CentraleSupélec

ESSEC BUSINESS SCHOOL

Network Science Analytics

Option Applied Math and M.Sc. in DSBA

Lecture 3B

Centrality criteria and link analysis algorithms

Fragkiskos Malliaros

Friday, February 2, 2018

Acknowledgements

- The lecture is partially based on material by
  - Jure Leskovec, Stanford University
  - Aaron Clauset, CU Boulder
  - Manos Papagelis, York University
  - Gonzalo Mateos, University of Rochester
  - Albert-László Barabási, Northeastern University
  - Christos Faloutsos, CMU
  - Konstantinos Pelechrinis, University of Pittsburgh
  - Panagiotis Tsaparas, University of Ioannina
  - R. Zafarani, M. A. Abbasi, and H. Liu, Social Media Mining: An Introduction, Cambridge University Press, 2014. Free book and slides at http://socialmediamining.info/

Thank you!

In this Lecture

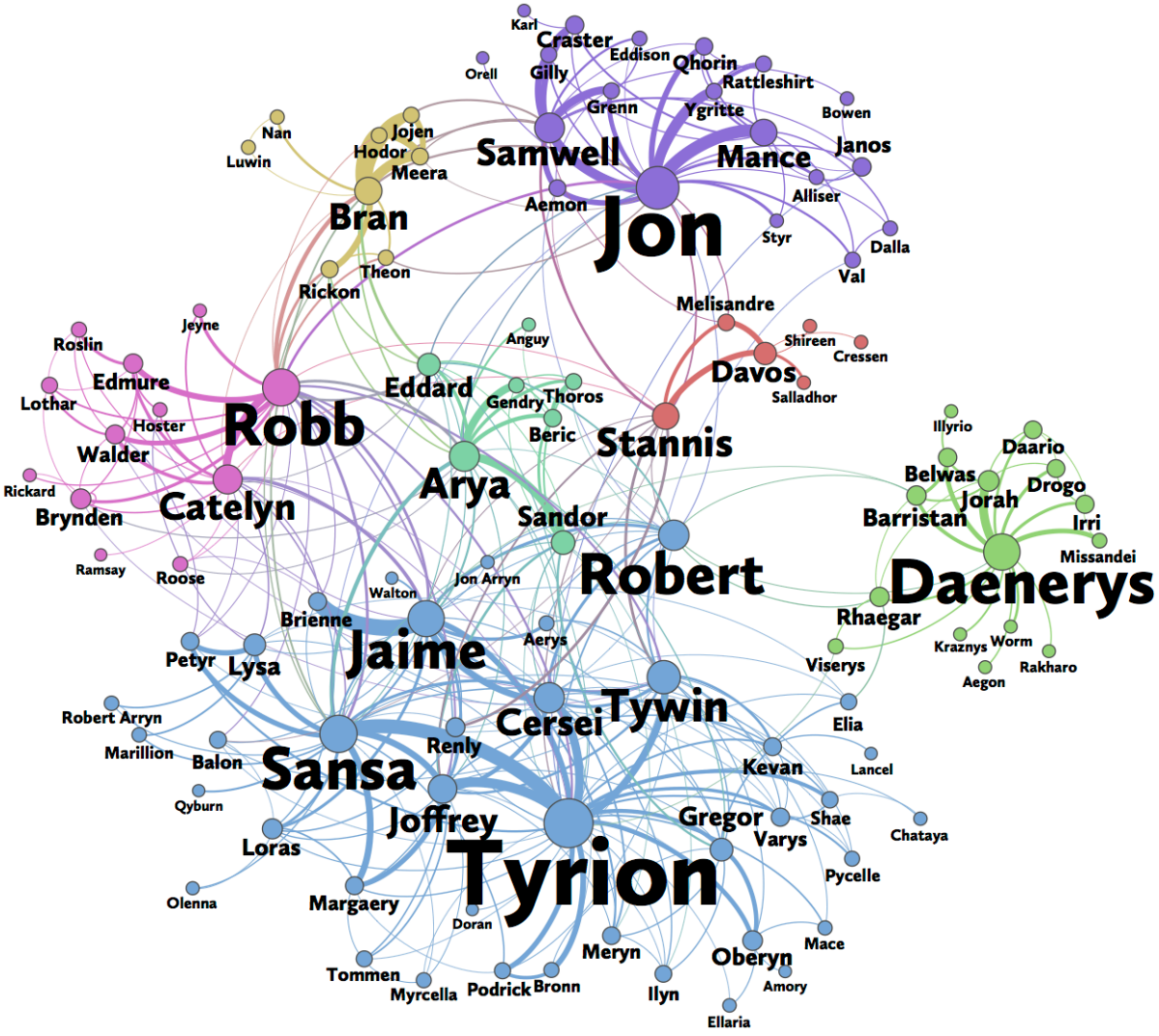
- Centrality criteria in graphs
- Link analysis algorithms

Centrality criteria

Centrality in Networks (1/2)

- Determine the relative importance of a node in the network
  - Applications in Social Network Analysis, the Internet, Epidemiology, Urban informatics, …
- What do we mean by centrality?
  - A central node is more important or powerful …
  - Or, more influential ...
  - Or, is more critical due to its location in the graph
- Also, very closely related to the problem of ranking in the context of Web search
  - Each webpage can be considered as a 'user'
  - Each hyperlink is an endorsement relationship
  - Centrality measures provide a query independent link-based score of importance of a web page

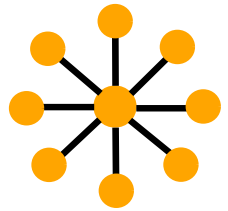
Centrality in Networks (2/2)



