Network Analysis:

The Hidden Structures behind the Webs We Weave

17-338 / 17-668

Power and Centrality in Social Networks

Tuesday, September 23, 2025

Patrick Park

Carnegie Mellon University
School of Computer Science

S3D
Software and Societal Systems Department

“Central” actors are powerful

Relational view of the individual

So far, we covered dyads and triads as basic building blocks of networks

The relational view has been the overarching theme

- relations between two people
- relations among three people

The same relational view to study the individual

This approach sharply contrasts with widely used approaches

- Surveys ask individuals about their attributes, beliefs, and actions
- Majority of statistical models assume that individuals are like disconnected atoms
    - Independent, identically distributed variables assumption (IID).

Which node is the most “prominent”?

In contrast, network analysis views an individual in relation to their network neighbors

- **Opinions** are influenced by network neighbors
- Individual’s **status** and importance comes from their position in the network
- Individuals can leverage their position to **influence opinions, selectively spread or block information, and control opportunities**
- **Power** that an individual can wield is fundamentally relational

A network position of power and prominence is a “central” position

- Power and prominence are multifaceted
- Therefore, “central” can reflect different aspects of power
    - “Centrality” can be defined and measured in different ways

So many ways to define “centrality”

Centrality

Degree

`degree_centrality` (G)	Compute the degree centrality for nodes.
`in_degree_centrality` (G)	Compute the in-degree centrality for nodes.
`out_degree_centrality` (G)	Compute the out-degree centrality for nodes.

Eigenvector

`eigenvector_centrality` (G[, max_iter, tol, ...])	Compute the eigenvector centrality for the graph `G`.
`eigenvector_centrality_numpy` (G[, weight, ...])	Compute the eigenvector centrality for the graph G.
`katz_centrality` (G[, alpha, beta, max_iter, ...])	Compute the Katz centrality for the nodes of the graph G.
`katz_centrality_numpy` (G[, alpha, beta, ...])	Compute the Katz centrality for the graph G.

Closeness

`closeness_centrality` (G[, u, distance, ...])	Compute closeness centrality for nodes.
`incremental_closeness_centrality` (G, edge[, ...])	Incremental closeness centrality for nodes.

Current Flow Closeness

`current_flow_closeness_centrality` (G[, ...])	Compute current-flow closeness centrality for nodes.
`information_centrality` (G[, weight, dtype, ...])	Compute current-flow closeness centrality for nodes.

(Shortest Path) Betweenness

`betweenness_centrality` (G[, k, normalized, ...])	Compute the shortest-path betweenness centrality for nodes.
`betweenness_centrality_subset` (G, sources, ...)	Compute betweenness centrality for a subset of nodes.
`edge_betweenness_centrality` (G[, k, ...])	Compute betweenness centrality for edges.
`edge_betweenness_centrality_subset` (G, ... [, ...])	Compute betweenness centrality for edges for a subset of nodes.

Current Flow Betweenness

`current_flow_betweenness_centrality` (G[, ...])	Compute current-flow betweenness centrality for nodes.
`edge_current_flow_betweenness_centrality` (G)	Compute current-flow betweenness centrality for edges.
`approximate_current_flow_betweenness_centrality` (G)	Compute the approximate current-flow betweenness centrality for nodes.
`current_flow_betweenness_centrality_subset` (G, ...)	Compute current-flow betweenness centrality for subsets of nodes.
`edge_current_flow_betweenness_centrality_subset` (G, ...)	Compute current-flow betweenness centrality for edges using subsets of nodes.

Communicability Betweenness

`communicability_betweenness_centrality` (G)	Returns subgraph communicability for all pairs of nodes in G.

Group Centrality

`group_betweenness_centrality` (G, C[, ...])	Compute the group betweenness centrality for a group of nodes.
`group_closeness_centrality` (G, S[, weight])	Compute the group closeness centrality for a group of nodes.
`group_degree_centrality` (G, S)	Compute the group degree centrality for a group of nodes.
`group_in_degree_centrality` (G, S)	Compute the group in-degree centrality for a group of nodes.
`group_out_degree_centrality` (G, S)	Compute the group out-degree centrality for a group of nodes.
`prominent_group` (G, k[, weight, C, ...])	Find the prominent group of size k in graph G.

Load

`load_centrality` (G[, v, cutoff, normalized, ...])	Compute load centrality for nodes.
`edge_load_centrality` (G[, cutoff])	Compute edge load.

Subgraph

`subgraph_centrality` (G)	Returns subgraph centrality for each node in G.
`subgraph_centrality_exp` (G)	Returns the subgraph centrality for each node of G.
`estrada_index` (G)	Returns the Estrada index of a the graph G.

Harmonic Centrality

`harmonic_centrality` (G[, nbunch, distance, ...])	Compute harmonic centrality for nodes.

Dispersion

`dispersion` (G[, u, v, normalized, alpha, b, c])	Calculate dispersion between `u` and `v` in `G`.

Reaching

`local_reaching_centrality` (G, v[, paths, ...])	Returns the local reaching centrality of a node in a directed graph.
`global_reaching_centrality` (G[, weight, ...])	Returns the global reaching centrality of a directed graph.

Percolation

`percolation_centrality` (G[, attribute, ...])	Compute the percolation centrality for nodes.

Second Order Centrality

`second_order_centrality` (G)	Compute the second order centrality for nodes of G.

Trophic

`trophic_levels` (G[, weight])
`trophic_differences` (G[, weight])
`trophic_incoherence_parameter` (G[, weigh

VoteRank

`voterank` (G[, number_of_nodes])	Select a

Laplacian

`laplacian_centrality` (G[, normalized, ...])

...

