Modelling and Analysis of Complex Networks

— Semester 3, Master of Data Science —

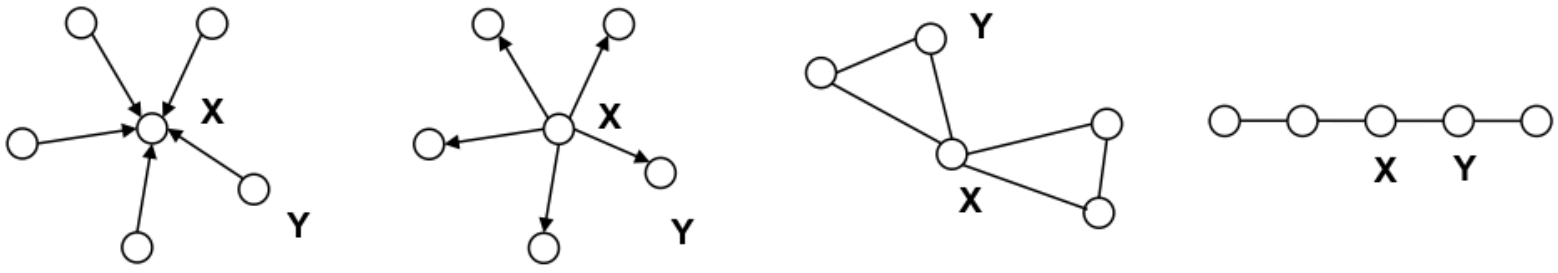
Jun Pang
University of Luxembourg

Centrality Measures

Node Centrality

Centrality is a node's measure w.r.t. others

- A central node is important and/or powerful
- A central node has an influential position in the network
- A central node has an advantageous position in the network

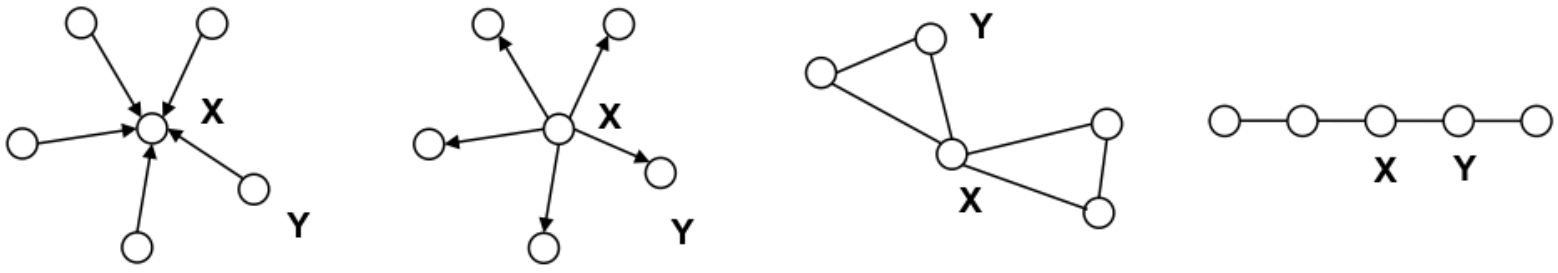


- Poor terminology: nothing to do with being central in general
- Centralities can be used as node features for ML on graphs

Node Centrality

Centrality is a node's measure w.r.t. others

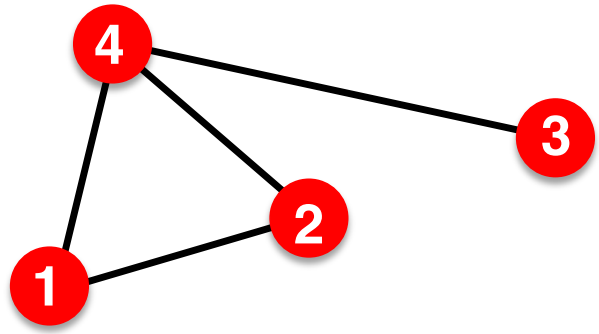
- A central node is important and/or powerful
- A central node has an influential position in the network
- A central node has an advantageous position in the network



- Poor terminology: nothing to do with being central in general
- Centralities can be used as node features for ML on graphs

Degree Centrality

Number of connections of a node



$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$k_i = \sum_{j=1}^{N} A_{ij}$$

$$k_j = \sum_{i=1}^{N} A_{ij}$$

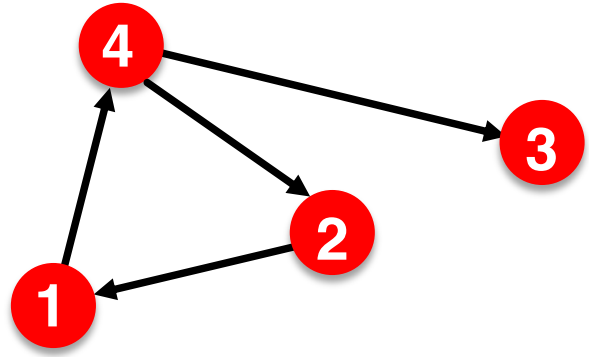
$$A_{ij} = A_{ji}$$

$$A_{ii} = 0$$

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i = \frac{1}{2}\sum_{ij} A_{ij}$$

Degree Centrality

Number of connections of a node



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

$$k_i^{out} = \sum_{j=1}^{N} A_{ij}$$

$$k_j^{in} = \sum_{i=1}^{N} A_{ij}$$

$$A_{ij} \neq A_{ji}$$
$$A_{ii} = 0$$

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{j=1}^{N} k_j^{out} = \sum_{i,j} A_{ij}$$

Degree Centrality

- Degree: how many neighbours
- Often enough to find important nodes (e.g., main characters of a series talk with the more people)
- But not always (e.g., Facebook users with the most friends are spam; Wikipedia pages with the most links are simple lists of references.)