Source: https://www.macalester.edu/~abeverid/thrones.html

Types of Centrality

- Starting point: the central node of a star is the most important
- Why?
  - The node with the highest degree
  - The node that is closest to the rest nodes (e.g., has the smallest average distance to other nodes)
  - The node through which all shortest paths pass
  - The node that maximizes the dominant eigenvector (the one that corresponds to the largest eigenvalue) of the adjacency matrix
  - The node with highest probability in the stationary distribution of a random walk on the graph



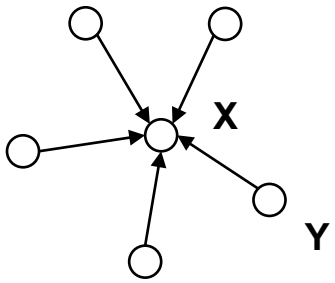
Various competing views of centrality

Measures of Centrality

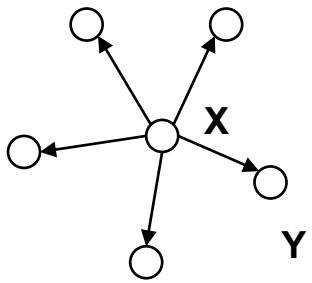
- This observation leads to the following classes of indices of centrality:
  - Measures based on distances (e.g., degree, closeness)
  - Measures based on paths (e.g., betweenness, Katz's index)
  - Spectral measures (eigenvector, PageRank, HITS, SALSA, random walks with restarts)
  - Measure based on groups of nodes (e.g., cliques, plexes, cores)
    - Related to the "clustering" structure
    - More on that in another lecture

A First Example

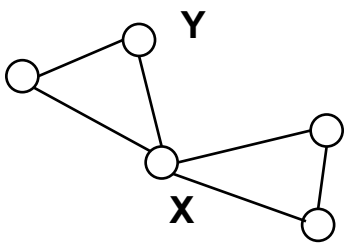
In each of the following networks, X has higher centrality than Y according to a particular measure



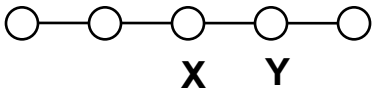
in-degree



out-degree



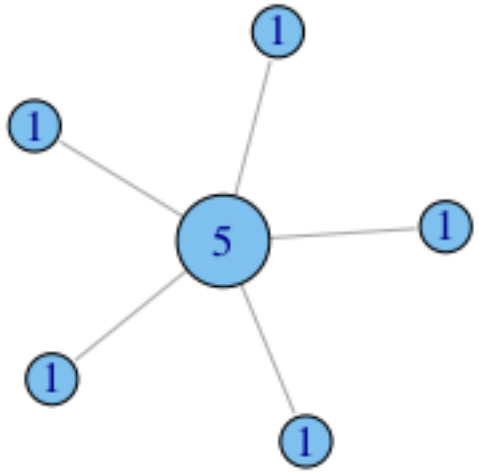
betweenness



closeness

Degree Centrality (1/2)

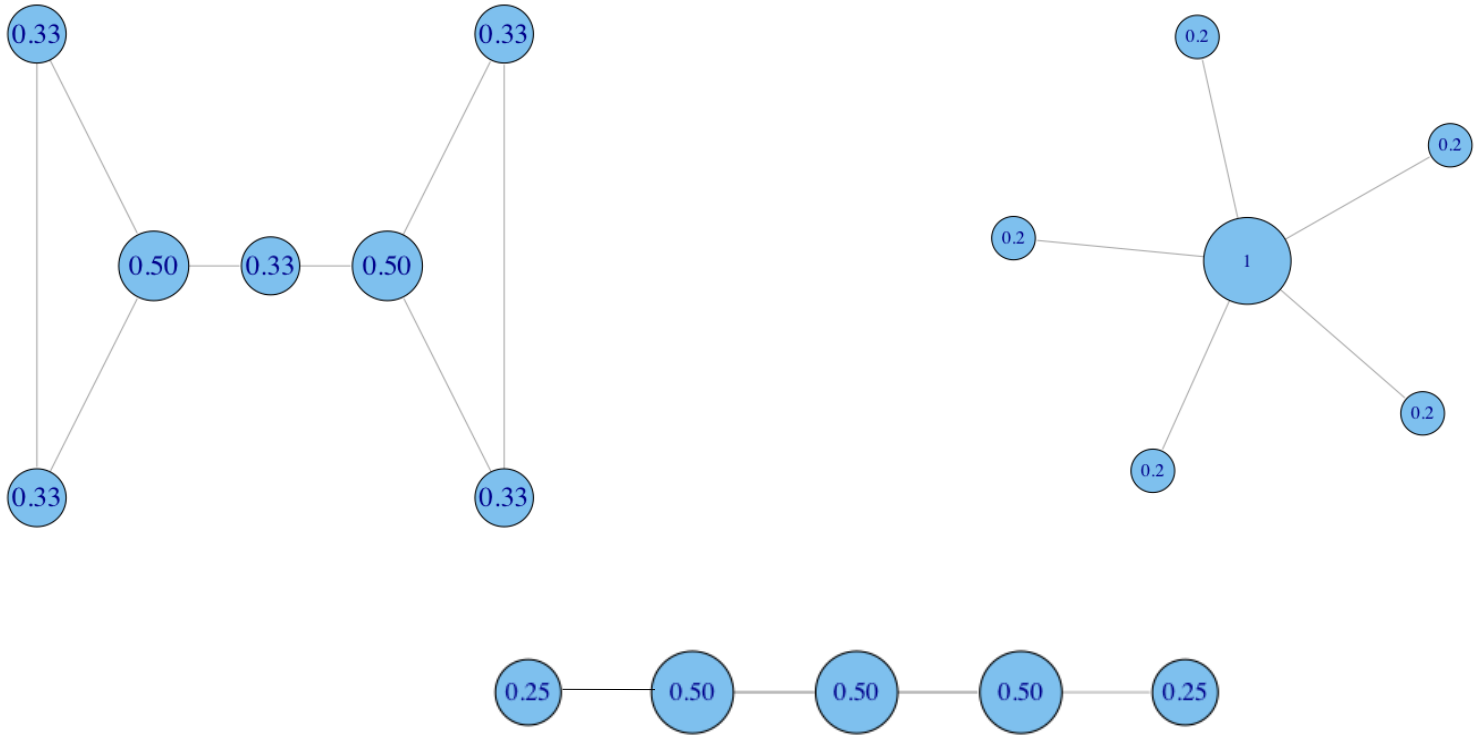
- Idea: A central node is one with many connections