So many ways to define “centrality”

We will cover the four most widely used centrality measures in network analysis

- Degree centrality
- Closeness centrality
- Betweenness centrality
- Eigenvector centrality

Degree Centrality

Insight: **Central nodes have many connections**

- Access to information
    - Gate keepers

Degree Centrality

Insight: **Central nodes have many connections**

- Access to information
    - Gate keepers

- Concentration of attention (attention is scarce)
    - Opinion leaders

Degree Centrality

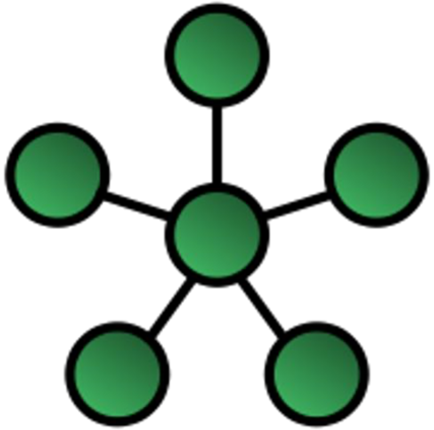
Insight: **Central nodes have many connections**

- Access to information
    - Gate keepers

- Concentration of attention (attention is scarce)
    - Opinion leaders

- Signal of “quality” under high uncertainty
    - Influencers
    - ‘There must be some reason why he has so many connections.’

Degree Centrality



Insight: Central nodes have many connections

The number of ties of a node, **normalized by network size** (g-1).

$$C_D(i) = \frac{k_i}{g - 1}$$

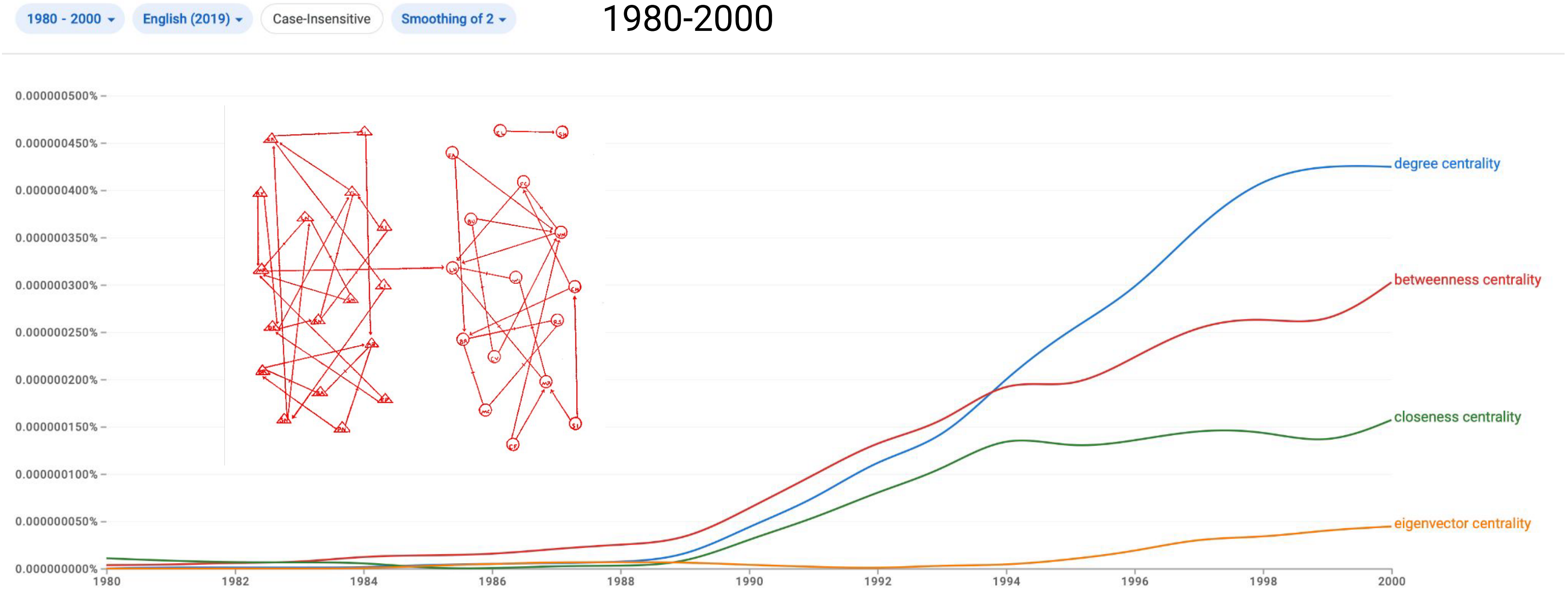
Normalization allows for comparison across different networks