Degree Centrality

- Degree: how many neighbours
- Often enough to find important nodes (e.g., main characters of a series talk with the more people)
- But not always (e.g., Facebook users with the most friends are spam; Wikipedia pages with the most links are simple lists of references.)

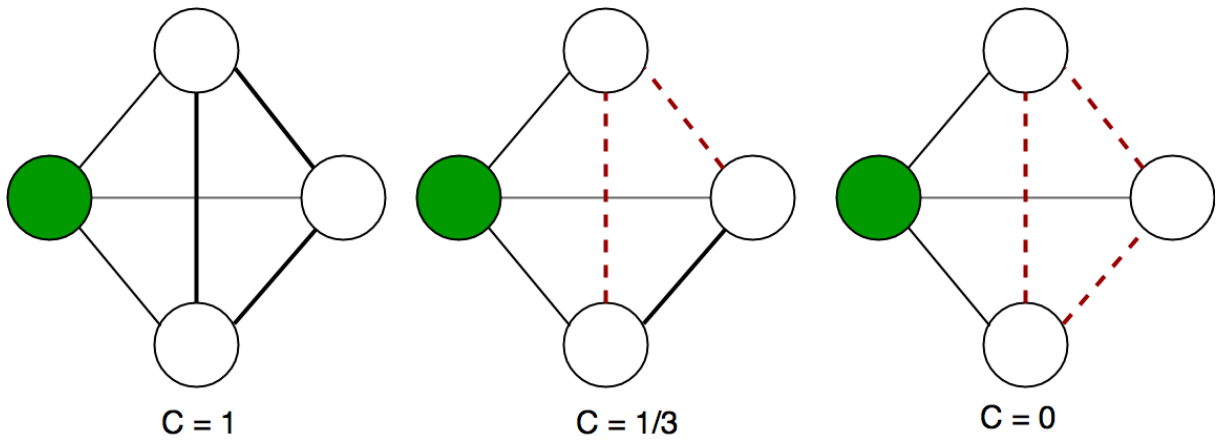
Degree Clustering Coefficient

Clustering coefficient: density of neighbourhood

- What portion of node i's neighbours are connected?
- Node i with degree k_i

$$C_i = \frac{2e_i}{k_i(k_i - 1)}$$

where e_i is the number of edges between node i's neighbours.



Degree Clustering Coefficient

Clustering coefficient: density of neighbourhood

- Be careful!
  - Degree 2: value 0 or 1
  - Degree 100: not 0 or 1
  - Ranking clustering coefficients is not meaningful
- Can be used as a proxy for "communities" belonging:
  - High value: node belongs to a single group
  - Low value: node belongs to several groups

Degree Clustering Coefficient

Clustering coefficient: density of neighbourhood

- Be careful!
  - Degree 2: value 0 or 1
  - Degree 100: not 0 or 1
  - Ranking clustering coefficients is not meaningful
- Can be used as a proxy for "communities" belonging:
  - High value: node belongs to a single group
  - Low value: node belongs to several groups

Link Clustering Coefficient

Edge clustering C^e of an edge (i, j) is the fraction of the neighbours of at least one of the two nodes which are neighbours of both of them, i.e.

$$C^e(i,j) = \frac{| N_i \cap N_j |}{| N_i \cup N_j | - 2}$$

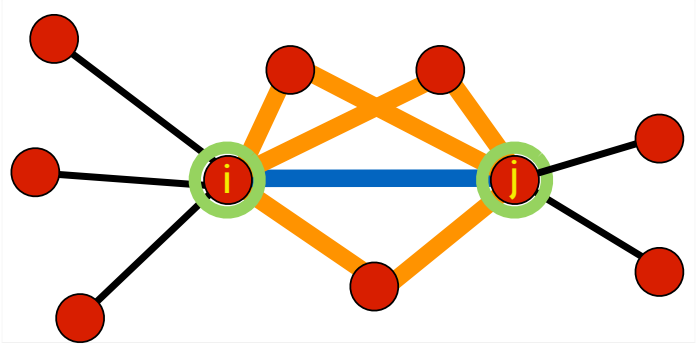
- Link property
- Fraction of common neighbours of a connected pair

Link Clustering Coefficient

Edge clustering C^e of an edge (i, j) is the fraction of the neighbours of at least one of the two nodes which are neighbours of both of them, i.e.

$$C^e(i,j) = \frac{| N_i \cap N_j |}{| N_i \cup N_j | - 2}$$

- Link property
- Fraction of common neighbours of a connected pair



Farness and Closeness

Farness: average distance to all other nodes in the network

$$\text{Farness}(i) = \frac{1}{N-1} \sum_{j \in V, i \neq j} h_{ij}$$

Closeness: inverse of farness, i.e., how close the node is to all other nodes in terms of shortest paths

$$\text{Closeness}(i) = \frac{N-1}{\sum_{j \in V, i \neq j} h_{ij}}$$

Farness and Closeness

Farness: average distance to all other nodes in the network

$$\text{Farness}(i) = \frac{1}{N-1} \sum_{j \in V, i \neq j} h_{ij}$$

Closeness: inverse of farness, i.e., how close the node is to all other nodes in terms of shortest paths