- $C_d\ (i) = k(i)$, where $k(i)$ is the degree of node i

Degree Centrality (2/2)

- Idea: A central node is one with many connections



- Normalized degree centrality: divide by the max possible degree (*n-1*)

Degree Centrality in Directed Graphs

- In directed graphs, we can use the
  - in-degree
  - out-degree
  - Combination of them (e.g., sum of in- and out-degree)

- Which one is more important?
  - In practice, mostly the in-degree is used
  - Why? Think about the Web graph

Closeness Centrality (1/2)

- Motivation: it measures the ability to quickly access or pass information through the graph

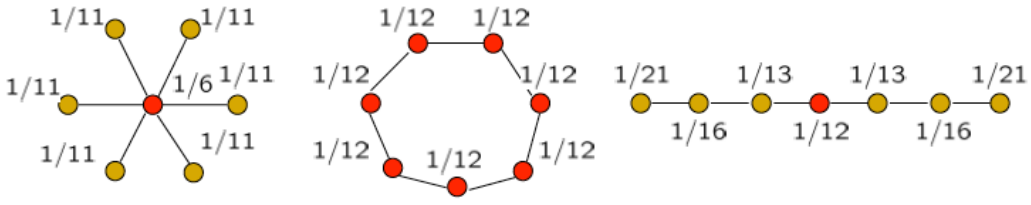
$$C_{cl}(i) = \frac{n-1}{\sum_{j \neq i} d(i,j)}$$

values in the range [0,1]

Mean distance from a node to other nodes

d(i, j) is the length of the shortest path between *i* and *j* (geodesic distance)

- The closeness of a node is defined as the inverse of the sum of the shortest path (SP) distances between the node and all other nodes in the graph



Be close to everybody else
(e.g., influence on other nodes)

Why inverse the distance?
- Nodes with low mean distance should get high score

Closeness Centrality (2/2)

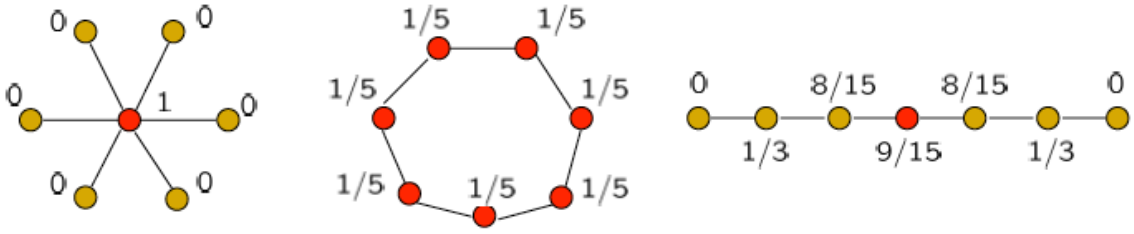
- Complexity: we need all the pairwise shortest path distances in G
  - Dijkstra's algorithm in $O(n^2 \log n + nm)$
- Limitation 1: values tend to span a small dynamic range
  - Why? Geodesic distances in real networks tend to be small: smallest distance is 1; largest distance is $O(\log n)$
  - Difficult to distinguish between central and less central nodes
  - Sensitivity: small fluctuations in the structure of the graph may change the centrality values
- Limitation 2: assumes connectivity (otherwise $C_{cl}(i) = 0$, for all the nodes in the graph)
  - Compute centrality indices in different components

Betweenness Centrality (1/2)

- Motivation: a node is important if it lies in many shortest paths

$$C_{bt}(i) = \sum_{s \neq i \neq t \in V} \frac{\sigma(s,t|i)}{\sigma(s,t)}$$

- $\sigma(s, t)$ is the total number of shortest paths from s to t
- $\sigma(s, t|i)$ is the number of shortest paths from s to t that pass through i



Essential nodes in passing information through the network

Oftentimes it is normalized: $\frac{C_{bt}(i)}{\binom{n-1}{2}}$

Betweenness Centrality (2/2)

Computational issues

- Notice that a s-t shortest path goes through v if and only if

$$d(s,t) = d(s,v) + d(v,t)$$

- Naïve computation for all nodes v
  - Step 1: Run Dijkstra to compute $d(s,t)$ and $\sigma(s,t)$ for all s, t
  - Step 2: Identify $\sigma(s, t|v)$ for all nodes v
  - Step 3: Sum the fractions to obtain $C_{bt}(v)$ for all v

Running Time of BC

$$C_{bt}(i) = \sum_{s \neq i \neq t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