One of the most basic and fundamental quantities in network science

Degree Centrality

Degree centrality was, by far, the most popular measure up to the early 2000s

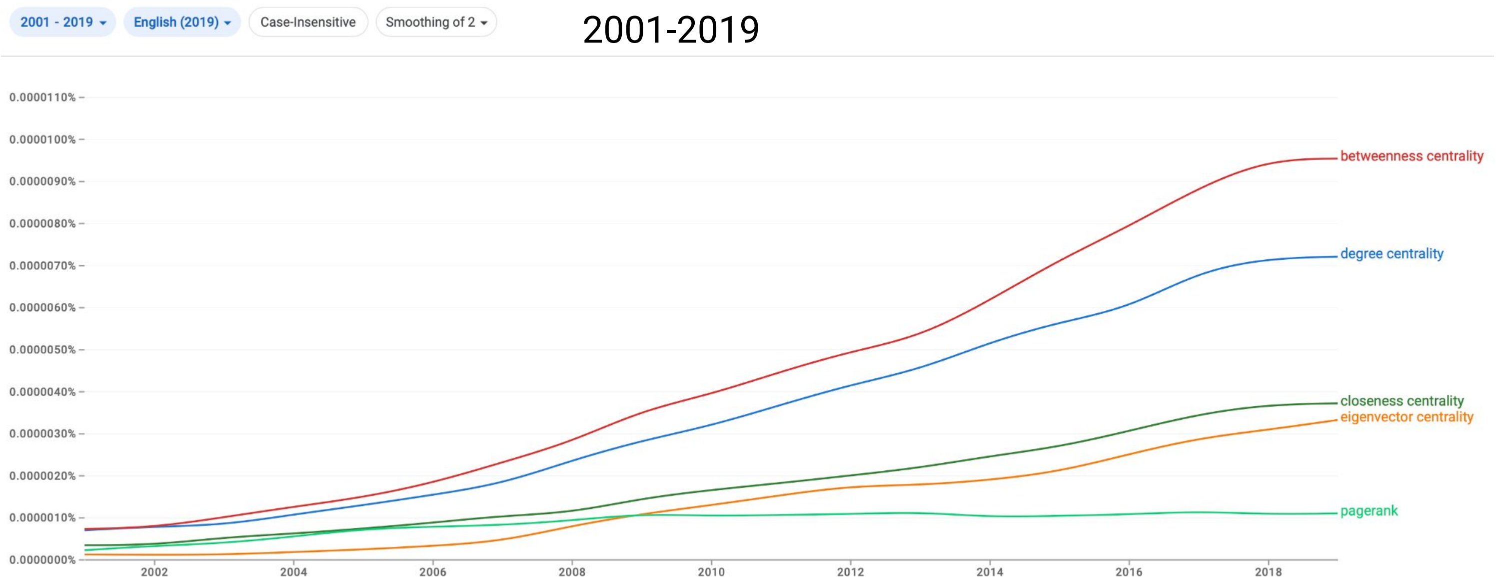
1980-2000



Degree Centrality

However, its prominence arguably diminished in the past two decades

Why?

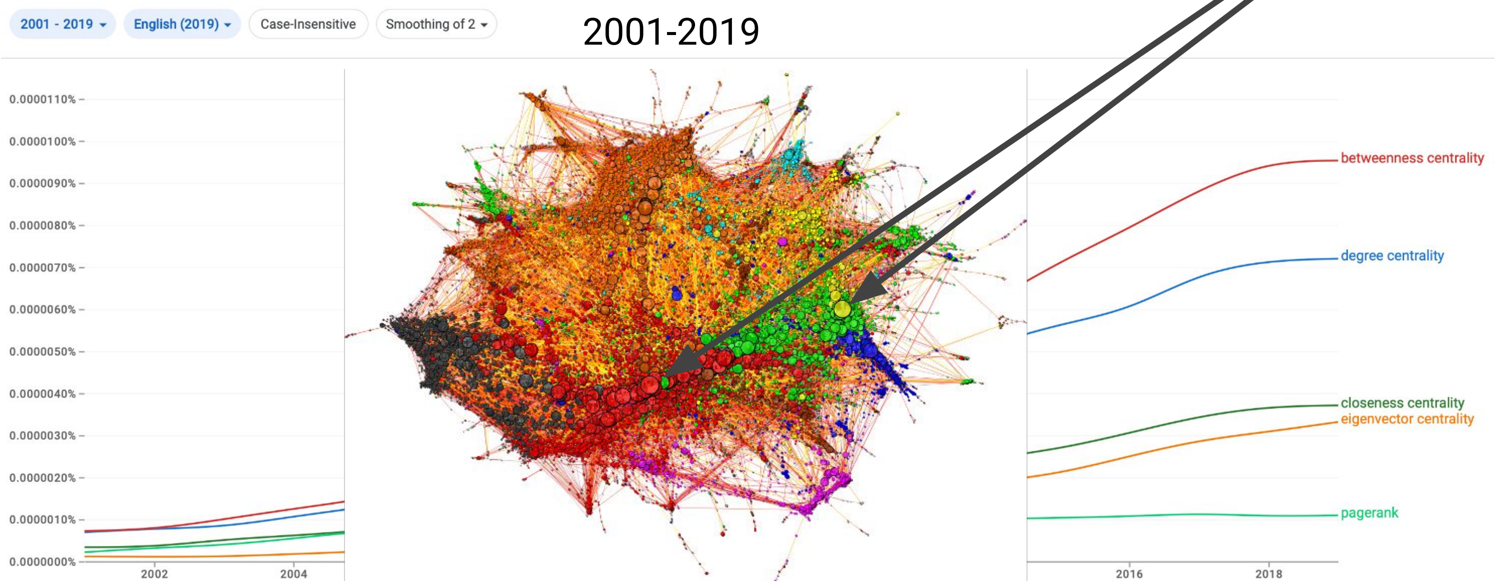


Degree Centrality

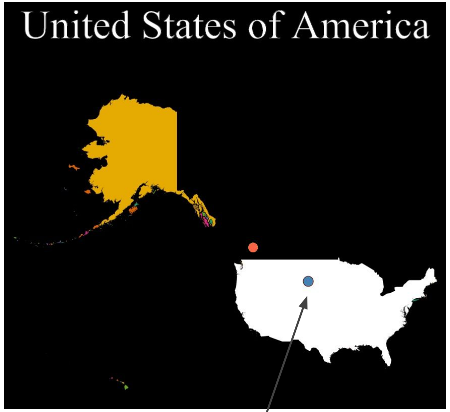
However, its prominence arguably diminished in the past two decades

Why?