$$\text{Closeness}(i) = \frac{N-1}{\sum_{j \in V, i \neq j} h_{ij}}$$

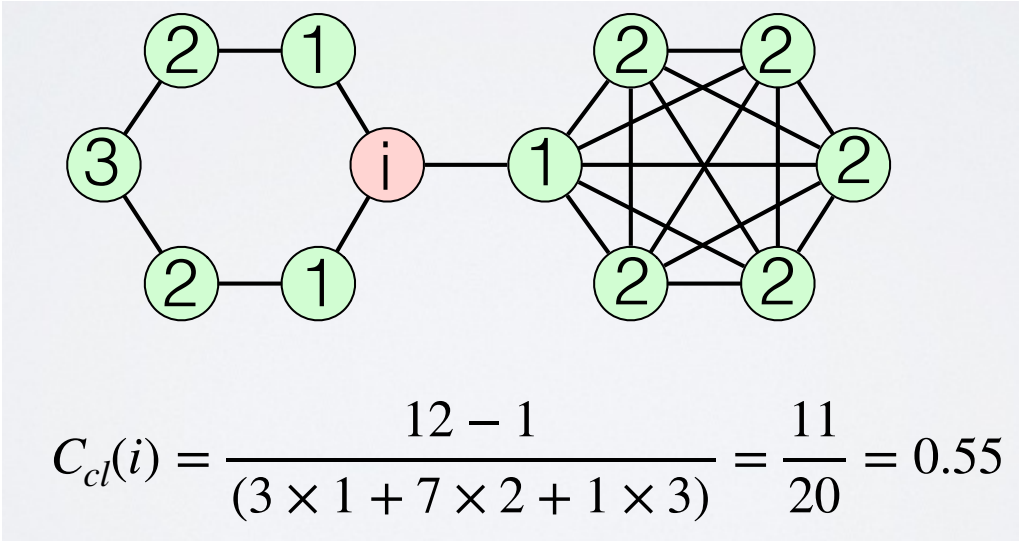
Farness and Closeness

Farness: average distance to all other nodes in the network

$$\text{Farness}(i) = \frac{1}{N-1} \sum_{j \in V, i \neq j} h_{ij}$$

Closeness: inverse of farness, i.e., how close the node is to all other nodes in terms of shortest paths

$$\text{Closeness}(i) = \frac{N-1}{\sum_{j \in V, i \neq j} h_{ij}}$$



$$C_{cl}(i) = \frac{12-1}{(3 \times 1 + 7 \times 2 + 1 \times 3)} = \frac{11}{20} = 0.55$$

Harmonic Closeness

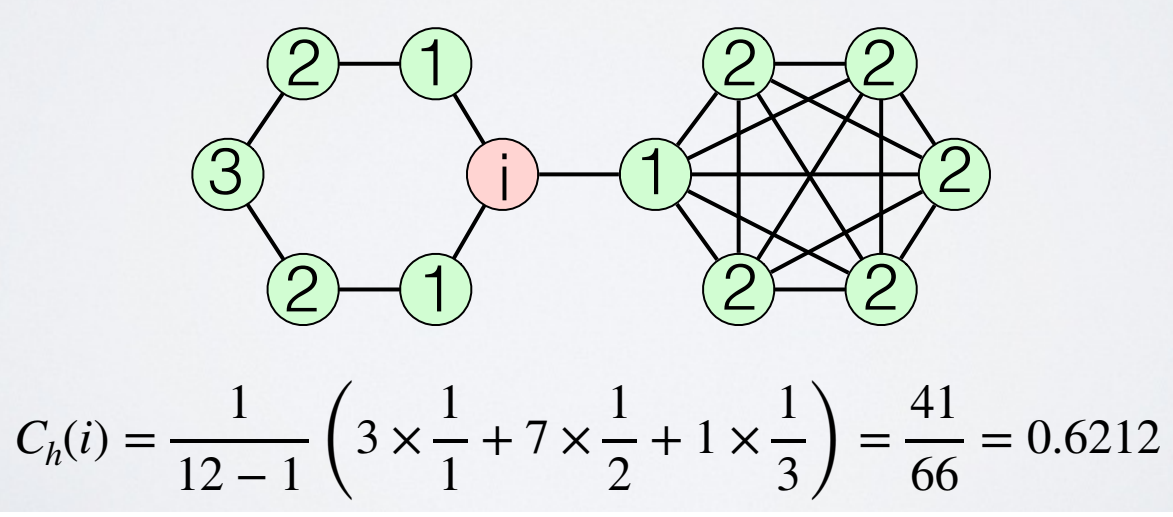
Harmonic Centrality: a variant of the closeness, and it is defined as the average of the inverse of distance to all other nodes (harmonic mean).

$$\text{Harmonic}(i) = \frac{1}{N-1} \sum_{j \in V, i \neq j} \frac{1}{h_{ij}}$$

Harmonic Closeness

Harmonic Centrality: a variant of the closeness, and it is defined as the average of the inverse of distance to all other nodes (harmonic mean).

$$\text{Harmonic}(i) = \frac{1}{N-1} \sum_{j \in V, i \neq j} \frac{1}{h_{ij}}$$



$$C_h(i) = \frac{1}{12-1}\left(3 \times \frac{1}{1} + 7 \times \frac{1}{2} + 1 \times \frac{1}{3}\right) = \frac{41}{66} = 0.6212$$

Betweenness Closeness

- Measure how much the node plays the role of a bridge
- Betweenness of node i: fraction of all the shortest paths between all the pairs of nodes going through i.

$$C_B(i) = \sum_{j \neq i \neq k \in V} \frac{\sigma_{jk}(i)}{\sigma_{jk}}$$

where σ_{jk} is the number of shortest paths between nodes j and k and $\sigma_{jk}(i)$ the number of those paths passing through i.

