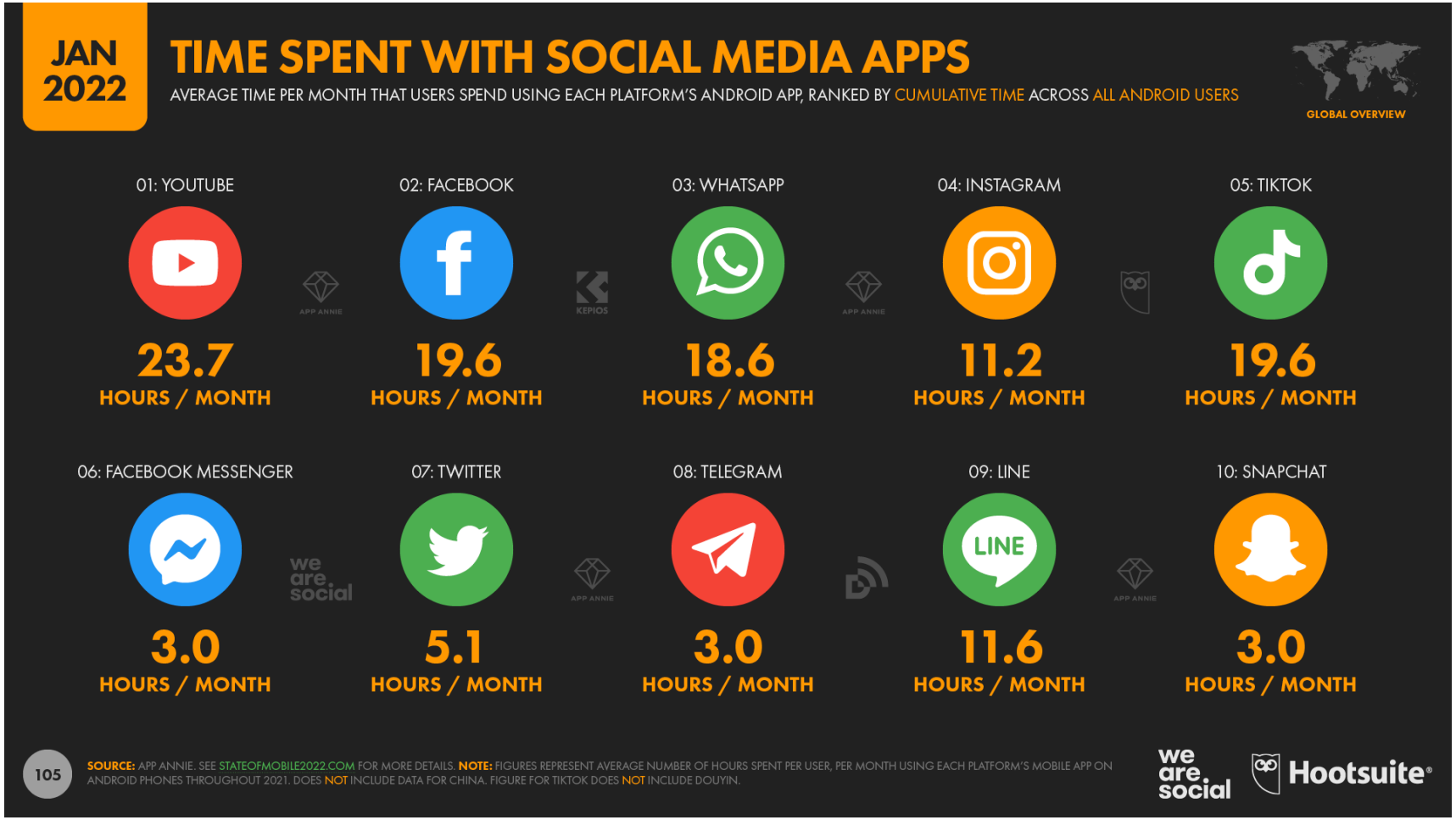
A small fraction of a social network
(social network structure)



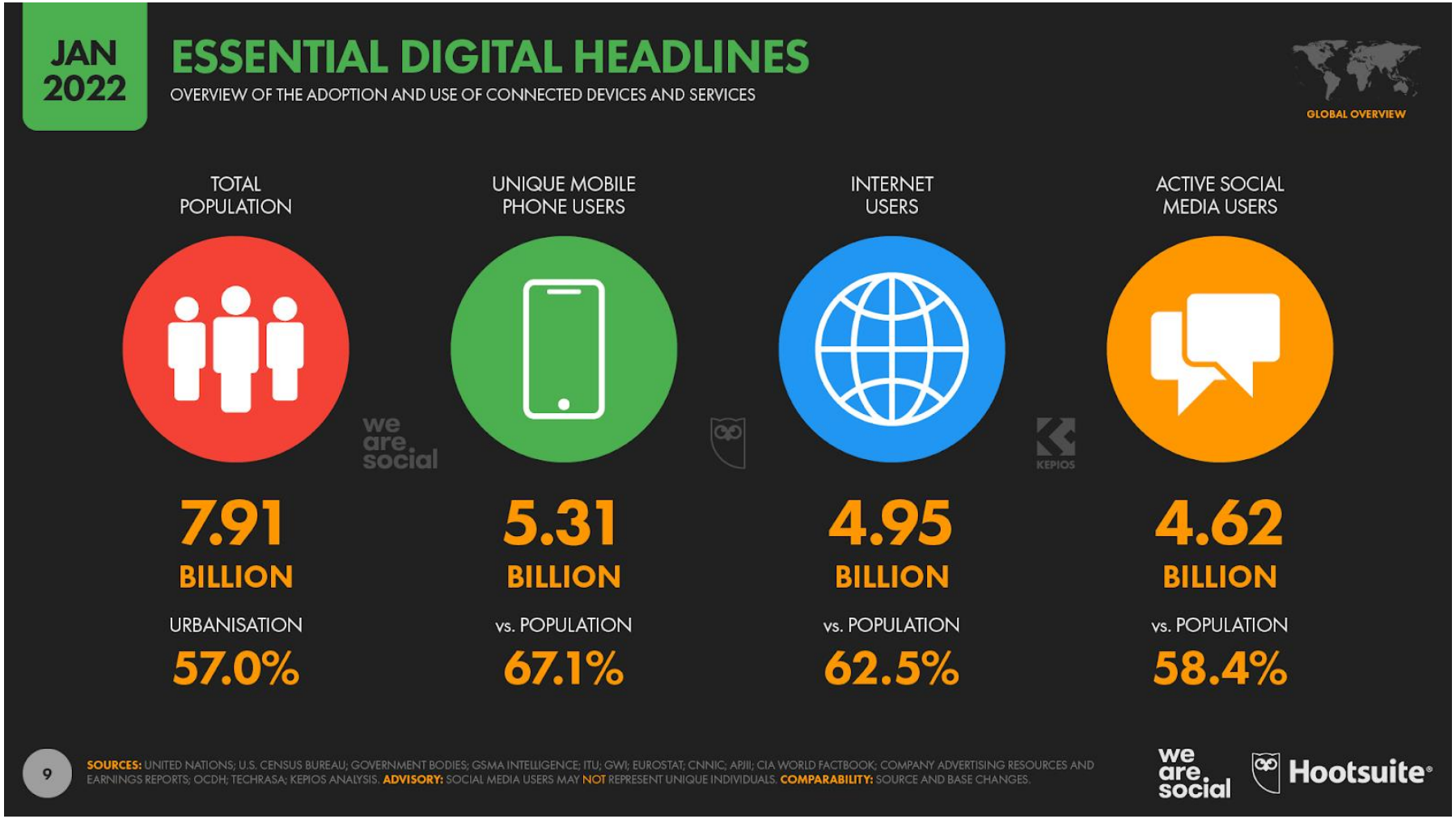
Is social network analysis useful …



Is social network analysis useful …

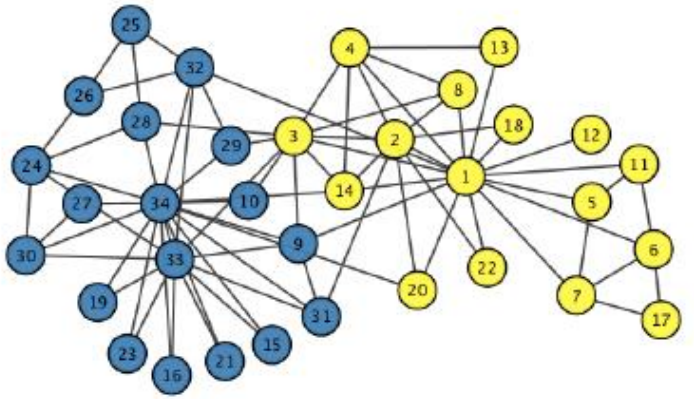


Is social network analysis useful …



Network Mining

- We are interested in mining Web structure ....
  - To locate important online resources
    - e.g., ranking web search results
  - To discover communities of online
    - e.g., organization of online new media (say, different political orientations)
- ... and social network structure
  - Important online resources -> influential users in social network



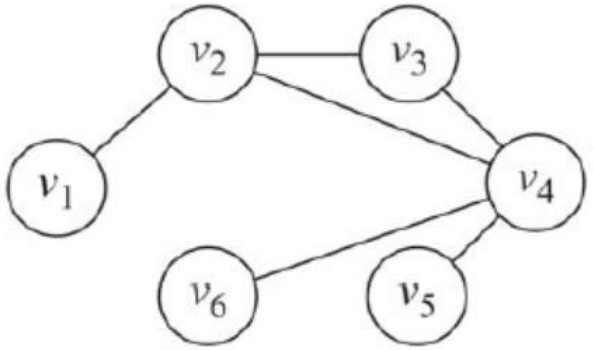
Outline

- Web and Social Networks
- Node Influence and Centrality
  - How you are connected to neighbors
  - How you connect others and how you reach out to others
- Similarity among nodes
- Community Analysis
  - Member-based Community Detection
  - Group-based Community Detection

How to measure influence?

Formalize Social Networks

- Social Network => Graph => Adjacency Matrix



(a) Graph

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	0	0
v_2	1	0	1	1	0	0
v_3	0	1	0	1	0	0
v_4	0	1	1	0	1	1
v_5	0	0	0	1	0	0
v_6	0	0	0	1	0	0

(b) Adjacency Matrix

$$A_{i,j} = \begin{cases} 1 & if\ v_i\ is\ connected\ to\ v_j \\ 0 & otherwise \end{cases}$$

Centrality Measures

- A centrality measure is used to quantify the importance or influence of a node within a network.
- Formally, it is a function that takes a node as input and returns a score.
- Node scores can then be used to rank the nodes of a network by importance.