In large-scale networks (post-2000s), degree centrality becomes a “local” measure



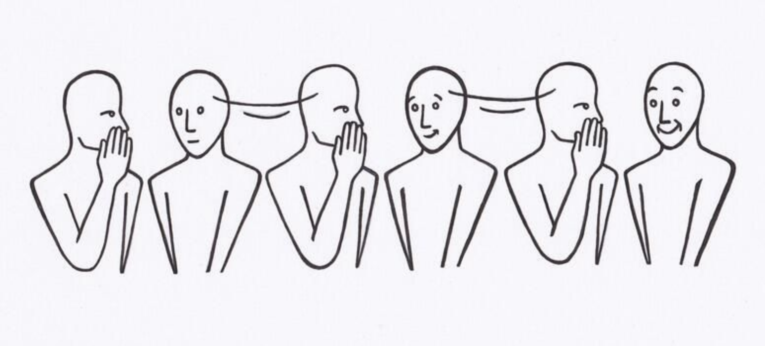
Closeness Centrality



Centroid of the U.S.

Insight: The position that minimizes the distance to all other positions is the most central

Closeness Centrality



Telephone game

Insight: The position that minimizes the distance to all other positions is the most central

Application:

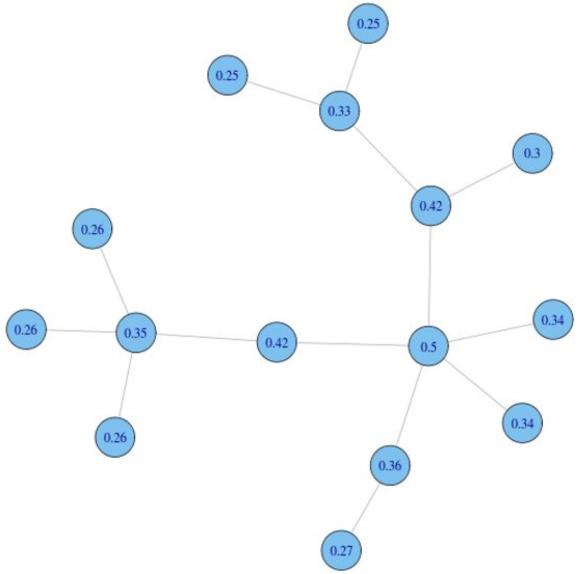
- Systems where traversing the network is costly
- Information attrition rate is high

→ The node closest to all other nodes can obtain more accurate messages

→ Faster access

→ A source of power

Closeness Centrality



Central nodes have short paths to other nodes

Calculated as the reciprocal of the sum of pairwise path distances

$$C_c(i) = [\sum_{j=1}^{g} d(i,j)]^{-1}$$

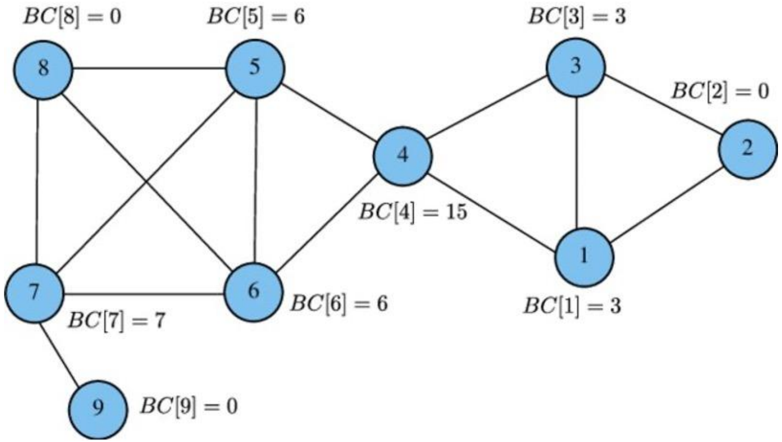
Normalized by size of network

$$C_c(i) = (g-1)[\sum_{j=1}^{g} d(i,j)]^{-1}$$

A global measure that uses information from the entire network

Betweenness Centrality

Number of shortest paths node is on



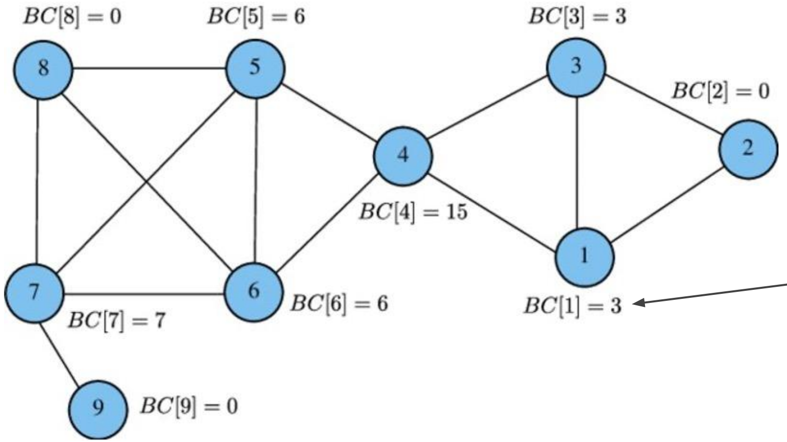
Node is central if it sits on many shortest paths

Nodes positioned at Information bottlenecks are central

Those nodes have more control over the distribution of information and other resources