- The betweenness tends to grow with the network size, thus a normalised version is often used.

$$C_B^{nom}(i) = \frac{C_B(i)}{(N-1)(N-2)}$$

Betweenness Closeness

- Measure how much the node plays the role of a bridge
- Betweenness of node i: fraction of all the shortest paths between all the pairs of nodes going through i.

$$C_B(i) = \sum_{j \neq i \neq k \in V} \frac{\sigma_{jk}(i)}{\sigma_{jk}}$$

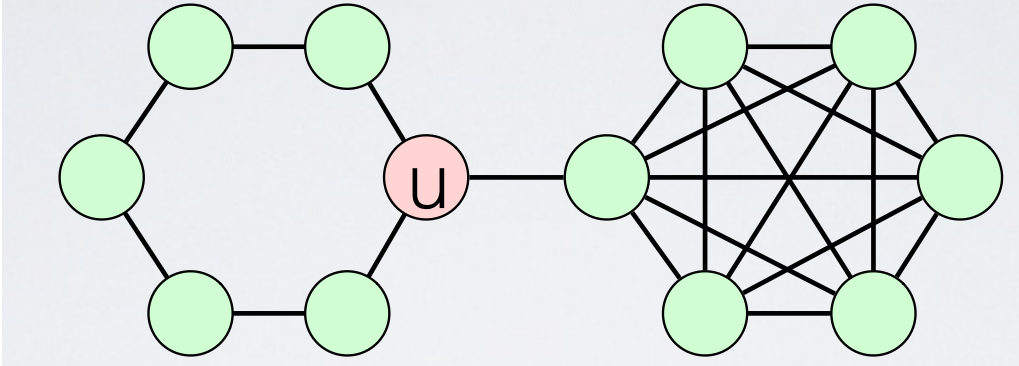
  where σ_{jk} is the number of shortest paths between nodes j and k and $\sigma_{jk}(i)$ the number of those paths passing through i.

- The betweenness tends to grow with the network size, thus a normalised version is often used.

$$C_B^{nom}(i) = \frac{C_B(i)}{(N-1)(N-2)}$$

Betweenness Closeness

$$C_B^{nom}(i) = \frac{C_B(i)}{(N-1)(N-2)}$$

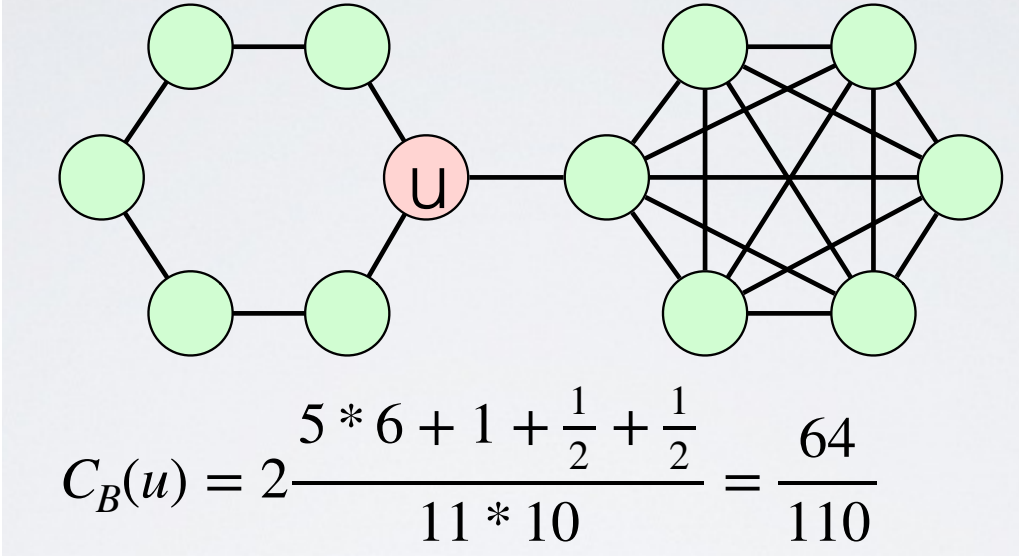


Complexity

- Exact: $\mathcal{O}(N^3)$ (time complexity), $\mathcal{O}(N^2)$ (space complexity)
- Approximate: $\mathcal{O}(N(M + N\log N))$ (time complexity)

Betweenness Closeness

$$C_B^{nom}(i) = \frac{C_B(i)}{(N-1)(N-2)}$$



Complexity

- Exact: $\mathcal{O}(N^3)$ (time complexity), $\mathcal{O}(N^2)$ (space complexity)
- Approximate: $\mathcal{O}(N(M + N\log N))$ (time complexity)

Recursive Definitions

Recursive Definitions

Recursive importance:

- Important nodes are those connected to important nodes
- Several centralities: eigenvector centrality, PageRank, etc.

Recursive Definitions

Recursive importance:

- Each node has a score (centrality)
- If every node "sends" its score to its neighbours, the sum of all scores received by each node will be equal to its original score

$$C^{t+1}(i) = \alpha \sum_{j \in N_i} C^t(j) = \alpha \sum_j A_{ij} C^t(j)$$

with α as a normalisation constant.

Recursive Definitions

Recursive importance:

- This problem can be solved by the Power method:
  1. We initialise all scores to random values
  2. Each score is updated according to the desired rule, until reaching a stable value (after normalisation)
- Why does it converge?
  - Perron-Frobenius theorem
  - True for undirected graphs with a single connected component

Recursive Definitions

Recursive importance:

- This problem can be solved by the Power method:
  1. We initialise all scores to random values
  2. Each score is updated according to the desired rule, until reaching a stable value (after normalisation)
- Why does it converge?
  - Perron-Frobenius theorem
  - True for undirected graphs with a single connected component

Eigenvector Centrality

- What we described in previous slides is eigenvector centrality
- A couple eigenvector x and eigenvalue λ is defined by the following relation

$$Ax = \lambda x$$

  where x is a column vector of size N, interpreted as the importance scores.