- There are a variety of centrality measures that analyze different aspects of centrality or importance of a node

Outline

- Web and Social Networks
- Node Influence and Centrality
  - How you are connected to neighbors
  - How you connect others and how you reach out to others
- Similarity among nodes
- Community Analysis
  - Member-based Community Detection
  - Group-based Community Detection

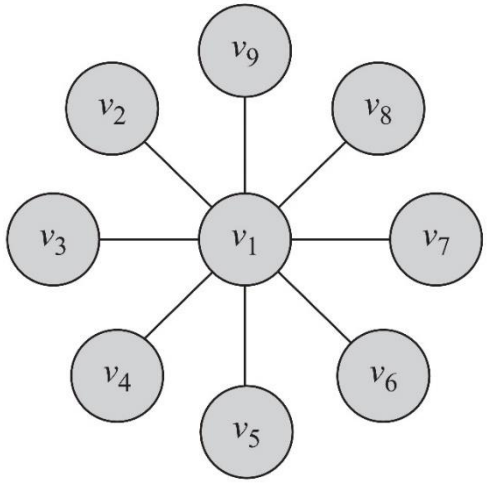
Degree Centrality

- **Degree centrality**: ranks nodes with more connections higher in terms of centrality

$$C_d(v_i) = d_i$$

- d_i is the degree (number of friends) for node v_i
  - i.e., the number of length-1 paths (can be generalized)

In this graph, degree centrality for node v_1 is d_1=8 and for all others is $d_j = 1, j \neq 1$



Degree Centrality in Directed Graphs

- In directed graphs, we can either use the in-degree, the out-degree, or the combination as the degree centrality value:
- In practice, mostly in-degree is used.

$$C_d(v_i) = d_i^{\text{in}} \quad (prestige)$$
$$C_d(v_i) = d_i^{\text{out}} \quad (gregariousness)$$
$$C_d(v_i) = d_i^{\text{in}} + d_i^{\text{out}}$$

d_i^{out} is the number of outgoing links for node v_i

Normalized Degree Centrality

- Normalized by the maximum possible degree

$$C_d^{\text{norm}}(v_i) = \frac{d_i}{n-1}$$

- Normalized by the maximum degree

$$C_d^{\max}(v_i) = \frac{d_i}{\max_j d_j}$$

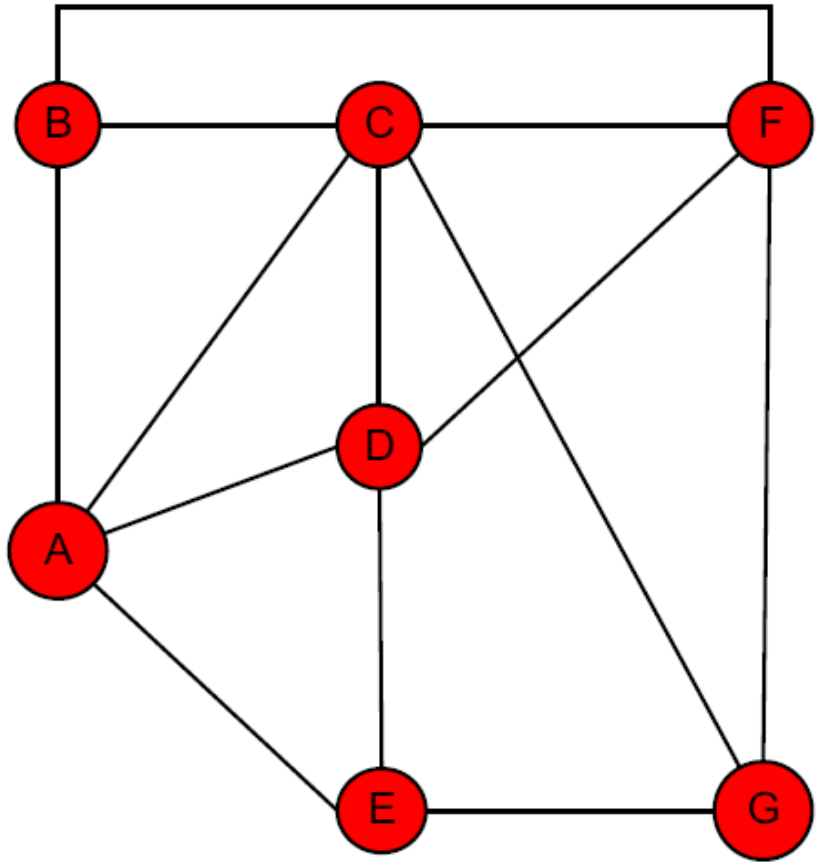
- Normalized by the degree sum

$$C_d^{\text{sum}}(v_i) = \frac{d_i}{\sum_j d_j} = \frac{d_i}{2|E|} = \frac{d_i}{2m}$$

Degree Centrality (Example – Undirected Graph)

Centrality Used:

= Degree normalized by the maximum possible degree.

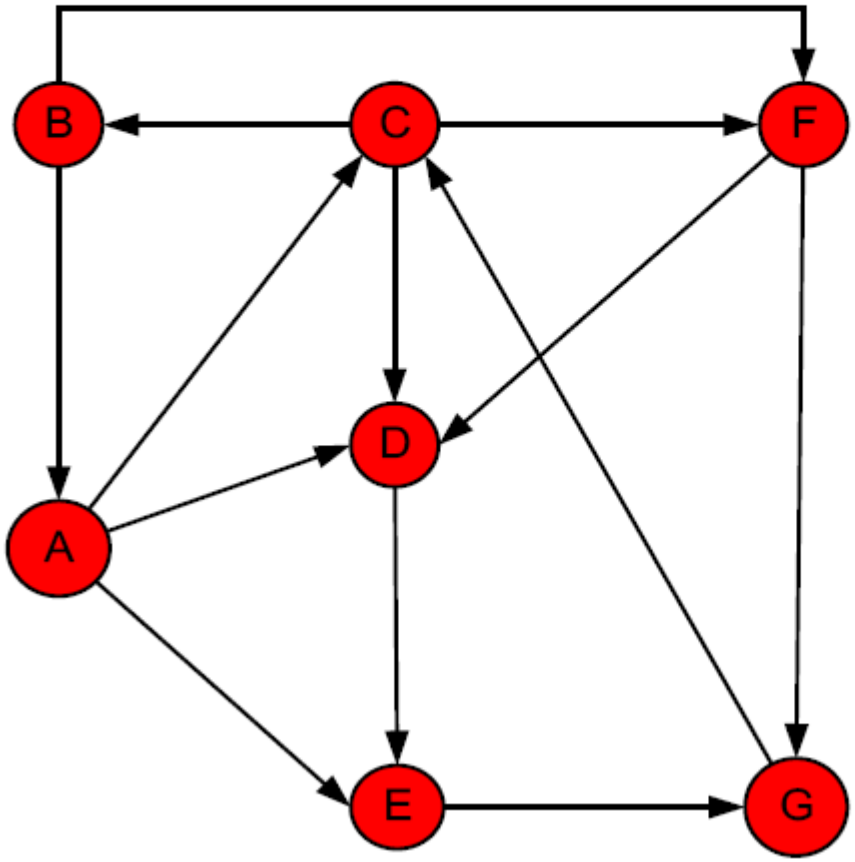


Node	Out-Degree	Centrality	Rank
A	4	4/6=2/3	2
B	3	3/6=1/2	5
C	5	5/6	1
D	4	4/6=2/3	2
E	3	3/6=1/2	5
F	4	4/6=2/3	2
G	3	3/6=1/2	5

Degree Centrality (Example – Directed Graph)

Centrality Used:

= Out-degree normalized by the maximum possible degree.



Node	In-Degree	Out-Degree	Centrality	Rank
A	1	3	3/6=1/2	1
B	1	2	2/6=1/3	3
C	2	3	3/6=1/2	1
D	3	1	1/6	5
E	2	1	1/6	5
F	2	2	2/6=1/3	3
G	2	1	1/6	5

Eigenvector Centrality

- Limitation of Degree Centrality
  - Having more friends does NOT guarantee influential
  - Having more important friends provides a stronger signal

- Eigenvector Centrality
  - Generalizes degree centrality by incorporating the importance of the neighbors
  - Key idea: Each node should carry and transmit a score indicating its importance

Eigenvector Centrality

- Denote the eigenvector centrality of a node v_i as $c_e(v_i)$ to indicate its importance.

- We want the value of $c_e(v_i)$ to be higher when important neighbors (i.e., node v_j with high $c_e(v_j)$) point to it (can you see circular relationship?).

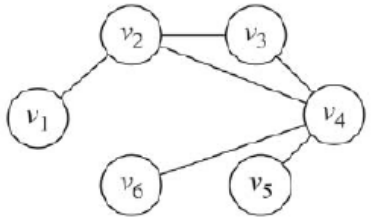
- We can compute v_i's centrality $c_e(v_i)$ as the ***summation of its neighbors' centralities***.

Eigenvector Centrality

- We can assume that v_i's centrality is the weighted sum of its neighbors' centralities

$$c_e(v_i) = \frac{1}{\lambda} \sum_{j=1}^{N} A_{j,i} c_e(v_j)$$

Summation of Neighbors' Centralities



(a) Graph

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	0	0
v_2	1	0	1	1	0	0
v_3	0	1	0	1	0	0
v_4	0	1	1	0	1	1
v_5	0	0	0	1	0	0
v_6	0	0	0	1	0	0

(b) Adjacency Matrix

$$\boldsymbol{c_e(v_i) = \frac{1}{\lambda}\sum_{j=1}^{6} A_{j,i} c_e(v_j)}$$

$$c_e(v_1) = \frac{1}{\lambda}\left(c_e(v_2)\right)$$

$$c_e(v_2) = \frac{1}{\lambda}\left(c_e(v_1) + c_e(v_3) + c_e(v_4)\right)$$

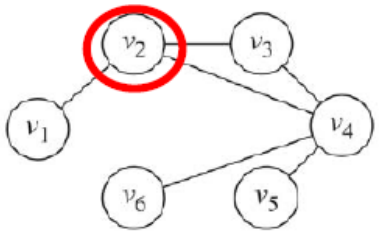
$$c_e(v_3) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_4)\right)$$

$$c_e(v_4) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_3) + c_e(v_5) + c_e(v_6)\right)$$

$$c_e(v_5) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

$$c_e(v_6) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

Summation of Neighbors' Centralities



(a) Graph

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	0	0
v_2	1	0	1	1	0	0
v_3	0	1	0	1	0	0
v_4	0	1	1	0	1	1
v_5	0	0	0	1	0	0
v_6	0	0	0	1	0	0

(b) Adjacency Matrix

$$\boldsymbol{c_e(v_i) = \frac{1}{\lambda}\sum_{j=1}^{6} A_{j,i} c_e(v_j)}$$

$$c_e(v_1) = \frac{1}{\lambda}\left(c_e(v_2)\right)$$

$$c_e(v_2) = \frac{1}{\lambda}\left(c_e(v_1) + c_e(v_3) + c_e(v_4)\right)$$

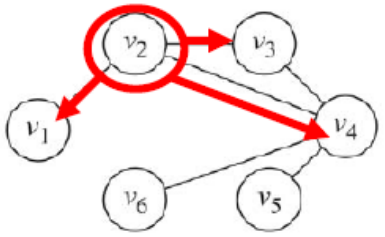
$$c_e(v_3) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_4)\right)$$

$$c_e(v_4) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_3) + c_e(v_5) + c_e(v_6)\right)$$

$$c_e(v_5) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

$$c_e(v_6) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

Summation of Neighbors' Centralities



(a) Graph

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	0	0
v_2	1	0	1	1	0	0
v_3	0	1	0	1	0	0
v_4	0	1	1	0	1	1
v_5	0	0	0	1	0	0
v_6	0	0	0	1	0	0

(b) Adjacency Matrix

$$\boldsymbol{c_e(v_i) = \frac{1}{\lambda}\sum_{j=1}^{6} A_{j,i} c_e(v_j)}$$

$$c_e(v_1) = \frac{1}{\lambda}\left(c_e(v_2)\right)$$

$$c_e(v_2) = \frac{1}{\lambda}\left(c_e(v_1) + c_e(v_3) + c_e(v_4)\right)$$

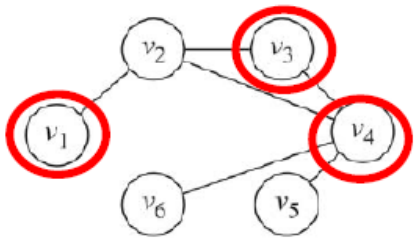
$$c_e(v_3) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_4)\right)$$

$$c_e(v_4) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_3) + c_e(v_5) + c_e(v_6)\right)$$

$$c_e(v_5) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

$$c_e(v_6) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

Summation of Neighbors' Centralities



(a) Graph

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	0	0
v_2	1	0	1	1	0	0
v_3	0	1	0	1	0	0
v_4	0	1	1	0	1	1
v_5	0	0	0	1	0	0
v_6	0	0	0	1	0	0