Betweenness Centrality

Number of shortest paths node is on



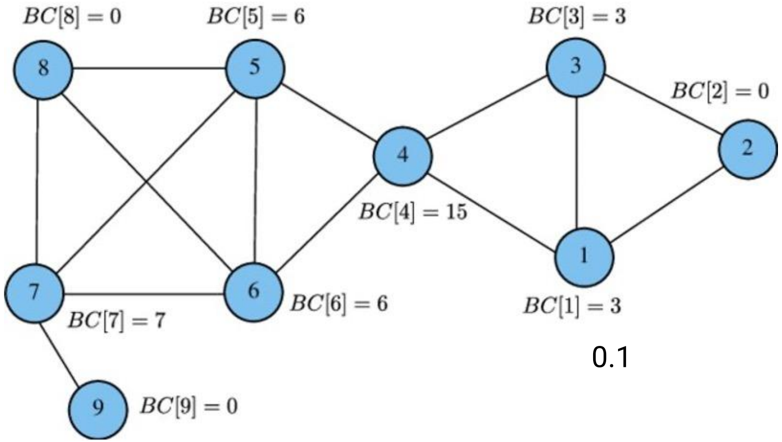
Node *i*'s betweenness is the sum of the probabilities that *i* is on the shortest paths of all node pairs in the network

$$C_B(i) = \sum_{j<k} PL(i,j,k)/PL(j,k)$$

PL(i,j,k) → Number of shortest paths involving i

PL(j,k) → Number of shortest paths between j and k

Betweenness Centrality



0.1

$$C_B(i) = \sum_{j<k} PL(i,j,k)/PL(j,k)$$

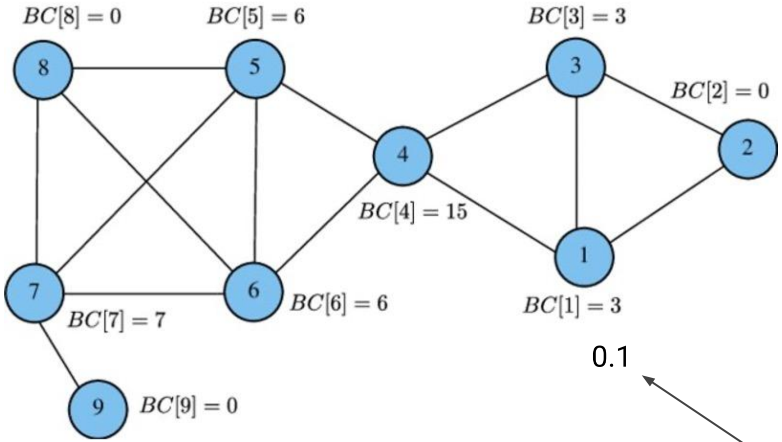
Maximum probability *PL(i,j,k)* / *PL(j,k)* = 1 is when *i* sits on every shortest path between *j* and *k*.

Then, the theoretical maximum *C(i)* is when *i* sits on every shortest path for every pair of nodes (excluding *i*)

Q: What is the number of node pairs (dyads) excluding *i* in a graph with *g* nodes?

Betweenness Centrality

$$C_B(i) = \sum_{j<k} PL(i,j,k)/PL(j,k)$$



0.1

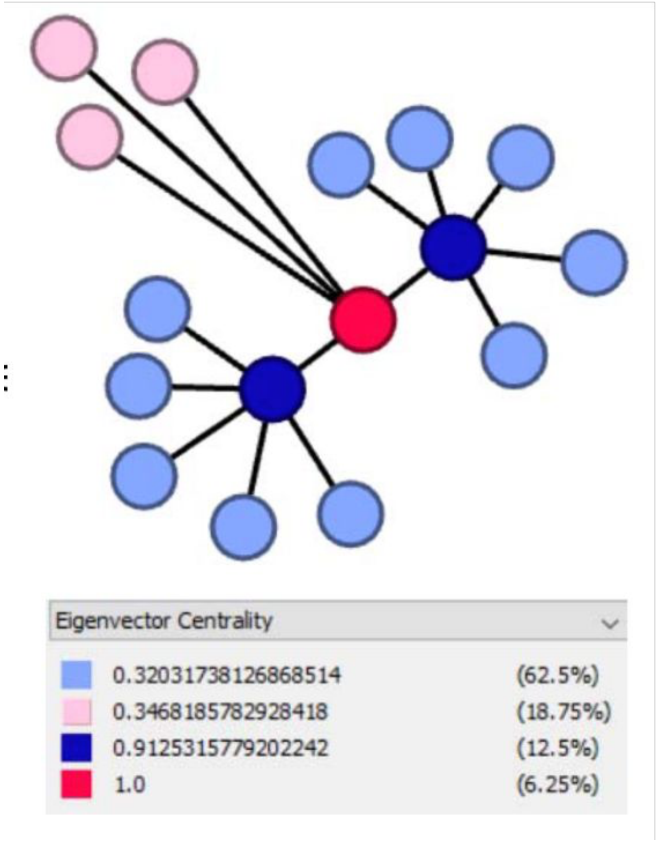
Maximum probability *PL(i,j,k) / PL(j,k)* = 1 is when *i* sits on every shortest path between *j* and *k*.