- What Perron-Frobenius theorem says is that the Power method will always converge to the leading eigenvector, i.e., the eigenvector associated with the highest eigenvalue.
- Eigenvector centrality does not work in directed acyclic networks, e.g., citation networks.
- Solution: only in strongly connected component

Eigenvector Centrality

- What we described in previous slides is eigenvector centrality
- A couple eigenvector x and eigenvalue λ is defined by the following relation

$$Ax = \lambda x$$

  where x is a column vector of size N, interpreted as the importance scores.
- What Perron-Frobenius theorem says is that the Power method will always converge to the leading eigenvector, i.e., the eigenvector associated with the highest eigenvalue.
- Eigenvector centrality does not work in directed acyclic networks, e.g., citation networks.
- Solution: only in strongly connected component

Eigenvector Centrality

- What we described in previous slides is eigenvector centrality
- A couple eigenvector x and eigenvalue λ is defined by the following relation

$$Ax = \lambda x$$

  where x is a column vector of size N, interpreted as the importance scores.
- What Perron-Frobenius theorem says is that the Power method will always converge to the leading eigenvector, i.e., the eigenvector associated with the highest eigenvalue.
- Eigenvector centrality does not work in directed acyclic networks, e.g., citation networks.
- Solution: only in strongly connected component

Katz or α Centrality

Give each vertex a small amount of centrality for free!

$$C^{t+1}(i) = \alpha \sum_j A_{ij} C^t(j) + \beta$$

where α and β are positive constants. β is the free contribution for all vertices; hence, no vertex has zero centrality and will contribute at least β to other vertices' centrality.

This works in directed acyclic graphs!

PageRank (the Google Algorithm)

PageRank Centrality

An improvement over α centrality.

PageRank

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Brin, S. and Page, L. (1998) The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: Seventh International World-Wide Web Conference (WWW 1998), April 14-18, 1998, Brisbane, Australia.

Sergey Brin and Lawrence Page

Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

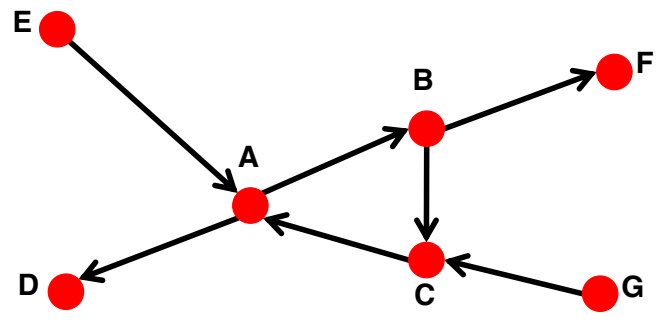
Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/

The Web as a Graph

Web as a directed graph [Broder et al. 2000]:

- Nodes: web pages
- Edges: hyperlinks
- Give node v, what can v reach (i.e., $In(v)$)?
- What other nodes can reach v (i.e., $Out(v)$)?

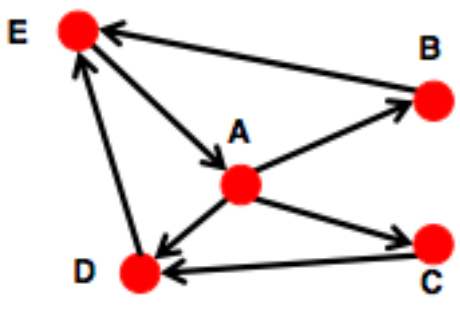


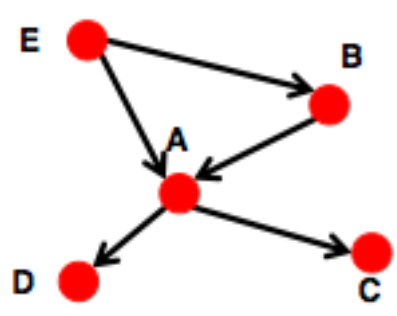
$$Out(A) = \{A, B, C, E, G\},\ In(A) = \{A, B, C, D, F\}$$

The Web as a Graph

Two types of directed graphs:

- Strongly connected: any node can reach any node via a directed graph ($In(v) = Out(v)$)
- Directed acyclic graph (DAG): no cycles (if u can reach v, then v cannot reach u)

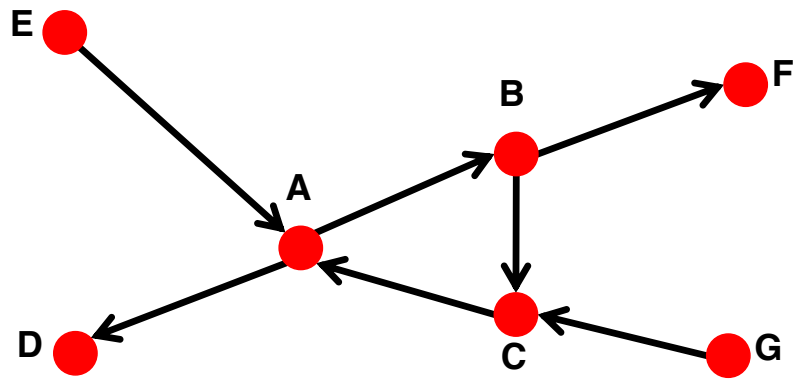




The Web as a Graph

Strongly connected component (SCC): a set of nodes S such that

- Every pair of nodes in S can reach each other.
- There is no larger set containing S with this property.



SCCs: $\{A, B, C\}$, $\{D\}$, $\{E\}$, $\{F\}$,$\{G\}$

The Web as a Graph

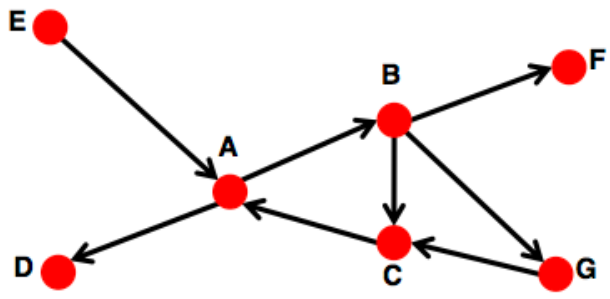
Every directed graph is a DAG on its SCCs.