(b) Adjacency Matrix

$$c_e(v_i) = \frac{1}{\lambda}\sum_{j=1}^{6} A_{j,i}c_e(v_j)$$

$$c_e(v_1) = \frac{1}{\lambda}\left(c_e(v_2)\right)$$

$$c_e(v_2) = \frac{1}{\lambda}\left(c_e(v_1) + c_e(v_3) + c_e(v_4)\right)$$

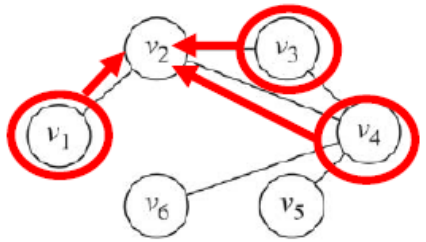
$$c_e(v_3) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_4)\right)$$

$$c_e(v_4) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_3) + c_e(v_5) + c_e(v_6)\right)$$

$$c_e(v_5) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

$$c_e(v_6) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

Summation of Neighbors' Centralities



(a) Graph

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	0	0
v_2	1	0	1	1	0	0
v_3	0	1	0	1	0	0
v_4	0	1	1	0	1	1
v_5	0	0	0	1	0	0
v_6	0	0	0	1	0	0

(b) Adjacency Matrix

$$c_e(v_i) = \frac{1}{\lambda}\sum_{j=1}^{6} A_{j,i}c_e(v_j)$$

$$c_e(v_1) = \frac{1}{\lambda}\left(c_e(v_2)\right)$$

$c_e(v_2)$ will be updated again!

$$c_e(v_2) = \frac{1}{\lambda}\left(c_e(v_1) + c_e(v_3) + c_e(v_4)\right)$$

$$c_e(v_3) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_4)\right)$$

Also, for the other nodes.

$$c_e(v_4) = \frac{1}{\lambda}\left(c_e(v_2) + c_e(v_3) + c_e(v_5) + c_e(v_6)\right)$$

$$c_e(v_5) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

$$c_e(v_6) = \frac{1}{\lambda}\left(c_e(v_4)\right)$$

Will it converge?

Eigenvector Centrality

- We can assume that v_i's centrality is the weighted sum of its neighbors' centralities

$$c_e(v_i) = \frac{1}{\lambda}\sum_{j=1}^{N} A_{j,i}c_e(v_j)$$

- For **matrix notation**,

$$\boldsymbol{c}_e = \left(c_e(v_1), c_e(v_2), \ldots, c_e(v_N)\right)^T$$

$$\lambda\, \boldsymbol{c}_e = A^T \boldsymbol{c}_e$$

- Here $\boldsymbol{c}_e$ is an *eigenvector* of A^T and λ is the corresponding *eigenvalue*.

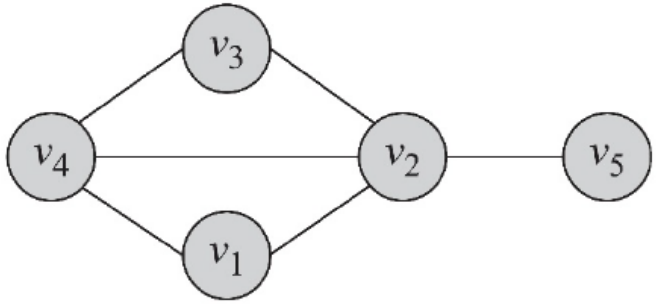
… but which eigenvector?

- We need positive values as centralities

Theorem 1 (Perron-Frobenius Theorem). *Let $A \in \mathbb{R}^{n \times n}$ represent the adjacency matrix for a [strongly] connected graph or $A : A_{i,j} > 0$ (i.e. a positive n by n matrix). There exists a positive real number (Perron-Frobenius eigenvalue) $\lambda_{\max}$, such that $\lambda_{\max}$ is an eigenvalue of A and any other eigenvalue of A is strictly smaller than $\lambda_{\max}$. Furthermore, there exists a corresponding eigenvector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ of A with eigenvalue $\lambda_{\max}$ such that $\forall v_i > 0$.*

- To obtain the eigenvector centrality for each node in the graph defined by A:
  - Compute the eigenvalues of A^T
  - Select the largest eigenvalue λ
  - Compute the corresponding eigenvector c_e of λ
  - c_e contains the eigenvector centralities of all the nodes in the graph

Eigenvector Centrality: Example



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \rightarrow \lambda = (2.68,\ -1.74,\ -1.27,\ 0.33,\ 0.00)$$

Eigenvalues Vector

$\lambda_{max} = 2.68 \rightarrow$

$$C_e = \begin{bmatrix} 0.4119 \\ 0.5825 \\ 0.4119 \\ 0.5237 \\ 0.2169 \end{bmatrix} \begin{matrix} \leftarrow v_1 \\ \leftarrow v_2 \\ \leftarrow v_3 \\ \leftarrow v_4 \\ \leftarrow v_5 \end{matrix}$$

Katz Centrality

- A major problem with eigenvector centrality arises when it deals with directed graphs

- Centrality only passes over *outgoing* edges
- What if there is a pit/sink node? (a node without outgoing edges)
- The pit nodes absorb all the centrality/importance!!!
- An issue for Directed Acyclic Graphs (DAGs)

- To resolve this problem we add bias term β to the centrality values for all nodes (a kind of "smoothing")

Eigenvector Centrality

$$C_{\mathrm{Katz}}(v_i) = \alpha \sum_{j=1}^{\mathrm{n}} A_{j,i} C_{\mathrm{Katz}}(v_j) + \beta$$

Katz Centrality

$$C_{\mathrm{Katz}}(v_i) = \alpha \sum_{j=1}^{\mathrm{n}} A_{j,i} C_{\mathrm{Katz}}(v_j) + \beta$$

Controlling term

Bias term

Rewriting equation in a vector form

$$\mathbf{C}_{\mathrm{Katz}} = \alpha A^T \mathbf{C}_{\mathrm{Katz}} + \beta \mathbf{1}$$

vector of all 1's

Katz centrality: $\mathbf{C}_{\mathrm{Katz}} = \beta(\mathbf{I} - \alpha A^T)^{-1} \cdot \mathbf{1}$

In practice we select $\boldsymbol{\alpha} < \mathbf{1}/\boldsymbol{\lambda}$, where λ is the largest eigenvalue of $\boldsymbol{A^T}$

PageRank Centrality

- Problem with Katz Centrality: once a node becomes an authority (high centrality), it passes all its centrality along all of its out-links
- This is less desirable since not everyone known by a well-known person is well-known

- PageRank modifies Eigenvector and Katz centralities so that:
  - Each connected neighbor of a node gets a fraction of the node's centrality value (divided by its outdegree)
  - If a node v_i has a significant number of neighbors with high centrality values pointing to it, the sum of those discounted values will still give the node v_i a high centrality value

PageRank Centrality

- Proper recurrence equation for PageRank

$$C_p(v_i) = \alpha \sum_{j=1}^{n} A_{j,i} \frac{C_p(v_j)}{d_j^{\text{out}}} + \beta$$

- Simplified recurrence equation for PageRank

$$c_e(v_i) = \sum_{j=1}^{N} \frac{A_{j,i}}{d_j^{out}} c_e(v_j)$$

Simplified PageRank

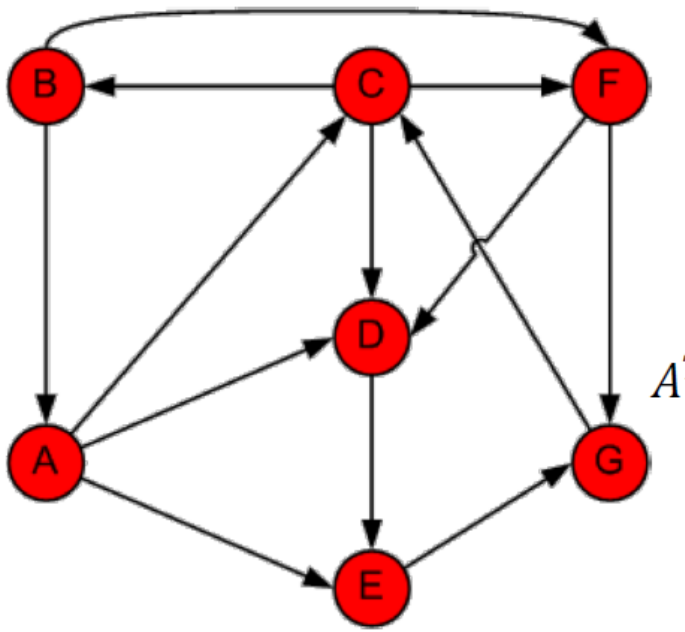
$$c_e(v_i) = \sum_{j=1}^{N} \frac{A_{j,i}}{d_j^{out}} c_e(v_j)$$

$$\boldsymbol{c}_e = A^T D^{-1} \boldsymbol{c}_e \quad D = diag(d_1^{out}, d_2^{out}, \ldots, d_n^{out})$$

We can then obtain simplified PageRank scores by performing eigenvalue decomposition on $A^T D^{-1}$.

Simplified PageRank

$$\boldsymbol{c}_e = A^T D^{-1} \boldsymbol{c}_e$$



$$A^T = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$A^T D^{-1} = \begin{pmatrix} 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/2 & 0 \\ 1/3 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1/2 & 0 \end{pmatrix}$$

column stochastic
(i.e., sum over column = 1)

PageRank: computational issues

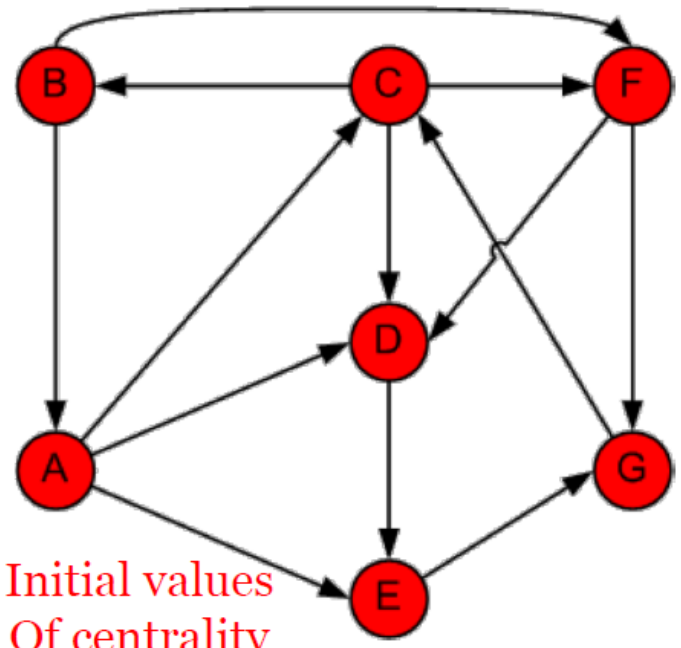
- PageRank is usually applied in search engines to measures the importance of a web page (Google)
- There are so many web pages that eigendecomposition is infeasible
- Power Iteration Method is computationally acceptable methodology for obtaining the PageRank

Simplified Page Rank by Power Iteration Method

$$\boldsymbol{c}_e = A^T D^{-1} \boldsymbol{c}_e$$

- Iterative numerical algorithm called power iteration method is commonly used.
- Initialize each node with arbitrary values of centrality and then update the centrality scores iteratively.