- $\sigma(s, t)$ is the total number of shortest paths (SP) from s to t
- $\sigma(s, t|v)$ is the number of shortest paths from s to t that pass through v

- Two main steps in the algorithm:
  - Compute $\sigma(s,t)$ and $\sigma(s, t|v)$ for all s, t, v via all pair shortest paths
  - Perform the aggregation to obtain $C_{bt}(i)$ for all i
  - Total: $\Theta(n^3)$

Brandes' algorithm interleaves the SP computations with the aggregation
Running time: $O(nm + n^2 \log n)$

[Brandes '01]

Dependencies

[Brandes '01]

- Definition: Dependency of *s* on *v* :

$$\delta_s(v) = \sum_{t \neq s \neq v} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

Ratio of SP between s-t that v lies on

- Therefore:

$$C_{bt}(v) = \sum_{s \neq v} \delta_s(v)$$

Sum the dependencies over all nodes s in V

- Brandes proved that $\delta_s(v)$ obeys a recursive relation

$$\delta_s(v) = \sum_{w: v \in P_s(w)} \frac{\sigma(s,v)}{\sigma(s,w)}(1 + \delta_s(w))$$

$P_s(w)$: set of predecessors of *w* in the SPs from *s* to *w*

We can leverage this relation for efficient computation of betweenness

Brandes' Algorithm for BC

1. Initialize $\delta_s(v)$ to 0 for each v, s and $C_{bt}(w)$ to 0 for each w
2. Iterate the following loop for each vertex s :
    1. Run Dijkstra's algorithm from s, keeping track of $\sigma(s,v)$ for each encountered node v, and inserting the vertices in a max-heap H by distance from s
    2. While H is not empty
        1. Pop the max node t in H
        2. For each w in $P_s(t)$, increment $\delta(w)$ by
        3. Increment $C_{bt}(t)$ by $\delta_s(t)$

[Brandes '01]

Eigenvector Centrality (1/3)

- Degree centrality assumes all neighbors are equal
  - Only the number of neighbors matters

- However, in many circumstances the importance of a node increases if it is connected to other important nodes
  - Eigenvector centrality

- Assume initially everyone has score $x_i=1$ and we update its centrality using the sum of the scores/centralities of his neighbors

$$x'_i = \sum_j \mathbf{A}_{ij}\, x_j \Rightarrow \mathbf{x}' = \mathbf{A}\mathbf{x}$$

Eigenvector Centrality (2/3)

- After *t* updates we will have: $\mathbf{x}(t) = \mathbf{A}^t \mathbf{x}(0)$ Vector of centralities

- Expressing vector *x(0)* as a linear combination of the eigenvectors of *A* we have: $\mathbf{x}(0) = \sum_i c_i u_i$

- Then, $\mathbf{x}(t) = \mathbf{A}^t \sum_i c_i u_i = \sum_i c_i \lambda_i^t u_i = \lambda_i^t \sum_i c_i \left(\frac{\lambda_i}{\lambda_1}\right)^t u_i$ where λ_1 is the largest eigenvalue
  - Since all, but for *i=1*, the terms in the sum decay exponentially as *t* grows, we have for $t \to \infty \Rightarrow x(t) = c_1 \lambda_1^t u_1$ $\lambda_i / \lambda_1 < 1$
  - Therefore, vector *x* is basically the leading eigenvector of *A*

$$\mathbf{A}\mathbf{x} = \lambda_1 \mathbf{x}$$ Eigenvector centrality

Eigenvector Centrality (3/3)

- As wanted, the centrality x_i is proportional to the sum of the centralities of i's neighbors

$$x_i = \frac{1}{\lambda_1} \sum_j \mathbf{A}_{ij} x_j$$

- The eigenvector centrality of a node can be large for two reasons
  - The node has many neighbors and/or
  - The node has important neighbors

- Note: All eigenvector centrality values are non-negative

Eigenvector Centrality in Directed Networks

- The adjacency matrix is general asymmetric
  - Two leading eigenvectors (recall the SVD decomposition)
  - Which one to use?

- In most cases we use the right eigenvector
  - The centrality in most of the cases is conferred by other vertices pointing towards you

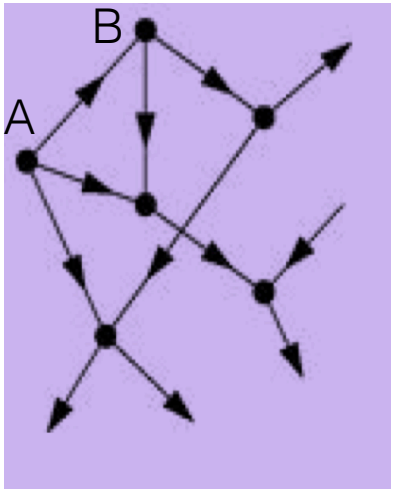
$$x_i = \frac{1}{\lambda_1} \sum_j \mathbf{A}_{ij} x_j$$

where *x* is the right leading eigenvector

Why?
- On WWW, the pages that point to your page provide a reasonable evidence of how important your page is

Any Problems with That?

- Q: What is the eigenvector centrality of node A?
  - Perhaps zero (no in-coming edges)
  - What about node B?