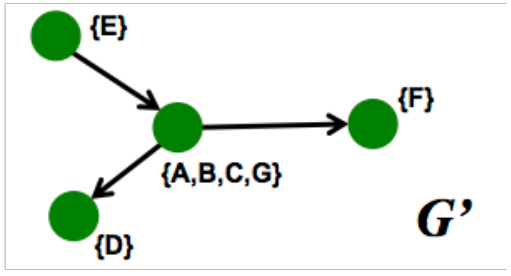
1. SCCs partition the nodes of G (each node in exactly one SCC)
2. If we build a graph G' whose nodes are SCCs, and with an edge between nodes of G' if there is an edge between corresponding SCCs in G, then G' is a DAG.

The Web as a Graph

Every directed graph is a DAG on its SCCs.

1. SCCs partition the nodes of G (each node in exactly one SCC)
2. If we build a graph G' whose nodes are SCCs, and with an edge between nodes of G' if there is an edge between corresponding SCCs in G, then G' is a DAG.





SCCs in G: $\{A, B, C, G\}$, $\{D\}$, $\{E\}$, $\{F\}$, and G' is a DAG

The Web as a Graph

Graph structure of the Web

- There is a single giant SCC (there won't be two SSCs)
- Assume two large SCCs, it just takes one page from one SCC to link to the other.
- If the two SCCs have millions of pages, the likelihood of this not happening is very very small.

The Web as a Graph

Graph structure of the Web

- There is a single giant SCC (there won't be two SSCs)
- Assume two large SCCs, it just takes one page from one SCC to link to the other.
- If the two SCCs have millions of pages, the likelihood of this not happening is very very small.

The Web as a Graph

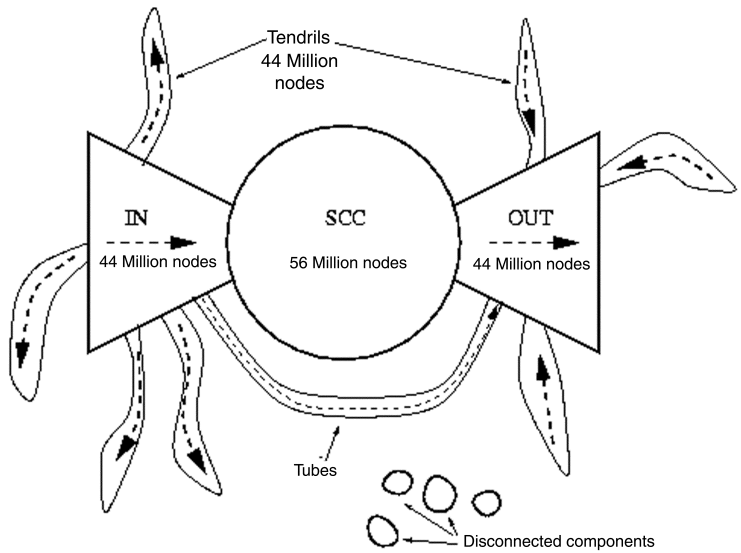
Graph structure of the Web: based on *In* and *Out* of a random node v in a large snapshot of the Web

- $In(v) \approx 100$ million (50% nodes)
- $Out(v) \approx 100$ million (50% nodes)
- largest SCC: $In(v) \cap Out(v) \approx 56$ million (28% nodes)

The Web as a Graph

Graph structure of the Web: based on *In* and *Out* of a random node v in a large snapshot of the Web

- $In(v) \approx 100$ million (50% nodes)
- $Out(v) \approx 100$ million (50% nodes)
- largest SCC: $In(v) \cap Out(v) \approx 56$ million (28% nodes)



A bowtie structure (203 million pages, 1.5 billion links) [Broder et tal. 2000].

The Web as a Graph

How to rank the pages on the Web graph structure?

- In-links as votes
- Links from important pages count more (recursive definition)

PageRank [Brin & Page 1998]

Originally conceived to rank pages in the Web (directed graph)

- $V = \{1, \ldots, n\}$ are the nodes (pages)
- $(i, j) \in E$ if page i points to page j (i.e., $A_{ij} = 1$)
- We associate to each page i, a real value $C(i)$ (i's PageRank).
- We impose that $\sum_{i=1}^{n} C(i) = 1$.

$$C(i) = \alpha \sum_{j=1}^{n} A_{ji} \frac{C(j)}{k_j^{out}} + \beta$$

where $\alpha, \beta > 0$, and k_j^{out} is j's out degree.

- An improvement over α centrality.

PageRank [Brin & Page 1998]

Originally conceived to rank pages in the Web (directed graph)

- $V = \{1, \ldots, n\}$ are the nodes (pages)
- $(i, j) \in E$ if page i points to page j (i.e., $A_{ij} = 1$)
- We associate to each page i, a real value $C(i)$ (i's PageRank).
- We impose that $\sum_{i=1}^{n} C(i) = 1$.

$$C(i) = \alpha \sum_{j=1}^{n} A_{ji} \frac{C(j)}{k_j^{out}} + \beta$$

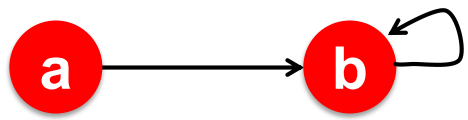
where $\alpha, \beta > 0$, and k_j^{out} is j's out degree.

- An improvement over α centrality.

PageRank [Brin & Page 1998]

Without α and β, there are two problems.

- The spider trap problem (eventually spider traps (all out-links are within the group) absorb all importance)

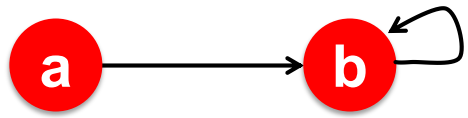


Initial: $C(a) = 1$, $C(b) = 0$; eventually: $C(a) = 0$, $C(b) = 1$.

PageRank [Brin & Page 1998]

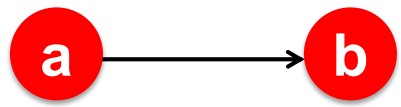
Without α and β, there are two problems.