Simplified Page Rank by Power Iteration Method



Initial values
Of centrality

Initial values
of centrality

$$c_e^{new} \leftarrow A^T D^{-1} \boldsymbol{c}_e =$$

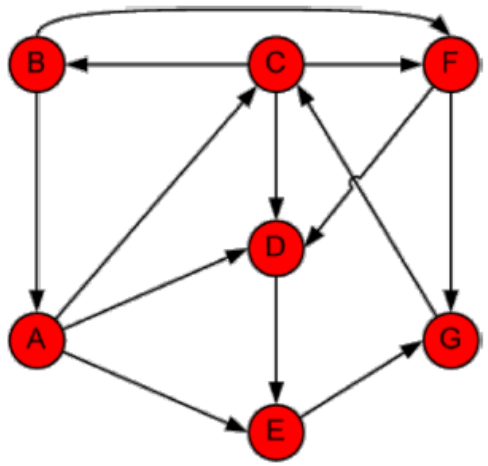
$$\begin{pmatrix} 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/2 & 0 \\ 1/3 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1/2 & 0 \end{pmatrix} \begin{bmatrix} 1/7 \leftarrow A \\ 1/7 \leftarrow B \\ 1/7 \leftarrow C \\ 1/7 \leftarrow D \\ 1/7 \leftarrow E \\ 1/7 \leftarrow F \\ 1/7 \leftarrow G \end{bmatrix}$$

Step	c_e(A)	c_e(B)	c_e(C)	c_e(D)	c_e(E)	c_e(F)	c_e(G)
0	1/7	1/7	1/7	1/7	1/7	1/7	1/7
update rule	c_e(B)/2	c_e(C)/3	c_e(A)/3 + c_e(G)	c_e(A)/3 + c_e(C)/3 + c_e(F)/2	c_e(A)/3 + c_e(D)	c_e(B)/2 + c_e(C)/3	c_e(E) + c_e(F)/2
1	0.071	0.048	0.190	0.167	0.190	0.119	0.214

Simplified Page Rank by Power Iteration Method

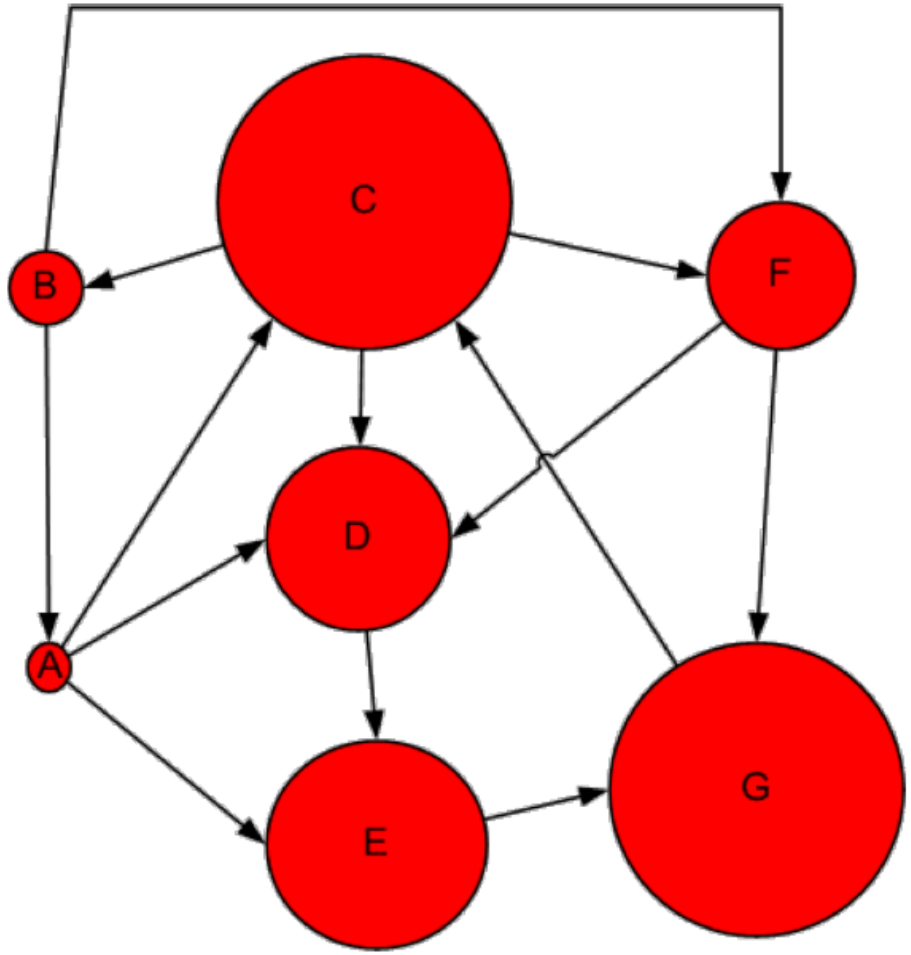
Step	$c_e(\mathrm{A})$	$c_e(\mathrm{B})$	$c_e(\mathrm{C})$	$c_e(\mathrm{D})$	$c_e(\mathrm{E})$	$c_e(\mathrm{F})$	$c_e(\mathrm{G})$	Sum
0	0.143	0.143	0.143	0.143	0.143	0.143	0.143	1.000
1	0.071	0.048	0.190	0.167	0.190	0.119	0.214	1.000
2	0.024	0.063	0.238	0.147	0.190	0.087	0.250	1.000
3	0.032	0.079	0.258	0.131	0.155	0.111	0.234	1.000
4	0.040	0.086	0.245	0.152	0.142	0.126	0.210	1.000
5	0.043	0.082	0.224	0.158	0.165	0.125	0.204	1.000
6	0.041	0.075	0.219	0.151	0.172	0.115	0.228	1.000
7	0.037	0.073	0.241	0.144	0.165	0.110	0.230	1.000
8	0.036	0.080	0.242	0.148	0.157	0.117	0.220	1.000
9	0.040	0.081	0.232	0.151	0.160	0.121	0.215	1.000
10	0.040	0.077	0.228	0.151	0.165	0.118	0.220	1.000
11	0.039	0.076	0.234	0.148	0.165	0.115	0.223	1.000
12	0.038	0.078	0.236	0.148	0.161	0.116	0.222	1.000
13	0.039	0.079	0.235	0.149	0.161	0.118	0.219	1.000
14	0.039	0.078	0.232	0.150	0.162	0.118	0.220	1.000
Rank	**7**	**6**	**1**	**4**	**3**	**5**	**2**	

PageRank: Typical Visualization



PageRank

Node	Rank
A	7
B	6
C	1
D	4
E	3
F	5
G	2

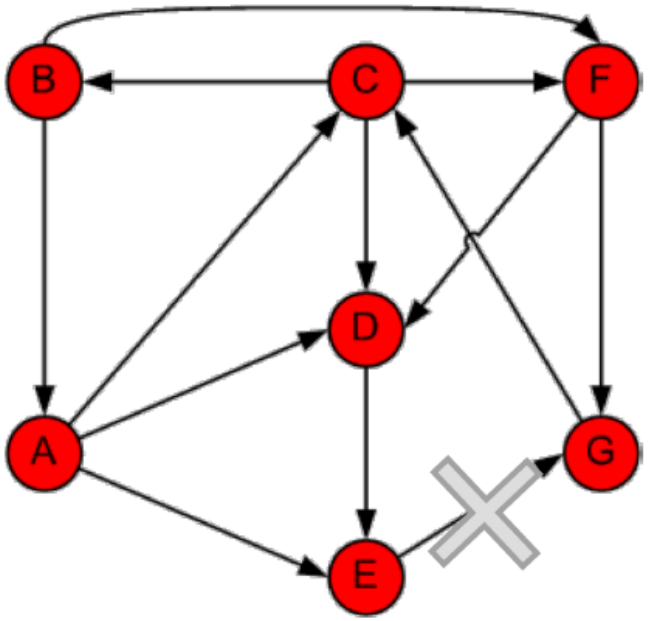


PageRank: issues of Dangling Links

$$c_e(v_i) = \sum_{j=1}^{N} \frac{A_{j,i}}{d_j^{out}} c_e(v_j)$$

What will be the converged value of $c_e(v_i)$ if some nodes have zero out-degree (that is some columns of $A^T D^{-1}$ are all zero)?

$$A^T D^{-1} = \begin{pmatrix} 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/2 & 0 \\ 1/3 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & 1/2 & 0 \end{pmatrix}$$



PageRank: issues of Dangling Links

Step	$c_e(\mathrm{A})$	$c_e(\mathrm{B})$	$c_e(\mathrm{C})$	$c_e(\mathrm{D})$	$c_e(\mathrm{E})$	$c_e(\mathrm{F})$	$c_e(\mathrm{G})$	Sum
0	0.143	0.143	0.143	0.143	0.143	0.143	0.143	1.000
1	0.071	0.048	0.190	0.167	0.190	0.119	0.071	0.857
2	0.024	0.063	0.095	0.147	0.190	0.087	0.060	0.667
3	0.032	0.032	0.067	0.083	0.155	0.063	0.044	0.476
4	0.016	0.022	0.054	0.065	0.094	0.038	0.032	0.321
5	0.011	0.018	0.037	0.043	0.070	0.029	0.019	0.228
6	0.009	0.012	0.023	0.031	0.046	0.021	0.015	0.157
7	0.006	0.008	0.018	0.021	0.034	0.014	0.011	0.111
8	0.004	0.006	0.013	0.015	0.023	0.010	0.007	0.077
9	0.003	0.004	0.008	0.010	0.016	0.007	0.005	0.054
10	0.002	0.003	0.006	0.007	0.011	0.005	0.004	0.038
11	0.001	0.002	0.004	0.005	0.008	0.003	0.002	0.026
12	0.001	0.001	0.003	0.004	0.006	0.002	0.002	0.018
13	0.001	0.001	0.002	0.002	0.004	0.002	0.001	0.013
14	0.000	0.001	0.001	0.002	0.003	0.001	0.001	0.009

?