Only nodes that are in a strongly connected component of two or more nodes can have non-zero eigenvector centrality

In acyclic graphs, the eigenvector centrality is completely useless

Katz Centrality

- Can we overcome problems as the above in directed networks?
  - Idea: we can give each node a small amount of centrality for free

$$x_i = \alpha \sum_j A_{ij} x_j + \beta \Rightarrow x = \alpha A x + \beta \vec{1}$$

$\vec{1}$ = [1, 1, …, 1]

  - α, β are positive constants
  - The first term is the normal eigenvector centrality and the second is the "free" part
  - Even nodes with zero in-degree still get centrality β

- By setting $\beta=1$ we have: $x = (I - \alpha A)^{-1}\vec{1}$
  - Katz centrality

How do we set the free parameter α?
(Balance between the eigenvector term and the constant term)

- β is just a multiplier
- We don't care about the absolute magnitude of centrality; we can set $\beta=1$

Choice of Parameter α

- Parameter α governs the balance between the eigenvector centrality contribution and the constant term
  - For small α the Katz centrality is dominated by the constant factor
    - Thus, all nodes have a centrality of 1 ($\beta=1$)
  - For large values of α Katz centrality diverges

- As we increase α from zero, the centralities increase and there comes a point at which they diverge
  - This happens at the point where $(I - \alpha A)^{-1}$ diverges
    - When is this happening: $\det(I - \alpha A) = 0 \rightarrow \det(A - \alpha^{-1} I) = 0$
  - The roots α^{-1} are equal to the eigenvalues of A
  - As α increases the determinant first crosses zero when $\alpha = 1/\lambda_1$, where λ_1 is the largest eigenvalue of A
    - Hence α should be less than $1/\lambda_1$

Eigenvalues: $Au=\lambda u$
Then, $(A - \lambda I)u=0$ has non-zero solutions for u only if $(A - \lambda I)$ cannot be inverted -> $\det(A-\lambda I)=0$

Extension of Katz Centrality

- A possible extension of the above definition is to assign different constant β centrality to different nodes

$$x_i' = \alpha \sum_j A_{ij} x_j + \beta_i \Rightarrow x = (I - \alpha A) + \beta$$

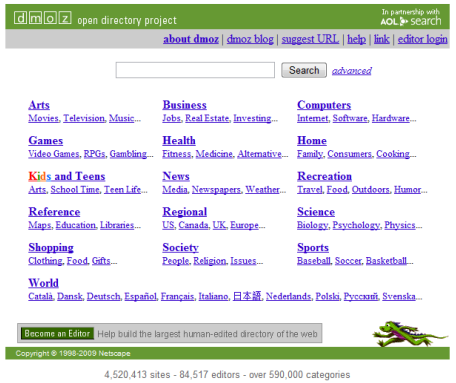
vector

  - β_i is some intrinsic, non-network contribution to the centrality for each node
  - E.g., in a social network the importance of an individual might depend on non-network factors as well, such as age or income, etc.

Link analysis algorithms

How to Organize the Web?

- How to organize the Web?
- First try: Human curated Web directories
  - Yahoo, DMOZ, LookSmart
- Second try: Web Search
  - Information Retrieval attempts to find relevant docs in a small and trusted set
    - Newspaper articles, patents, etc.
  - But: Web is huge, full of untrusted documents, random things, web spam, etc.
  - So we need a good way to rank webpages!



Web Search: Two Challenges

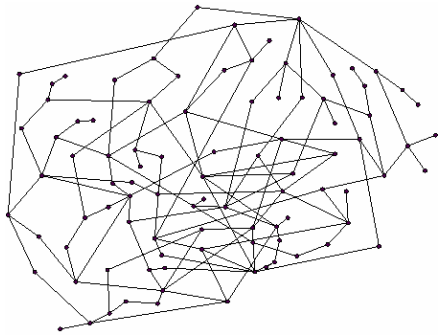
1. Web contains many sources of information
   Which one to "trust"?
   - Insight: Trustworthy pages may point to each other

2. What is the "best" answer to query "newspaper"?
   - No single right answer
   - Insight: Pages that actually know about newspapers might all be pointing to many newspapers

Ranking Pages on the Web Graph

- All web pages are not equally important

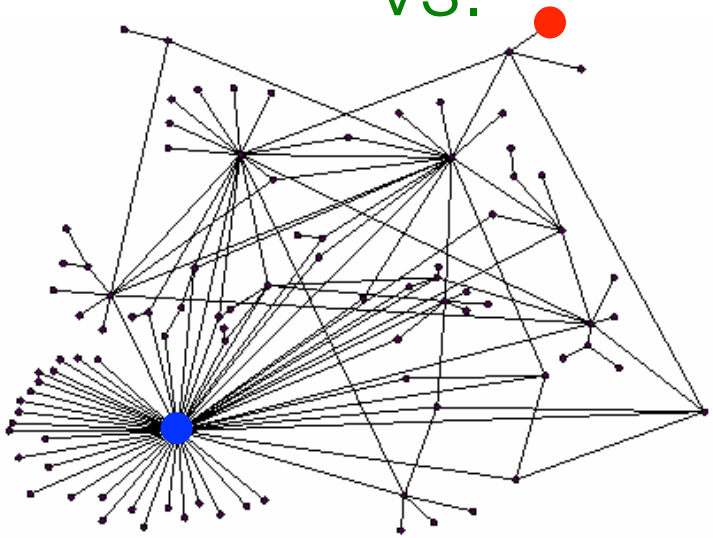
  www.joe-schmoe.com vs. www.centralesupelec.fr



- We already know

  that there is large diversity in the web-graph node connectivity

VS.