- The spider trap problem (eventually spider traps (all out-links are within the group) absorb all importance)



Initial: $C(a) = 1,\ C(b) = 0$; eventually: $C(a) = 0,\ C(b) = 1$.

- The dead end problem (pages having no out-links cause importance to "leak out")



Initial: $C(a) = 1,\ C(b) = 0$; eventually: $C(a) = 0,\ C(b) = 0$.

PageRank [Brin & Page 1998]

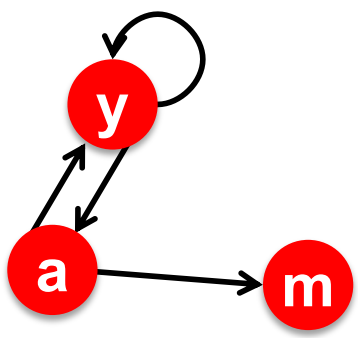
The Google solution for spider traps: at each time step, the random surfer has two options.

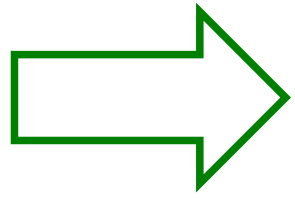
- With probability α, follow a link at random
- With probability $1 - \alpha$, jump to a random page
- Common values for α are in the range 0.8 to 0.9 ($\alpha = 0.85$)

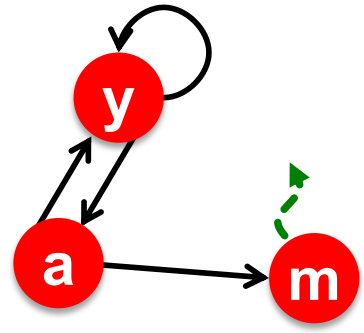
Surfer will teleport out of a spider trap within a few time steps!

PageRank [Brin & Page 1998]

The Google solution for dead ends: follow random teleport links with probability 1.0, adjust matrix accordingly







	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

PageRank [Brin & Page 1998]

Brin and Page consider $\alpha = 0.85$ and $\beta = \frac{1-\alpha}{n}$. Then, we have

$$C(i) = \alpha \sum_{j=1}^{n} A_{ji} \frac{C(j)}{k_j^{out}} + \frac{1-\alpha}{n}$$

Matrix interpretation: the "Google Matrix" G

- First, define (stochastic) matrix S as:
  - normalisation by columns of the original matrix A
  - columns with only 0 receive $\frac{1}{n}$
- Finally, $G_{ij} = \alpha S_{ij} + \frac{(1-\alpha)}{n}$

So, we see a solution π for $G\pi = \pi$.

PageRank [Brin & Page 1998]

Brin and Page consider $\alpha = 0.85$ and $\beta = \frac{1-\alpha}{n}$. Then, we have

$$C(i) = \alpha \sum_{j=1}^{n} A_{ji} \frac{C(j)}{k_j^{out}} + \frac{1-\alpha}{n}$$

Matrix interpretation: the "Google Matrix" G

- First, define (stochastic) matrix S as:
  - normalisation by columns of the original matrix A
  - columns with only 0 receive $\frac{1}{n}$
- Finally, $G_{ij} = \alpha S_{ij} + \frac{(1-\alpha)}{n}$

So, we see a solution π for $G\pi = \pi$.

PageRank [Brin & Page 1998]

$G\pi = \pi$ can be solved by the Power method, which converges fast to the PageRank solution.

1. We initialise all scores to random values
2. Each score is updated according to the desired rule, until reaching a stable value (after normalisation)

PageRank [Brin & Page 1998]

Theorem (Perron-Frobenius)

If M is stochastic, then it has at least one stationary vector, i.e., one non-zero vector p such that $M^T p = p$.

The transpose of Google matrix G is row-stochastic.

PageRank [Brin & Page 1998]

Theorem (Perron-Frobenius)

If M is stochastic, then it has at least one stationary vector, i.e., one non-zero vector p such that $M^T p = p$.

The transpose of Google matrix G is row-stochastic.