PageRank with Damping Factor

When we surf and reach a page, assume that there is a probability $(1 - d)$ that you will move arbitrarily to a different page from the web.

$$c_e(v_i) = \frac{1-d}{N} + d \sum_{j=1}^{N} \frac{A_{j,i}}{d_j^{out}} c_e(v_j)$$

$$\boldsymbol{c}_e = \frac{1-d}{N} \mathbf{1} + d\, A^T\, D^{-1} \boldsymbol{c}_e$$

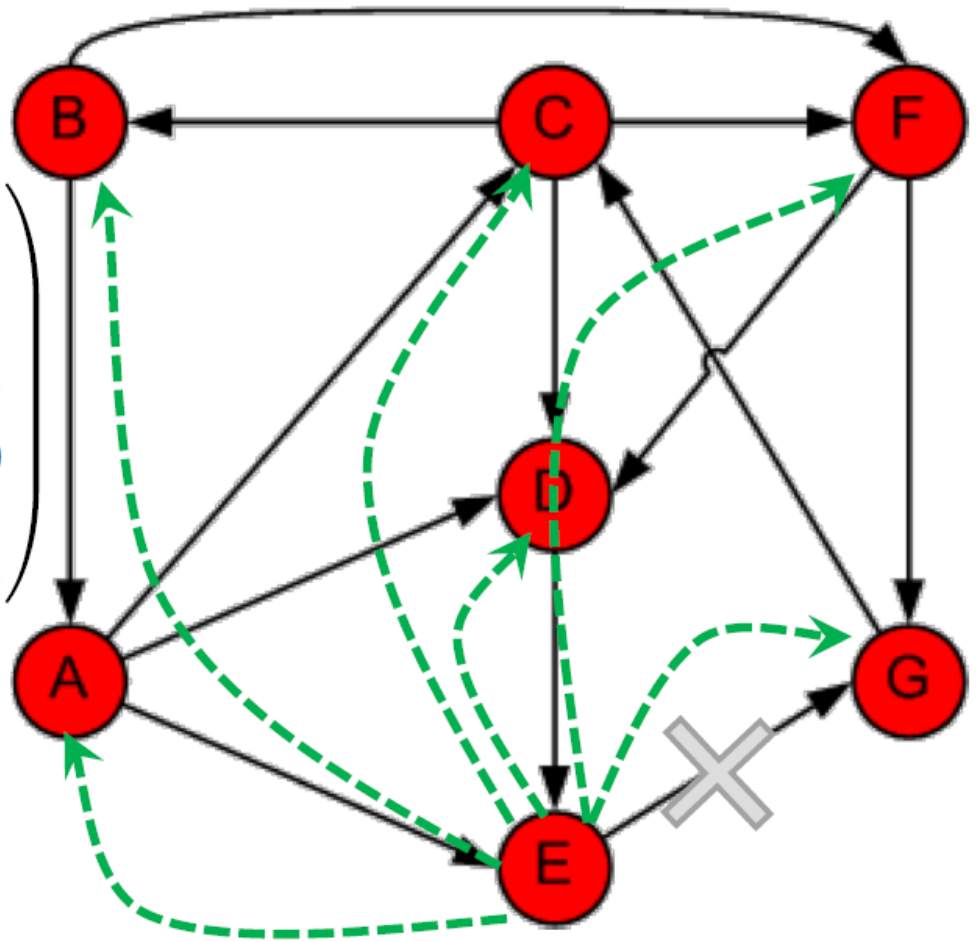
PageRank with Damping Factor (0.8)

Step	$c_e(\mathrm{A})$	$c_e(\mathrm{B})$	$c_e(\mathrm{C})$	$c_e(\mathrm{D})$	$c_e(\mathrm{E})$	$c_e(\mathrm{F})$	$c_e(\mathrm{G})$	Sum
0	0.143	0.143	0.143	0.143	0.143	0.143	0.143	1.000
1	0.086	0.067	0.181	0.162	0.181	0.124	0.086	0.886
2	0.055	0.077	0.120	0.149	0.181	0.103	0.078	0.764
3	0.059	0.061	0.106	0.117	0.163	0.091	0.070	0.666
4	0.053	0.057	0.100	0.109	0.138	0.081	0.065	0.603
5	0.051	0.055	0.095	0.102	0.130	0.078	0.061	0.572
6	0.051	0.054	0.091	0.099	0.124	0.076	0.060	0.554
7	0.050	0.053	0.090	0.097	0.121	0.074	0.059	0.544
8	0.050	0.053	0.089	0.096	0.119	0.074	0.058	0.538
9	0.050	0.052	0.088	0.095	0.118	0.073	0.058	0.535
10	0.050	0.052	0.088	0.095	0.118	0.073	0.058	0.533
11	0.049	0.052	0.088	0.095	0.118	0.073	0.058	0.533
12	0.049	0.052	0.088	0.094	0.117	0.073	0.058	0.532
13	0.049	0.052	0.088	0.094	0.117	0.073	0.058	0.532
14	0.049	0.052	0.088	0.094	0.117	0.073	0.058	0.531

Converged to non-zero! Good but the sum is still not ONE!

PageRank: Redistributing centrality

$$\begin{pmatrix} 0 & 1/2 & 0 & 0 & 1/6 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/6 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1/6 & 0 & 1 \\ 1/3 & 0 & 1/3 & 0 & 1/6 & 1/2 & 0 \\ 1/3 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/3 & 0 & 1/6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/6 & 1/2 & 0 \end{pmatrix}$$



PageRank with Damping + Redistributing

Step	$c_e(\mathrm{A})$	$c_e(\mathrm{B})$	$c_e(\mathrm{C})$	$c_e(\mathrm{D})$	$c_e(\mathrm{E})$	$c_e(\mathrm{F})$	$c_e(\mathrm{G})$	Sum
0	0.143	0.143	0.143	0.143	0.143	0.143	0.143	1.000
1	0.105	0.086	0.200	0.181	0.181	0.143	0.105	1.000
2	0.084	0.103	0.161	0.188	0.222	0.137	0.106	1.000
3	0.095	0.097	0.161	0.174	0.226	0.138	0.109	1.000
4	0.093	0.097	0.167	0.178	0.219	0.136	0.110	1.000
5	0.093	0.098	0.166	0.177	0.221	0.137	0.108	1.000
6	0.093	0.098	0.165	0.178	0.220	0.137	0.109	1.000
7	0.093	0.098	0.165	0.177	0.221	0.137	0.109	1.000
8	0.093	0.098	0.166	0.178	0.221	0.137	0.109	1.000
9	0.093	0.098	0.165	0.177	0.221	0.137	0.109	1.000
10	0.093	0.098	0.165	0.178	0.221	0.137	0.109	1.000
11	0.093	0.098	0.165	0.178	0.221	0.137	0.109	1.000
12	0.093	0.098	0.165	0.178	0.221	0.137	0.109	1.000
13	0.093	0.098	0.165	0.178	0.221	0.137	0.109	1.000
14	0.093	0.098	0.165	0.178	0.221	0.137	0.109	1.000

Converged and sum to ONE now!

Outline

- Web and Social Networks
- Node Influence and Centrality
  - How you are connected to neighbors
  - How you connect others and how you reach out to others
- Similarity among nodes
- Community Analysis
  - Member-based Community Detection
  - Group-based Community Detection

Betweeness Centrality

Another way of looking at centrality is by considering how important nodes are in connecting other nodes

$$C_b(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}}$$

σ_{st} The number of shortest paths from vertex s to t – *a.k.a.* information pathways

$\sigma_{st}(v_i)$ The number of **shortest paths** from s to t that pass through v_i

Normalizing Betweeness Centrality

- In the best case, node v_i is on all shortest paths from s to t, hence,

$$\frac{\sigma_{st}(v_i)}{\sigma_{st}} = 1$$

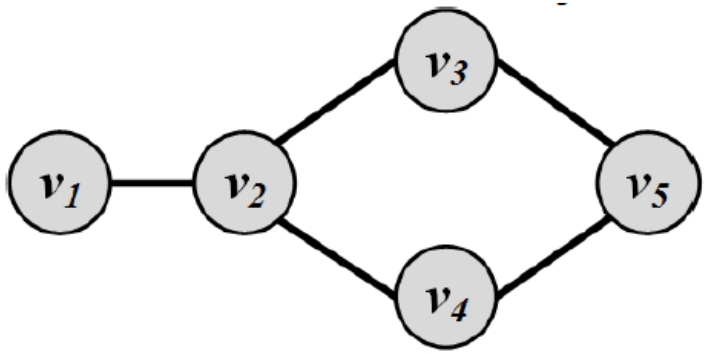
$$C_b(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}}$$

$$= \sum_{s \neq t \neq v_i} 1 = 2\binom{n-1}{2} = (n-1)(n-2)$$

Therefore, the maximum value is $(n-1)(n-2)$

Betweenness centrality: $C_b^{\text{norm}}(v_i) = \frac{C_b(v_i)}{2\binom{n-1}{2}}$

Betweeness Centrality: Example



$$C_b(v_2) = 2 \times (\underbrace{(1/1)}_{s=v_1,t=v_3} + \underbrace{(1/1)}_{s=v_1,t=v_4} + \underbrace{(2/2)}_{s=v_1,t=v_5} + \underbrace{(1/2)}_{s=v_3,t=v_4} + \underbrace{0}_{s=v_3,t=v_5} + \underbrace{0}_{s=v_4,t=v_5})$$
$$= 2 \times 3.5 = 7,$$
$$C_b(v_3) = 2 \times (\underbrace{0}_{s=v_1,t=v_2} + \underbrace{0}_{s=v_1,t=v_4} + \underbrace{(1/2)}_{s=v_1,t=v_5} + \underbrace{0}_{s=v_2,t=v_4} + \underbrace{(1/2)}_{s=v_2,t=v_5} + \underbrace{0}_{s=v_4,t=v_5})$$
$$= 2 \times 1.0 = 2,$$
$$C_b(v_4) = C_b(v_3) = 2 \times 1.0 = 2,$$
$$C_b(v_5) = 2 \times (\underbrace{0}_{s=v_1,t=v_2} + \underbrace{0}_{s=v_1,t=v_3} + \underbrace{0}_{s=v_1,t=v_4} + \underbrace{0}_{s=v_2,t=v_3} + \underbrace{0}_{s=v_2,t=v_4} + \underbrace{(1/2)}_{s=v_3,t=v_4})$$
$$= 2 \times 0.5 = 1,$$

Closeness Centrality

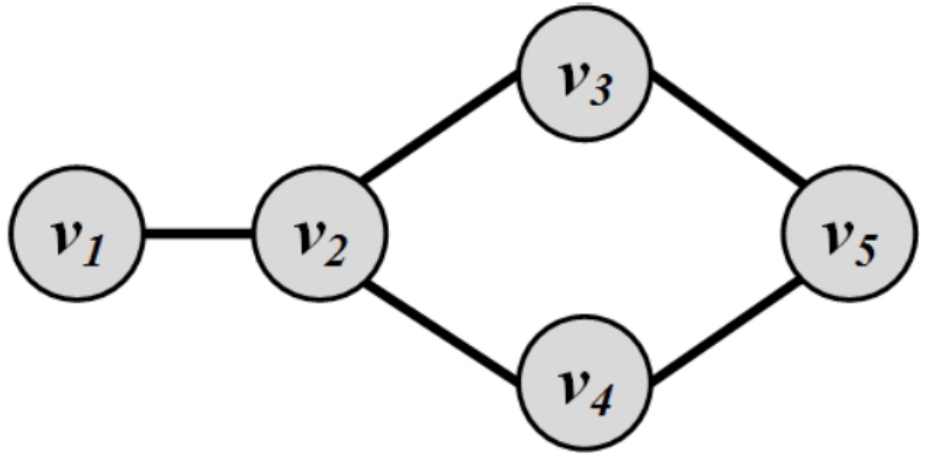
- The intuition is that influential/central nodes can quickly reach other nodes

- These nodes should have a smaller average shortest path length to others

- Closeness Centrality $C_c(v_i) = \frac{1}{\bar{l}_{v_i}}$

$\bar{l}_{v_i} = \frac{1}{n-1} \sum_{v_j \neq v_i} l_{i,j}$ where l_{ij} is the length of a shortest path from v_i to v_j

Closeness Centrality: Example



$$
\begin{aligned}
 & C_c(v_1) = 1 \,/\, (\,(1+2+2+3)/4\,) = 0.5, \\
 & C_c(v_2) = 1 \,/\, (\,(1+1+1+2)/4\,) = 0.8, \\
C_c(v_3) = \quad & C_b(v_4) = 1 \,/\, (\,(1+1+2+2)/4\,) = 0.66, \\
 & C_c(v_5) = 1 \,/\, (\,(1+1+2+3)/4\,) = 0.57.
\end{aligned}
$$

An interesting comparison

Comparing three centrality values: degree centrality, closeness centrality, betweenness centrality

- Generally, the 3 centrality types will be positively correlated
- When they are not (or low correlation), it usually reveals interesting information

Ego = node from which the analysis perspective is taken. Alter = a neighbor of the ego.