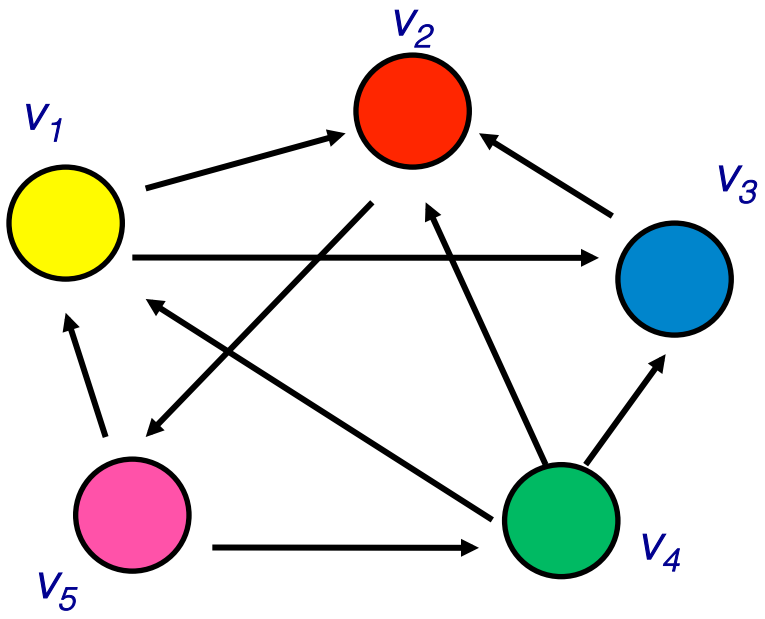
So, let's rank the pages using the web graph link structure

Link Analysis Algorithms

- We will cover the following Link Analysis approaches to compute importance of nodes in a graph:
  - Hubs and Authorities (HITS)
  - PageRank

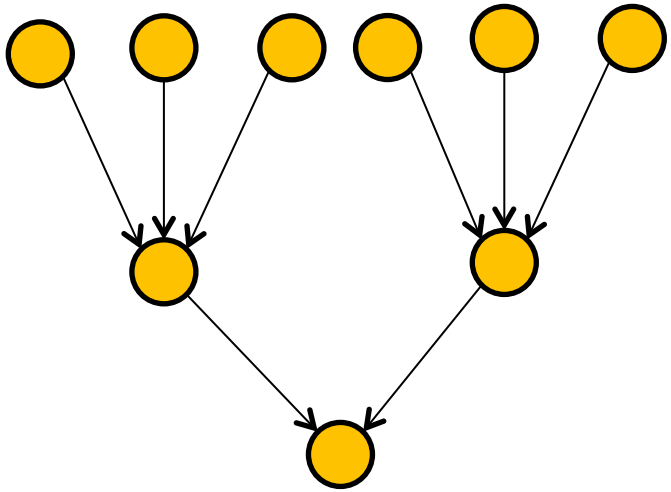
Rank by Popularity

- Rank pages according to the number of incoming edges (in-degree, degree centrality)



1. Red Page
2. Yellow Page
3. Blue Page
4. Purple Page
5. Green Page

Popularity



- It is not important only how many pages link to you
  - But also how important are the pages that link to you
- Good authorities are pointed by good authorities
  - Recursive definition of importance

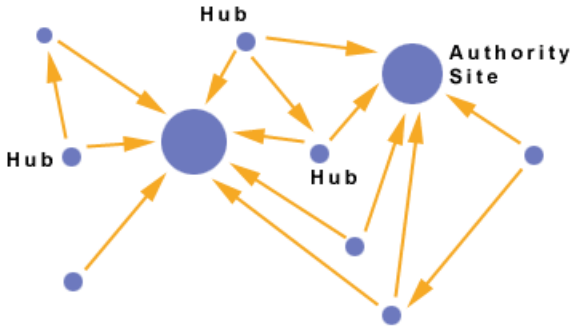
The HITS Algorithm

(Hubs and Authorities)

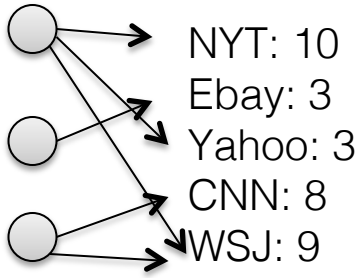
Hubs and Authorities (1/3)

Interesting pages fall into two classes:

1. Authorities are pages containing useful information
   - Newspaper home pages
   - Course home pages
   - Home pages of auto manufacturers

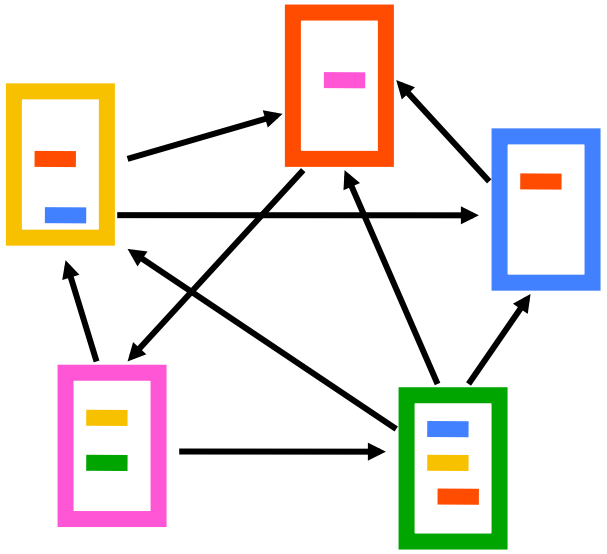


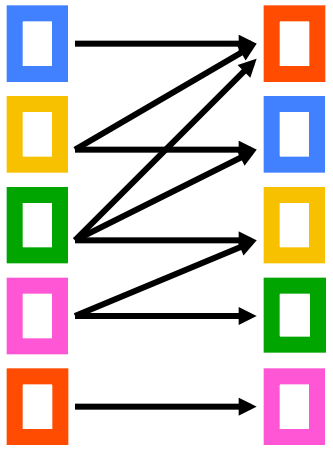
2. Hubs are pages that link to authorities
   - List of newspapers
   - Course bulletin
   - List of U.S. auto manufacturers