Then, the theoretical maximum *C(i)* is when *i* sits on every shortest path for every pair of nodes (excluding *i*)

→ Normalize by the total number of node pairs excluding *i* → (*g*-1)(*g*-2)/2

$$C_B{}'(i) = \frac{C_B(i)}{\left[\frac{(g-1)(g-2)}{2}\right]}$$

Eigenvector Centrality



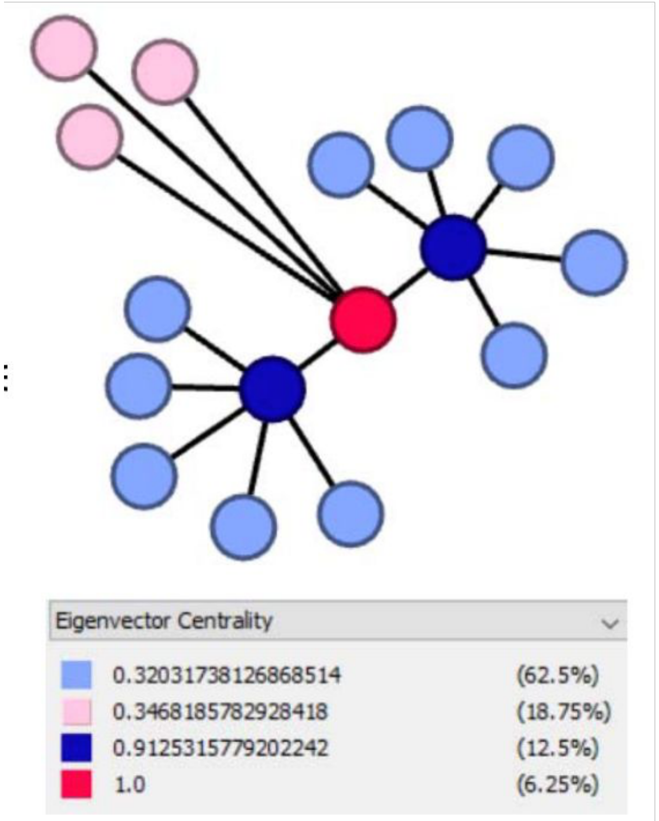
Who is more central?

→ Someone who knows five (well connected) celebrities vs. someone who knows five ordinary people

Eigenvector centrality quantifies this insight

→One's Eigenvector centrality is determined by the neighbors' eigenvector centrality, which in turn are determined by their neighbors' eigenvector centrality …

Eigenvector Centrality



Who is more central?

→ Someone who knows five (well connected) celebrities vs. someone who knows five ordinary people

Eigenvector centrality quantifies this intuition

→One's Eigenvector centrality is determined by the neighbors' eigenvector centrality, which in turn are determined by their neighbors' eigenvector centrality …

Red node has two (purple) friends who are themselves highly connected

→ Highest Eigenvector centrality

Pink nodes have same degree centrality as the sky blue nodes, but they are connected to the red node (highest Eigenvector centrality)

→ Eigenvector centrality: pink > sky blue

Eigenvector Centrality

Given an adjacency matrix ***A***, eigenvector centrality of node *i* is:

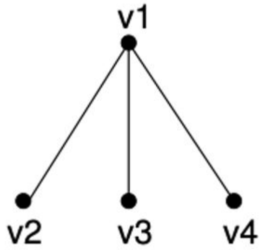
- *i*'s weighted degree where each of *i*'s edge is weighted by the degree of neighbor, *j*

$$\lambda C_E(i) = \sum_j A_{ij}\, C_E(j)$$

Given adjacency matrix ***A***, compute the eigenvector ***x*** corresponding to the principal eigenvalue λ^* of ***A***

Each element in ***x*** is the eigenvector centrality value of the corresponding node

Eigenvector Centrality



$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

For adjacency matrix *A*:

Eigenvector centrality of the nodes are

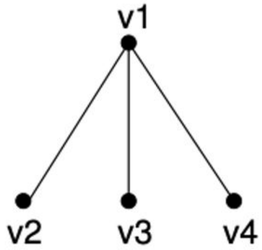
$$\boldsymbol{x_1} = \frac{1}{\lambda}(\boldsymbol{x_2} + \boldsymbol{x_3} + \boldsymbol{x_4})$$

$$\boldsymbol{x_2} = \frac{1}{\lambda}(\boldsymbol{x_1})$$

$$\boldsymbol{x_3} = \frac{1}{\lambda}(\boldsymbol{x_1})$$

$$\boldsymbol{x_4} = \frac{1}{\lambda}(\boldsymbol{x_1})$$

Eigenvector Centrality



$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\lambda \boldsymbol{x_1} = (\boldsymbol{x_2} + \boldsymbol{x_3} + \boldsymbol{x_4})$$

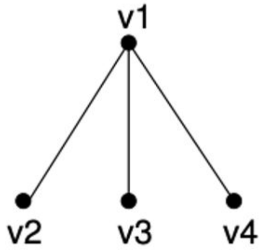
$$\lambda \boldsymbol{x_2} = \boldsymbol{x_1}$$

$$\lambda \boldsymbol{x_3} = \boldsymbol{x_1}$$

$$\lambda \boldsymbol{x_4} = \boldsymbol{x_1}$$

$$\lambda \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \boldsymbol{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \boldsymbol{x_4} \end{bmatrix}$$

Eigenvector Centrality

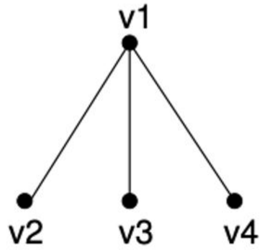


$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\lambda \boldsymbol{x_1} = (\boldsymbol{x_2} + \boldsymbol{x_3} + \boldsymbol{x_4})$$
$$\lambda \boldsymbol{x_2} = \boldsymbol{x_1}$$
$$\lambda \boldsymbol{x_3} = \boldsymbol{x_1}$$
$$\lambda \boldsymbol{x_4} = \boldsymbol{x_1}$$

$$\lambda \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \boldsymbol{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \boldsymbol{x_4} \end{bmatrix} \longrightarrow \lambda \mathrm{x} = \mathrm{Ax}$$

Eigenvector Centrality



$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\lambda \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \boldsymbol{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \boldsymbol{x_4} \end{bmatrix} \longrightarrow \lambda \mathrm{x} = \mathrm{Ax}$$

$$\mathbf{Ax} - \lambda\mathbf{x} = (\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$$