	Low Degree	**Low Closeness**	**Low Betweenness**
High Degree		*Node is embedded in a community that is far from the rest of the network*	*Ego's connections are redundant - communication bypasses the node*
High Closeness	*Key node connected to important/active alters*		*Probably multiple paths in the network, ego is near many people, but so are many others*
High Betweenness	*Ego's few ties are crucial for network flow*	*Very rare! Ego monopolizes the ties from a small number of people to many others.*	

Outline

- Web and Social Networks
- Node Influence and Centrality
  - How you are connected to neighbors
  - How you connect others and how you reach out to others
- Similarity among nodes
- Community Analysis
  - Member-based Community Detection
  - Group-based Community Detection

Nodes' Similarity

- Structural Equivalence:
  - We look at the neighborhood shared by two nodes
  - The size of this neighborhood defines how similar two nodes are

Structural Equivalence

- **Vertex similarity**: $\sigma(v_i, v_j) = |N(v_i) \cap N(v_j)|$

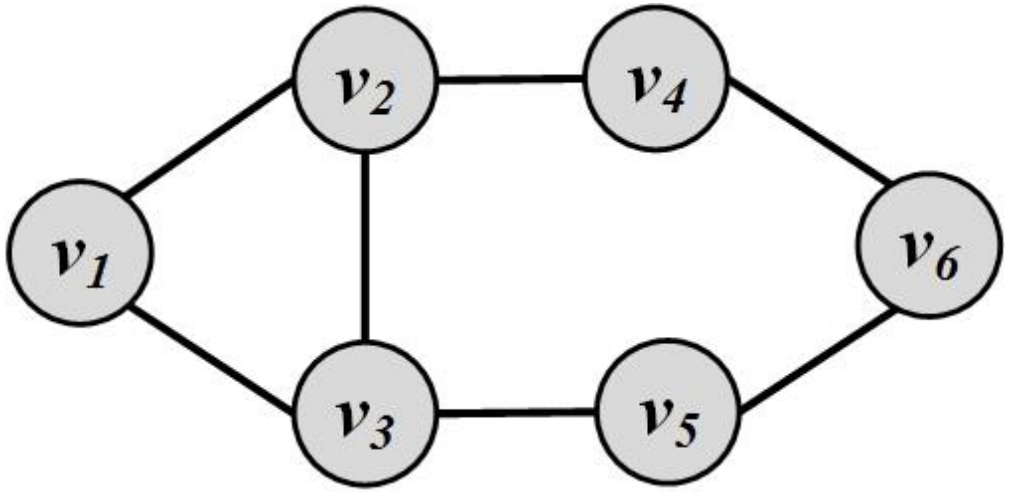
Normalize?

Jaccard Similarity: $\sigma_{Jaccard}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$

Cosine Similarity: $\sigma_{Cosine}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{\sqrt{|N(v_i)||N(v_j)|}}$

- The neighborhood $N(v)$ often excludes the node itself v.
  - **What can go wrong?**
    - Connected nodes not sharing a neighbor will be assigned zero similarity
  - **Solution:**
    - We can assume nodes are included in their neighborhoods

Similarity: Example



$$\sigma_{\text{Jaccard}}(v_2, v_5) = \frac{|\{v_1, v_3, v_4\} \cap \{v_3, v_6\}|}{|\{v_1, v_3, v_4, v_6\}|} = 0.25$$

$$\sigma_{\text{Cosine}}(v_2, v_5) = \frac{|\{v_1, v_3, v_4\} \cap \{v_3, v_6\}|}{\sqrt{|\{v_1, v_3, v_4\}||\{v_3, v_6\}|}} = 0.40.$$

Outline

- Web and Social Networks
- Node Influence and Centrality
  - How you are connected to neighbors
  - How you connect others and how you reach out to others
- Similarity among nodes
- Community Analysis
  - Member-based Community Detection
  - Group-based Community Detection

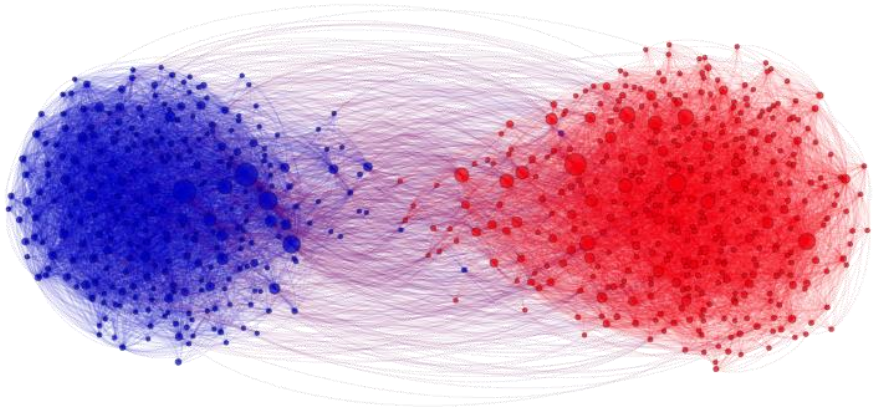
Community Analysis

Analyzing communities helps better understand users

- Users form groups based on their interests

Groups provide a clear global view of user interactions

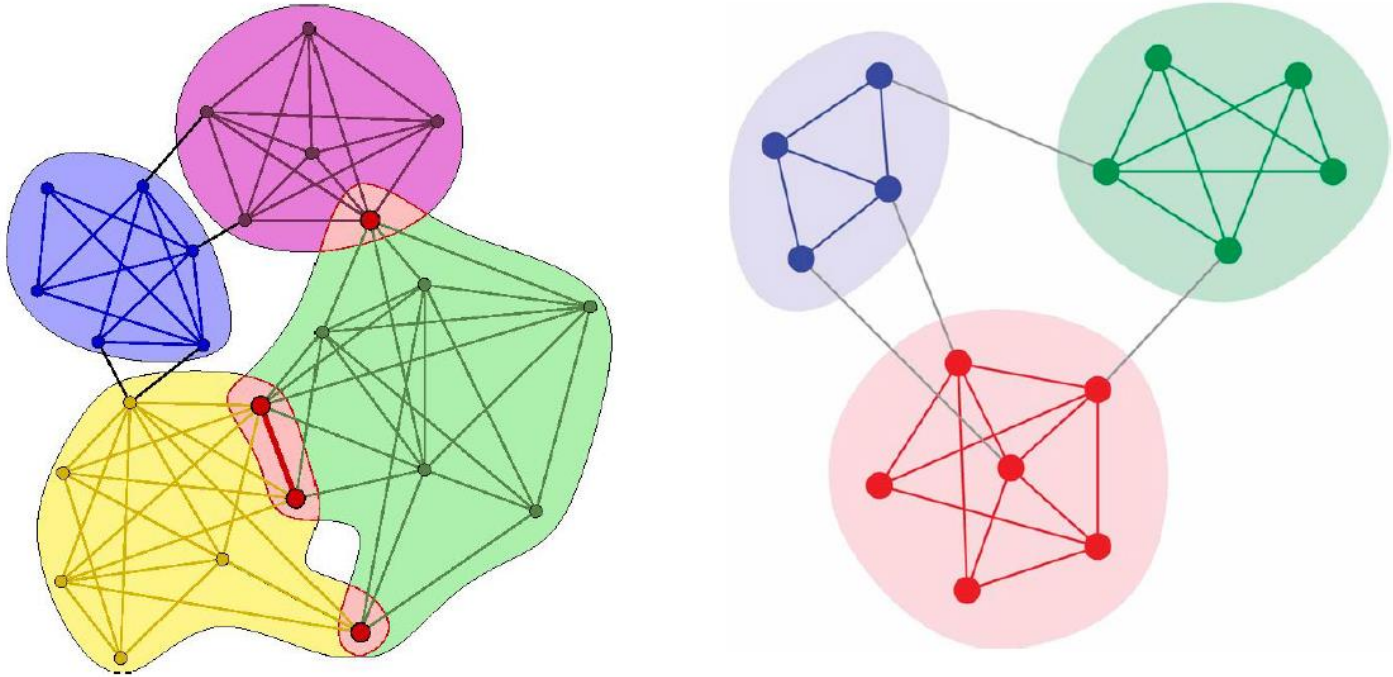
- E.g., find polarization



Social Media Communities

- **Explicit**: formed by user subscriptions
- **Implicit**: implicitly formed by social interactions (posting, comments, like, …)

Overlapping vs Disjoint Communities



Overlapping communities may be useful to find nodes that represents bridges between different communities and facilitate information diffusion.

In other applications, disjoint communities may be useful like e.g. in finding key leaders of different groups.

Community Detection

- The process of finding clusters of nodes (‘‘*communities*’’)
  - With **Strong** internal connections and
  - **Weak** connections between different communities

- Ideal decomposition of a large graph
  - Completely disjoint communities
  - There are no interactions between different communities.

- In practice,
  - find community partitions that are maximally decoupled.

Community Detection vs Clustering

Clustering

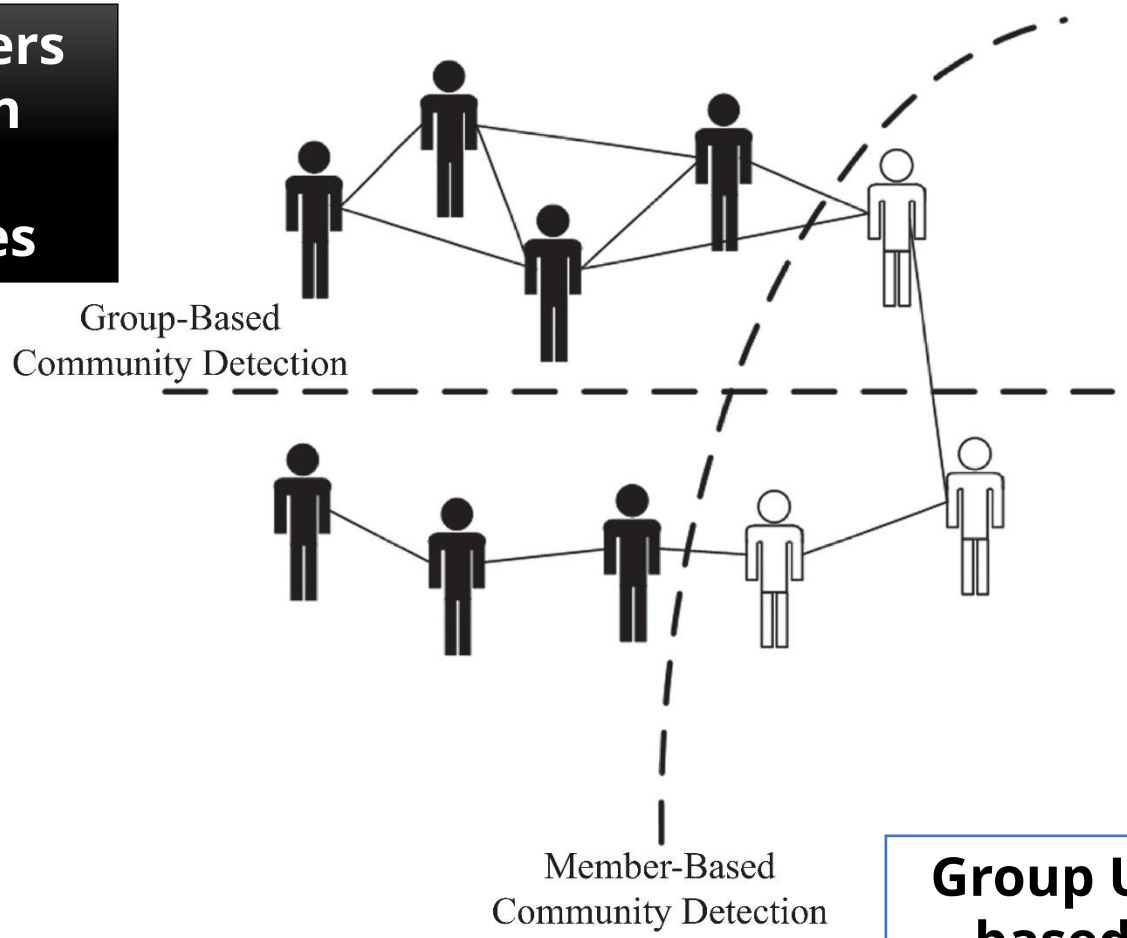
- Data is often non-linked (matrix rows)
- Clustering works on the distance or similarity matrix, e.g., k-means.
- If you use k-means with adjacency matrix rows, you are only considering the ego-centric network