Hubs and Authorities (2/3)

- Pages have double identity
  - Hub identity
  - Authority identity
- Good hubs point to good authorities
- Good authorities are pointed by good hubs





Hubs and Authorities (3/3)

- Two kind of weights:
  - Hub weight
  - Authority weight

- The hub weight is the sum of the authority weights of the authorities pointed to by the hub

- The authority weight is the sum of the hub weights that point to this authority

- Represented as vectors h and a, where the i^{th} element is the hub/authority score of the i^{th} node

HITS Algorithm

- Initialize: $\alpha_j^0 = 1/\sqrt{n}, \quad h_j^0 = 1/\sqrt{n}$
- Repeat until convergence
  - Authority: $\alpha_i^{(t+1)} = \sum_{j \to i} h_j^{(t)}, \quad \forall i$
  - Hub: $h_i^{(t+1)} = \sum_{i \to j} \alpha_j^{(t)}, \quad \forall i$
  - Normalize: $\sum_i \left(\alpha_i^{(t+1)}\right)^2 = 1$ and $\sum_j \left(h_j^{(t+1)}\right)^2 = 1$

[Kleinberg '98]

HITS and Eigenvectors

- HITS in vector notation
  - $\alpha = [\alpha_1, \alpha_2, ..., \alpha_n]^T$ and $h = [h_1, h_2, \ldots, h_n]^T$
- We can rewrite α_i and h_i based on the adjacency matrix

$$h_i = \sum_{i \to j} \alpha_j \quad \text{as} \quad h_i = \sum_j \mathbf{A}_{ij} \cdot \alpha_j$$

- Thus, $h = A\,\alpha$ and $\alpha = A^T h$

- $\alpha^{(t+1)} = A^T h^{(t)}$ and $h^{(t+1)} = A\,\alpha^{(t)}$
- $\alpha^{(t+1)} = A^T A\,\alpha^{(t)}$ and $h^{(t+1)} = AA^T h^{(t)}$

Repeated iterations will converge to the eigenvectors

SVD

- Authority weight vector α : eigenvector of A^TA
- Hub weight vector h : eigenvector of AA^T

The vectors α and h are the singular vectors of A

The PageRank Algorithm

PageRank

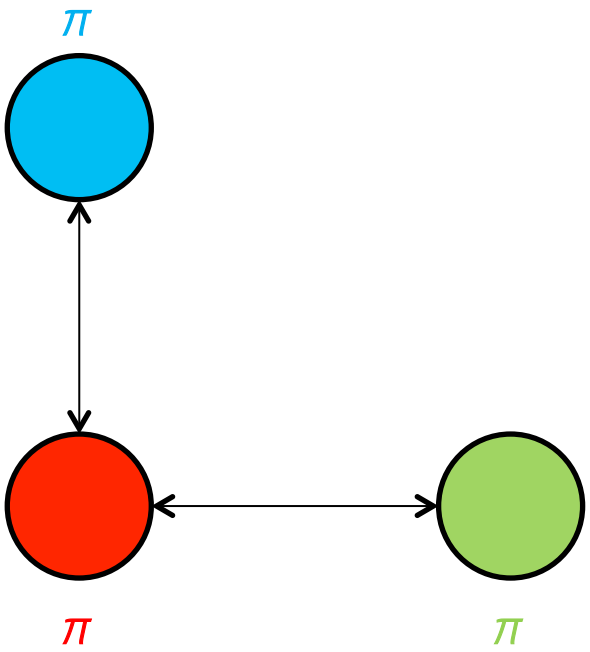
- Good authorities should be pointed by good authorities
  - The value of a node comes from the value of the nodes that point to it

- How do we implement that?
  - Assume that we have a unit of authority to distribute to all nodes
    - Initially, each node gets *1/n* amount of authority
  - Each node distributes its authority value to its neighbors
  - The authority value of each node is the sum of the authority fractions that they collect from their neighbors

$$\pi_v = \sum_{\forall (u,v) \in E} \frac{1}{k_{out}(u)} \pi_u$$

π_v: the PageRank value of node *v*
- Recursive definition

A Simple Example



$\pi + \pi + \pi = 1$

$\pi = \pi + \pi$

$\pi = ½\ \pi$

$\pi = ½\ \pi$

- Solving the system of equations we get the authority values for the nodes
  - $\pi = ½\ \ \pi = ¼\ \ \pi = ¼$