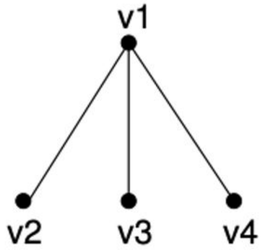
$$\mathbf{A} - \lambda\mathbf{I} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & \lambda \end{bmatrix} = \begin{bmatrix} -\lambda & 1 & 1 & 1 \\ 1 & -\lambda & 0 & 0 \\ 1 & 0 & -\lambda & 0 \\ 1 & 0 & 0 & -\lambda \end{bmatrix}$$

For matrix ***A***, characteristic equation in λ is:

$$\begin{bmatrix} -\lambda & 1 & 1 & 1 \\ 1 & -\lambda & 0 & 0 \\ 1 & 0 & -\lambda & 0 \\ 1 & 0 & 0 & -\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Eigenvector Centrality

$$\lambda \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \longrightarrow \lambda x = Ax$$



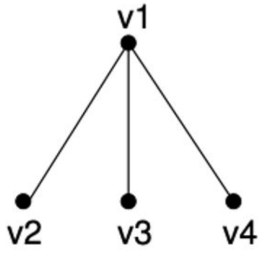
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Solution to the characteristic equation:
→ determinant of $\mathbf{A} - \lambda\mathbf{I}$ is **0**

$$\lambda^4 - 3\lambda^2 = \lambda^2(\lambda^2 - 3) = 0$$

Eigenvalues are -√3, 0, 0, √3
Principal eigenvalue **λ* = √3**

Eigenvector Centrality



$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{aligned} \lambda x_1 &= (x_2 + x_3 + x_4) \\ \lambda x_2 &= x_1 \\ \lambda x_3 &= x_1 \\ \lambda x_4 &= x_1 \end{aligned} \longrightarrow \begin{aligned} \sqrt{3}x_1 &= x_2 + x_3 + x_4 \\ \sqrt{3}x_2 &= x_1 \\ \sqrt{3}x_3 &= x_1 \\ \sqrt{3}x_4 &= x_1 \end{aligned}$$

$$\mathbf{x} = \begin{bmatrix} \sqrt{3} \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

v1 has highest eigenvector centrality, √3

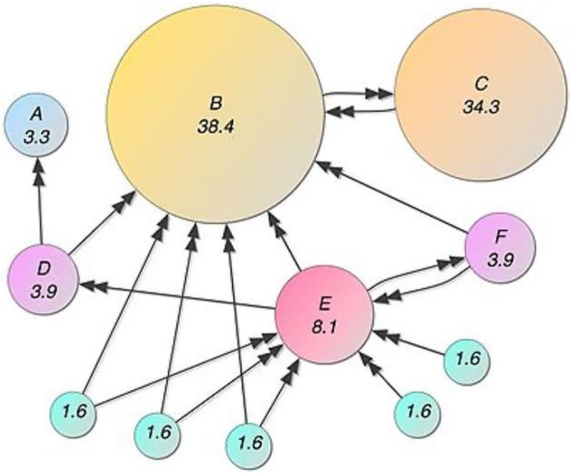
while v2 = v3 = v4 = 1

$$\lambda \mathrm{x} = \mathrm{Ax} \longrightarrow \sqrt{3} \begin{bmatrix} \sqrt{3} \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{3} \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Eigenvector Centrality

Google's original PageRank algorithm ranks web pages using a variant of eigenvector centrality

- A node's PageRank is a function of the PageRank of its in-neighbors
- Normalize the eigenvector centrality of the in-neighbor nodes with their outdegree



lambda

$$R(w) = k \sum_{v \in C_w} \frac{R(v)}{L_v}$$

PageRank of *w*

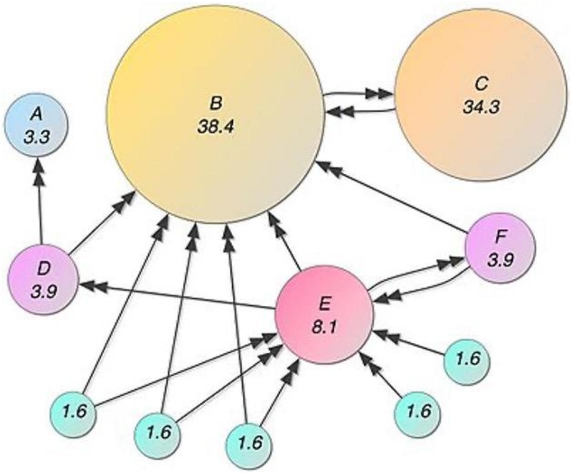
Node *w*'s in-link neighbors

Node *v*'s outdegree

Eigenvector Centrality

Google's original PageRank algorithm ranks web pages using a variant of eigenvector centrality

- A node's PageRank is a function of the PageRank of its in-link neighbors
- **Normalize the eigenvector centrality** of the in-link neighbors with their outdegrees
  - **Why?**



lambda

$$R(w) = k \sum_{v \in C_w} \frac{R(v)}{L_v}$$

PageRank of *w*

Node *w*'s in-link neighbors

Node *v*'s outdegree

Similarities among Centrality Measures

Average correlations between centrality measures (N=58).