Community detection

- Data is linked (a graph)
- Network data tends to be “discrete”, leading to algorithms using the graph property directly
  - k-clique, quasi-clique, or edge-betweenness

Community Detection Algorithms

Group Users based on Group attributes



Group Users based on Member attributes

Outline

- Web and Social Networks
- Node Influence and Centrality
  - How you are connected to neighbors
  - How you connect others and how you reach out to others
- Similarity among nodes
- Community Analysis
  - Member-based Community Detection
  - Group-based Community Detection

Member-based Community Detection

- Considering the characteristics of nodes to see if they are in the same community

Node Characteristics

A. Node Degree

- Nodes with same (or similar) degrees are in one community
- Example: cliques

B. Node Reachability

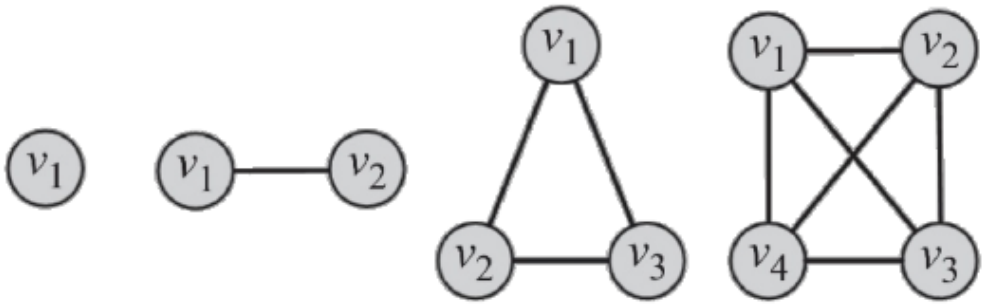
- Nodes that are close (small shortest paths) are in one community
- Example: k-cliques, k-clubs, and k-clans

C. Node Similarity

- Similar nodes are in the same community

A. Node Degree

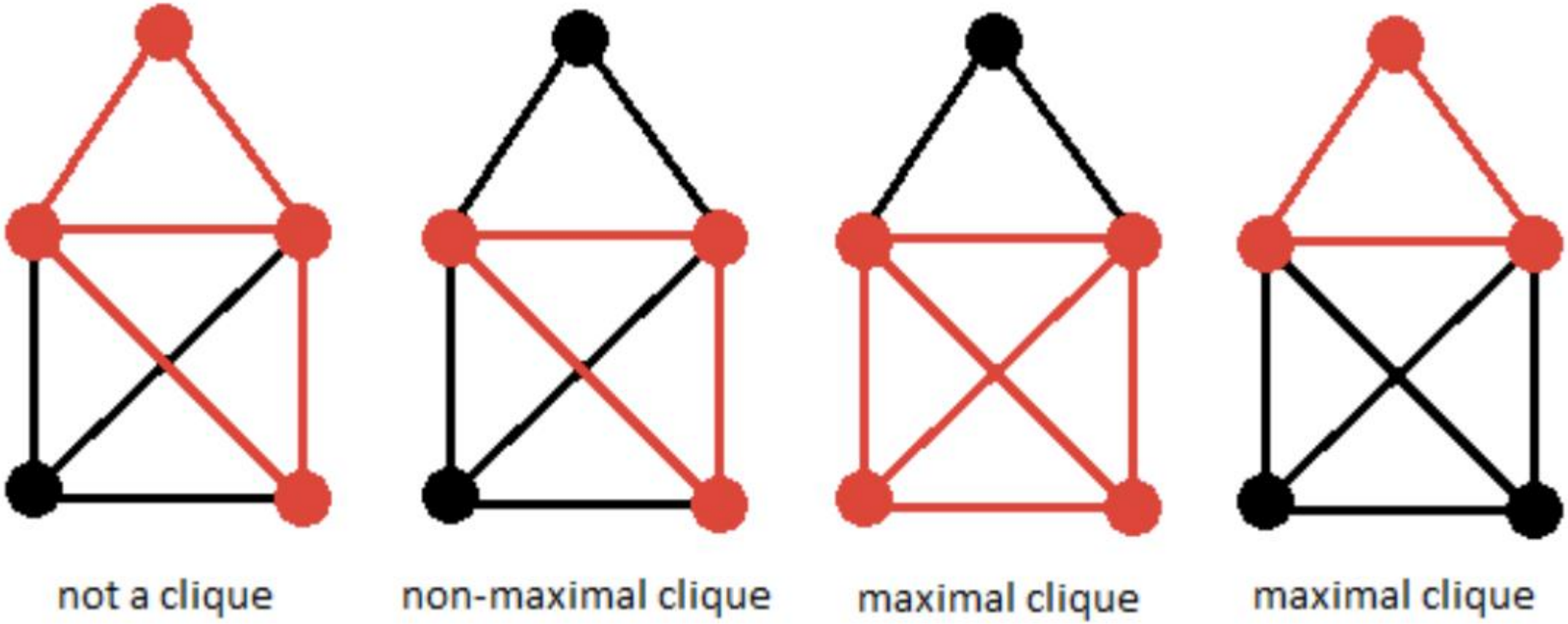
- **Clique**: a maximum complete (fully connected) subgraph in which all nodes inside the subgraph are adjacent to each other.



Find communities by searching for:

1. **The maximum clique**: The clique with the largest number of vertices, or
2. **All maximal cliques**: The cliques that are not subgraphs of a larger clique; i.e., cannot be further expanded

Maximum and Maximal Clique



Identifying Cliques

- NP-Hard = impractical for large graphs

- Relax to identifying cliques of size k or larger
  - Each node should have a degree greater or equal to $k - 1$
  - We can first prune all nodes (and edges connected to them) with degrees less than $k - 1$
    - More nodes will have degrees less than $k - 1$
    - Prune them recursively

- For large k, many nodes are pruned as a social network typically follows a power-law degree distribution (that means most of the nodes have small degree values)

Cliques as Communities?

- Cliques are rare in real networks
- A clique of 1000 nodes, has 999x1000/2 edges
- A single edge removal destroys the clique
- That is less than 0.0002% of the edges!

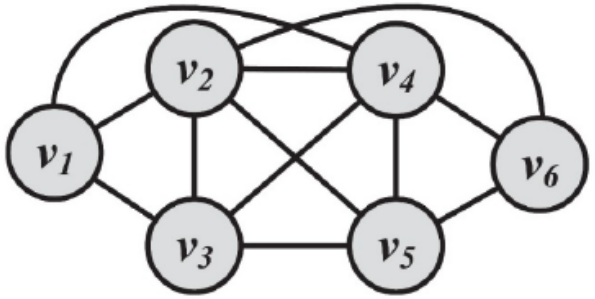
- So, can the definition of clique on node degree be relaxed?

Relaxing Cliques: k-plex

- ***k*-plex**: a set of nodes V in which we have

$$d_v \geq |V| - k, \forall v \in V$$

- d_v is the degree of v in the induced subgraph
  - Number of nodes from V that are connected to v
- A clique is a 1-plex (why?)
- Finding the maximal k-plex



1-plex : $\{v_2, v_3, v_4, v_5\}$

2-plex : $\{v_1, v_2, v_3, v_4, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$

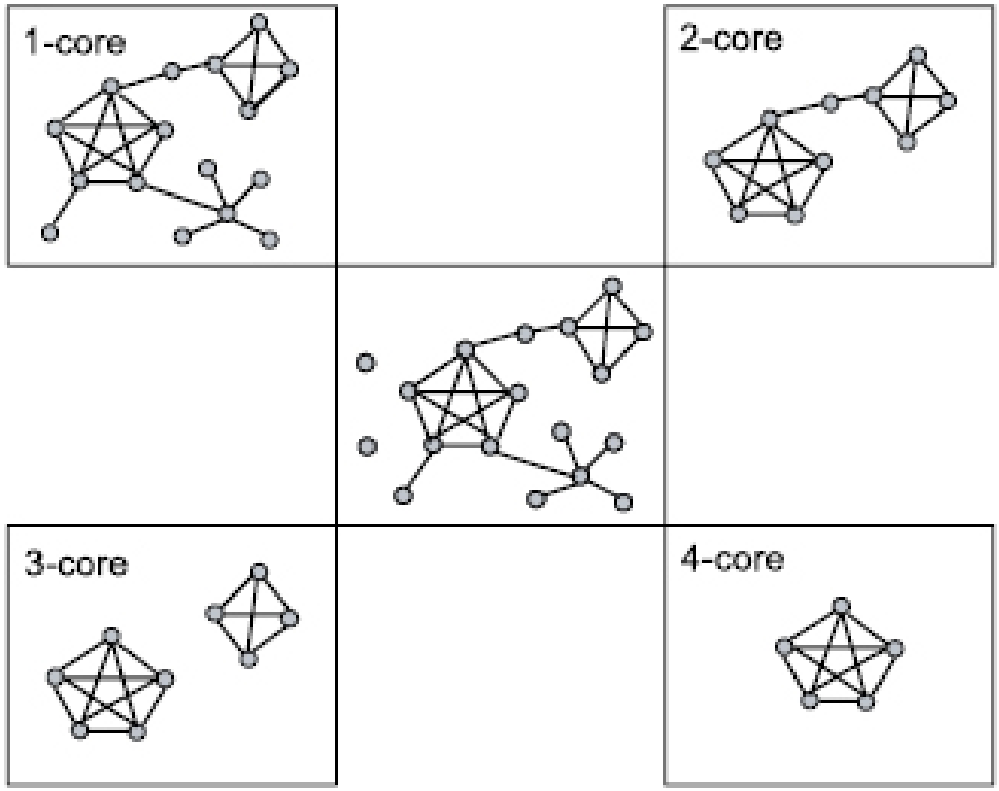
3-plex : $\{v_1, v_2, v_3, v_4, v_5, v_6\}$

Maximal k-plexes

Relaxing Cliques: k-core and k-shell

- $\boldsymbol{k}$-core: a maximal connected subgraph in which all nodes have degree at least $\boldsymbol{k}$

- $\boldsymbol{k}$-shell: nodes that are part of the k-core, but are not part of the $(k + 1)$-core.