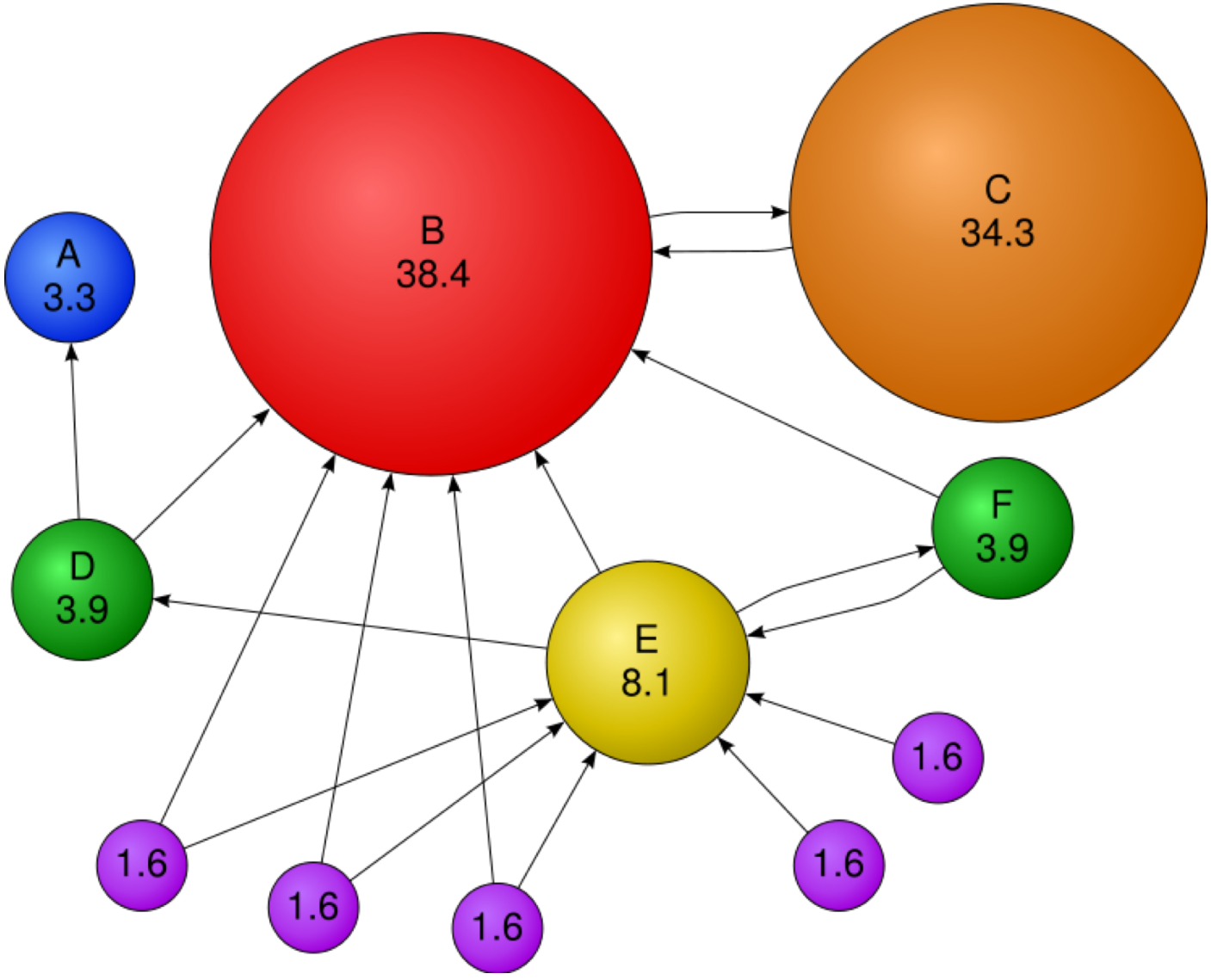
PageRank [Brin & Page 1998]

Theorem

If a matrix M is strongly connected and aperiodic, then

- $M^T p = p$ has exactly one non-zero solution such that $\sum_i p_i = 1$
- 1 is the largest eigenvalue of M^T
- the Power method converges for any initial non-zero vector
- furthermore, we have exponential fast convergence

PageRank [Brin & Page 1998]



PageRank [Brin & Page 1998]

The PageRank computation can be interpreted as a Random Walk process with restart.

- Teleportation probability: α gives the probability that in the next step of the random walk will follow a Markov process or with probability $1 - \alpha$ it will jump to a random node.
- Pagerank score of a node thus corresponds to the probability of this random walker to be on this node after an infinite number of hops.

Node Similarity

Similarity

It is desirable that it has the properties of a distance metric (except possibly for triangle inequality which may not hold if the graph is not complete).

- $d(x, y) \geq 0$ and $d(x, x) = 0$
- $d(x, y) = d(y, x)$
- $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

If we want to interpret high value of similarity as high similarity, and we are working with distance metric $d(x, y)$, we can consider its inverse:

$$s(x, y) = \frac{1}{d(x, y)}$$

Similarity

It is desirable that it has the properties of a distance metric (except possibly for triangle inequality which may not hold if the graph is not complete).

- $d(x, y) \geq 0$ and $d(x, x) = 0$
- $d(x, y) = d(y, x)$
- $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

If we want to interpret high value of similarity as high similarity, and we are working with distance metric $d(x, y)$, we can consider its inverse:

$$s(x, y) = \frac{1}{d(x, y)}$$

Node Similarity

Similarity between two nodes i and j based on their neighbourhood

- Common neighbours: $s_{ij} = \mid N_i \cap N_j \mid$
- Based on the common neighbours measure, high-degree nodes are likely to attain large similarity scores, even when only a small fraction of their neighbours is shared.

Node Similarity

Similarity between two nodes i and j based on their neighbourhood

- One way of correcting the bias is to divide by the size of the union of the two neighbourhoods.
- Jaccard similarity:

$$s(i,j) = \frac{|\ N_i \cap N_j\ |}{|\ N_i\ | + |\ N_j\ | - |\ N_i \cap N_j\ |}$$

This can also be defined, using the adjacency matrix A.

$$s(i,j) = \frac{\sum_k A_{ik} A_{kj}}{\sum_k (A_{ik} + A_{jk})}$$

Node Similarity

Similarity between two nodes i and j based on their neighbourhood

- One way of correcting the bias is to divide by the size of the union of the two neighbourhoods.
- Jaccard similarity:

$$s(i,j) = \frac{|\, N_i \cap N_j \,|}{|\, N_i \,| + |\, N_j \,| - |\, N_i \cap N_j \,|}$$

This can also be defined, using the adjacency matrix A.

$$s(i,j) = \frac{\sum_k A_{ik} A_{kj}}{\sum_k (A_{ik} + A_{jk})}$$

Node Similarity

Similarity between two nodes i and j based on their neighbourhood

- A second way of correcting the bias of the common neighbours similarity measure is to weight exclusive neighbours more heavily than neighbours that are shared by many nodes.
- Adamic and Adar score:

$$s(i,j) = \sum_{k \in N_i \cap N_j} \frac{1}{|\ N_k\ |}$$

Node Similarity

Similarity between two nodes i and j based on their neighbourhood

- Cosine similarity:

$$s(i,j) = \frac{|\ N_i \cap N_j\ |}{\sqrt{|\ N_i\ ||\ N_j\ |}}$$

- Number of common neighbours normalised by the geometric mean of their degrees.

Node Similarity

Euclidean distance: or rather Hamming distance since A is binary (a dissimilarity)

$$d(i,j) = \sum_k (A_{ik} - A_{jk})^2$$