Pearson correlation

		1	2	3	4	5	6	7	8	9	10	11	
1	Indegree												
2	Outdegree	0.3											
3	Degree	0.78	0.71										
4	Between	0.62	0.54	0.7									
5	S-Between	0.69	0.5	0.85	0.67								
6	Closeness-In	0.55	0.16	0.45	0.37	0.31							
7	Closeness-Out	0.18	0.81	0.56	0.39	0.38	0.02						
8	S-Closeness	0.4	0.64	0.66	0.37	0.44	0.42	0.65					
9	Integration	0.7	0.26	0.58	0.5	0.41	0.9	0.15	0.51				
10	Radiality	0.21	0.86	0.61	0.44	0.41	0.06	0.98	0.67	0.19			
11	S-Int/Rad	0.45	0.7	0.73	0.43	0.5	0.44	0.69	0.99	0.54	0.72		
12	Eigenvector	0.71	0.69	0.92	0.64	0.72	0.44	0.55	0.63	0.57	0.59	0.71	
Average		0.51	0.56	0.69	0.52	0.53	0.37	0.49	0.58	0.48	0.52	0.63	0.65
Standard Deviation		0.21	0.23	0.14	0.16	0.14	0.27	0.22	0.25	0.28	0.17	0.12	0.12

** **S**-Between: Betweenness centrality on **s**ymmetrized network

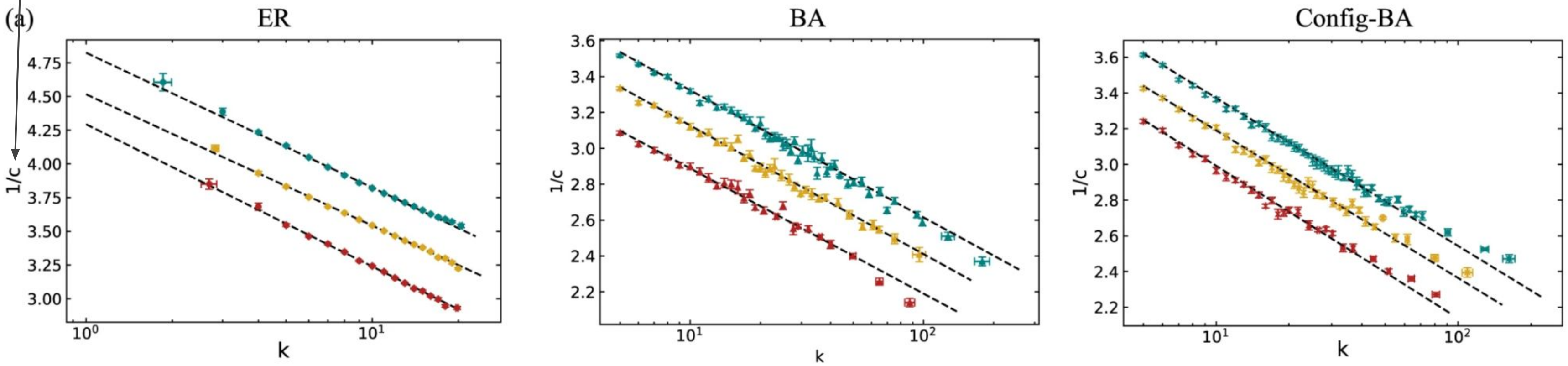
Valente et al. 2008

Similarities among Centrality Measures

Sometimes, the relationship is non-linear → lower linear correlation (Pearson)

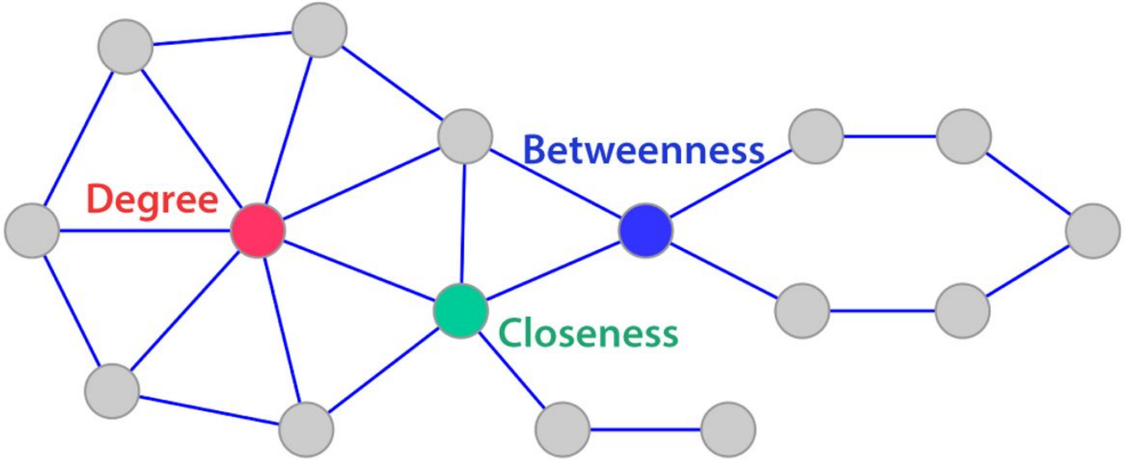
- Example: Closeness centrality and the log of degree centrality are highly correlated (redundant)
- If such a strong relationship, why do we care about computationally expensive closeness centrality?

Closeness centrality (inverse)



Evans and Chen 2022

Similar, but also different



Degree	ClosenessCentrality	BetweennessCentrality
7	0.45454545	0.29047619
5	0.51724138	0.42380952
4	0.48387097	0.4952381

Different centrality measures capture different intuitions

In a graph, the node with highest degree is not necessarily the node with the highest betweenness or closeness

Summary

Centrality as a general term for measuring a node's position of "prominence" in the network

Degree: sheer connections

Closeness: shortest path distance

Betweenness: node's presence in shortest paths

Eigenvector: degree weighted by the degree of the neighbors

These measures are highly correlated, but conceptually distinct