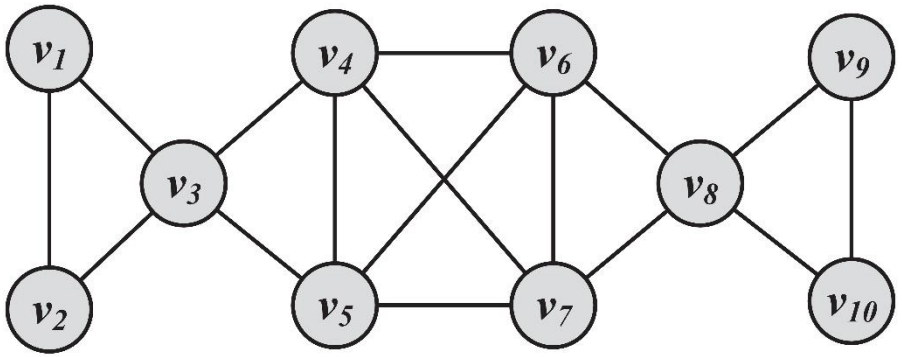
Using Cliques as Seeds of Communities

Clique Percolation Method (CPM)

- Uses cliques as seeds to find larger communities
- CPM finds overlapping communities

- **Input**
  - A parameter k, and a network
- Procedure
  - Find out all cliques of size k in the given network
  - Construct a clique graph.
    - Two cliques are adjacent if they share $k-1$ nodes
  - Each connected components in the clique graph form a community

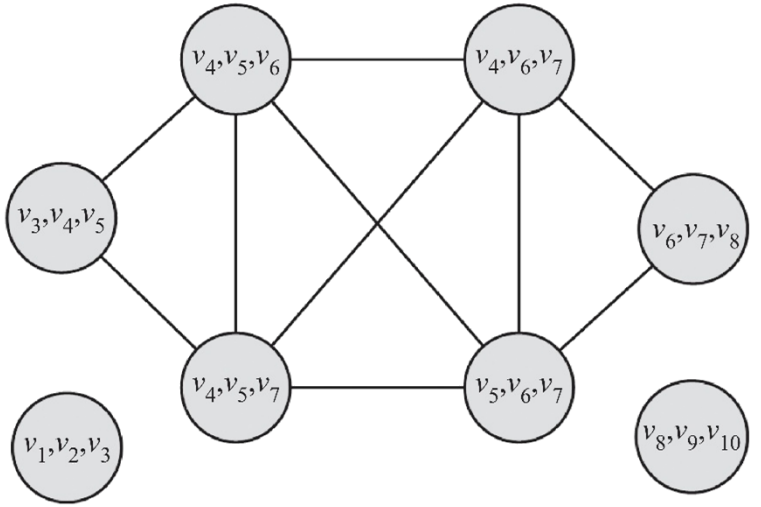
Clique Percolation Method: Example



(a) Graph

Cliques of size 3:

$\{v_1, v_2, v_3\}$, $\{v_3, v_4, v_5\}$,
$\{v_4, v_5, v_6\}$, $\{v_4, v_5, v_7\}$,
$\{v_4, v_6, v_7\}$, $\{v_5, v_6, v_7\}$,
$\{v_6, v_7, v_8\}$, $\{v_8, v_9, v_{10}\}$



(b) CPM Clique Graph

Communities:

$\{v_1, v_2, v_3\}$,
$\{v_8, v_9, v_{10}\}$,
$\{v_3, v_4, v_5, v_6, v_7, v_8\}$

B. Node Reachability

Nodes are assumed to be in the same community (two extreme cases)

1. If there is a path between them (regardless of the distance) or
2. If they are immediate neighbors of each others,

from **connected components**

to **cliques**

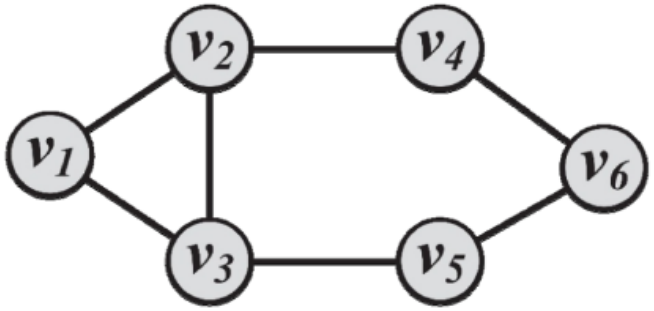
Solution: find communities that are in between cliques and connected components in terms of connectivity and have short shortest paths among their nodes.

Special Subgraphs for Node Reachability

1. k-**Clique**: a **maximal** subgraph in which the largest shortest path distance between any nodes is less than or equal to k

2. k-**Club**: follows the same definition as a k-clique
   - **Additional Constraint:** nodes on the shortest paths should be part of the subgraph

3. k-**Clan:** a k-clique where for all shortest paths within the subgraph the distance is equal or less than k.
   - All k-clans are k-cliques, but not vice versa.



2-cliques : $\{v_1, v_2, v_3, v_4, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$

2-clubs : $\{v_2, v_3, v_4, v_5, v_6\}, \{v_1, v_2, v_3, v_4\}, \{v_1, v_2, v_3, v_5\}$

2-clans : $\{v_2, v_3, v_4, v_5, v_6\}$

C. Node Similarity

- Use structural equivalence similarities (Jaccard or Cosine similarity)
- Then, apply clustering method (e.g., K-Means)

Outline

- Web and Social Networks
- Node Influence and Centrality
  - How you are connected to neighbors
  - How you connect others and how you reach out to others
- Similarity among nodes
- Community Analysis
  - Member-based Community Detection
  - Group-based Community Detection

Group-based Community Detection

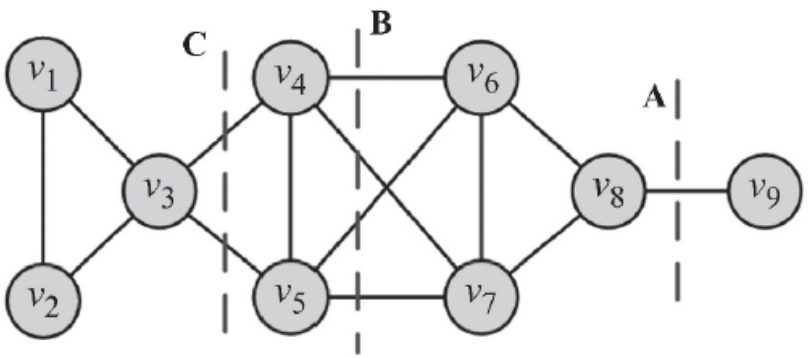
Group-based community detection: finding communities that have certain **group properties**

Group Properties:

I. **Balanced:** Spectral clustering
II. **Robust:** k-connected graphs
III. **Modular:** Modularity Maximization

I. Balanced Communities

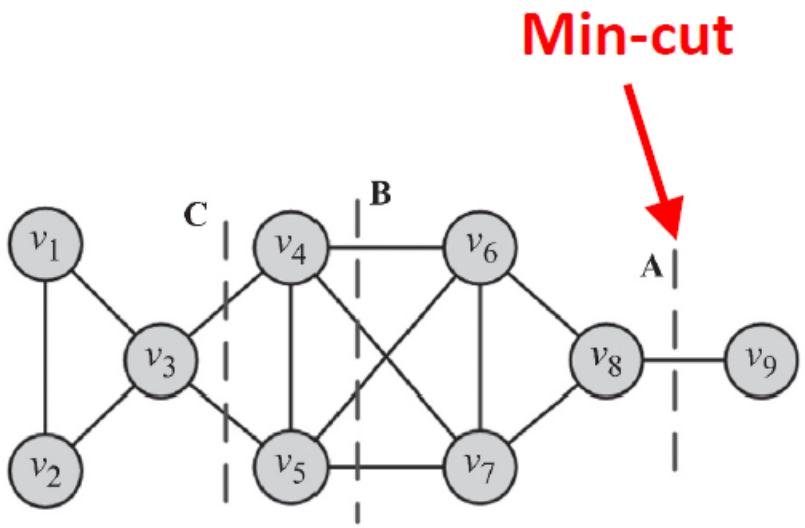
- Community detection can be thought of as *graph clustering*.
- **Graph clustering:** we *cut* the graph into several partitions and assume these partitions represent communities.
- **Cut**: partitioning of the graph into two (or more) sets
  - **The size of the cut** is the number of edges that are being cut.



A, B and C are three cuts of different sizes.

I. Balanced Communities

- **Minimum cut (min-cut) problem**: find a graph partition such that the number of edges between the two sets is minimized



Min-cuts can be computed efficiently using the max-flow algorithm.

Ratio Cut and Normalized Cut

- Min-cut often returns an *imbalanced* partition, with one set being a singleton.
- To mitigate the min-cut problem we can change the objective function to consider community size

$$\text{Ratio Cut}(P) = \frac{1}{k}\sum_{i=1}^{k}\frac{\text{cut}(P_i,\bar{P}_i)}{|P_i|}$$

$$\text{Normalized Cut}(P) = \frac{1}{k}\sum_{i=1}^{k}\frac{\text{cut}(P_i,\bar{P}_i)}{\text{vol}(P_i)}$$

Tailored algorithms exist, but Evolutionary Algorithms can be adopted!!!

- $\bar{P}_i = V - P_i$ is the complement cut set
- $cut(P_i, \bar{P}_i)$ is the size of the cut
- $vol(P_i) = \sum_{v \in P_i} d_v$

II. Robust Communities

- The goal is to find subgraphs robust enough such that removing some edges or vertices does not disconnect the subgraph

- **k-vertex connected (k-connected) graph:**
  - k is the minimum number of nodes that must be removed to disconnect the graph
- **k-edge connected:** at least k edges must be removed to disconnect the graph

- Examples:
  - Complete graph of size n: unique n-connected graph
  - A cycle: 2-connected graph

III. Modularity Maximization

- Consider a graph $G(V, E)$, where the degrees are known beforehand but edges are not
  - Consider two vertices v_i and v_j with degrees d_i & d_j.
- Expected number of edges between v_i and v_j

$$\frac{d_i d_j}{2m}$$

total number of edges

Modularity Maximization

- We assume that real-world networks should be far from such a random network.
- Therefore, the more distant they are from this randomly generated network, the more structural they are.
- Modularity defines this distance and modularity maximization tries to maximize this distance.

Tailored algorithms exist, but Evolutionary Algorithms can be adopted!!!

Normalized Modularity as Objective Function

Consider a partitioning of the data $P = (P_1, P_2, P_3, \ldots, P_k)$

For partition P_x, this distance can be defined as

$$\sum_{i,j \in P_x} A_{ij} - \frac{d_i d_j}{2m}$$

This distance can be generalized for a partitioning P

$$\sum_{x=1}^{k} \sum_{i,j \in P_x} A_{ij} - \frac{d_i d_j}{2m}$$

The normalized version of this distance is defined as Modularity

$$Q = \frac{1}{2m} \sum_{x=1}^{k} \sum_{i,j \in P_x} A_{ij} - \frac{d_i d_j}{2m}$$

Nowadays …

- The Clique Percolation Method is the most used method for finding overlapping communities
- Modularity Maximization is the most used method for finding disjoint communities

References

- R. Zafarani, M. A. Abbasi, and H. Liu, Social Media Mining: An Introduction, Cambridge University Press, 2014 [Chapter 3]. URL: http://socialmediamining.info/

- R. Zafarani, M. A. Abbasi, and H. Liu, Social Media Mining: An Introduction, Cambridge University Press, 2014 [Chapter 6]. URL: http://socialmediamining.info/