A More Complex Example

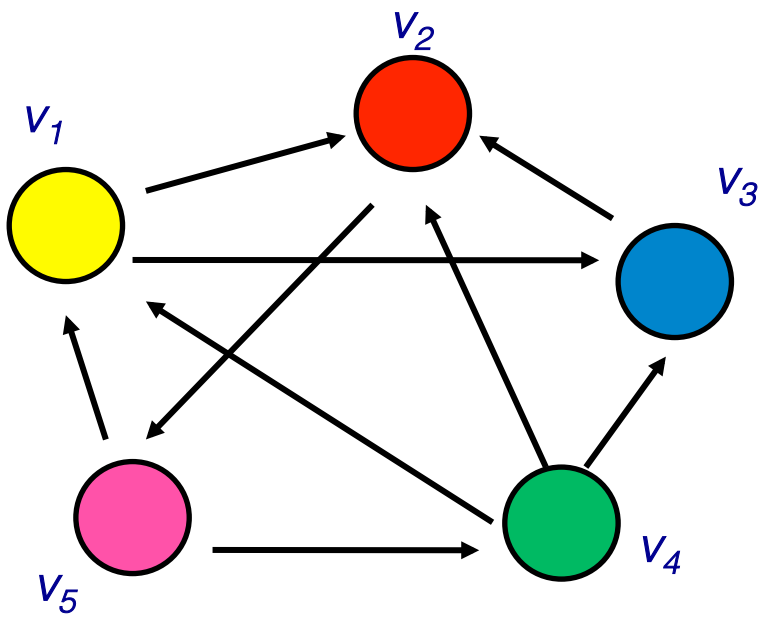
$\pi_1 = 1/3\ \pi_4 + 1/2\ \pi_5$

$\pi_2 = 1/2\ \pi_1 + \pi_3 + 1/3\ \pi_4$

$\pi_3 = 1/2\ \pi_1 + 1/3\ \pi_4$

$\pi_4 = 1/2\ \pi_5$

$\pi_5 = \pi_2$



$$\pi_v = \sum_{\forall (u,v) \in E} \frac{1}{k_{out}(u)} \pi_u$$

Computing PageRank Weights

- A simple way to compute the weights is by iteratively updating the weights

Initialize all PageRank weights to *1/n*

Repeat:

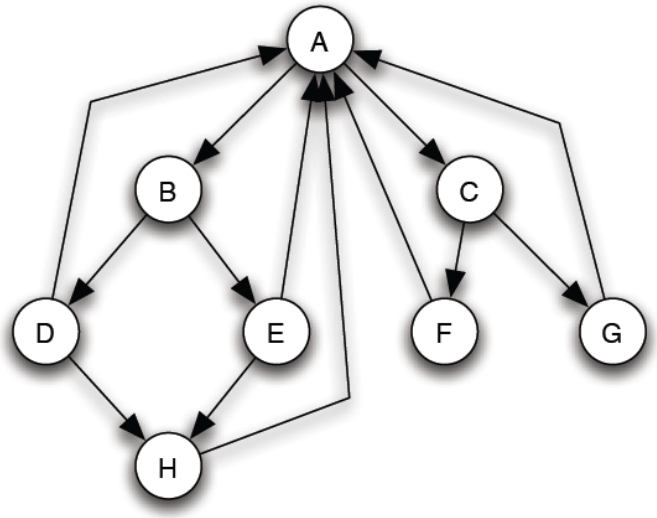
$$\pi_v = \sum_{\forall (u,v) \in E} \frac{1}{k_{out}(u)} \pi_u$$

Until the weights do not change

This process converges

Example

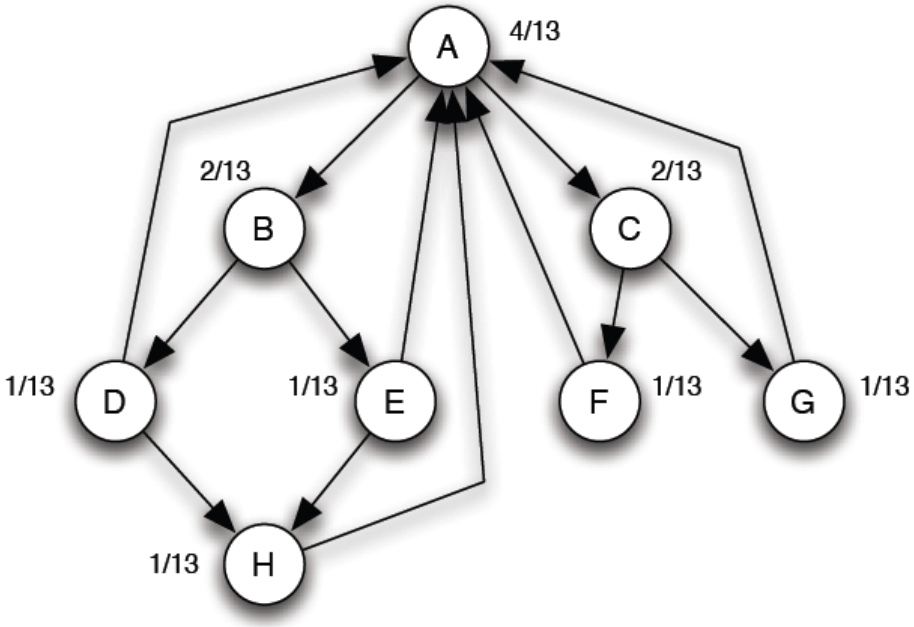
Initially, all nodes have PageRank score equal to1/8



Step	A	B	C	D	E	F	G	H
1	1/2	1/16	1/16	1/16	1/16	1/16	1/16	1/8
2	3/16	1/4	1/4	1/32	1/32	1/32	1/32	1/16

- As a kind of “fluid” that circulates through the network
- The total PageRank in the network remains constant (no need to normalize)

PageRank: Equilibrium



- A simple way to check whether an assignment of numbers forms an equilibrium set of PageRank values
  - Check that they sum to 1
  - And that when we apply the basic PageRank update rule, we get the same values back
- If the network is strongly connected, then there is a unique set of equilibrium values

Random Walks on Graphs

- The algorithm defines a random walk on the graph
- Random walk
  - Start from a node chosen uniformly at random with probability *1/n*
  - Pick one of the outgoing edges uniformly at random
  - Move to the destination of the edge
  - Repeat

The PageRank of node *v* is the probability that the random walk is at node *v* after a very large number of steps

The Random Surfer model

- Users wander on the web, following links

Random Walk - Example

- Q: What is the probability p_i^t of being at node i after t steps?

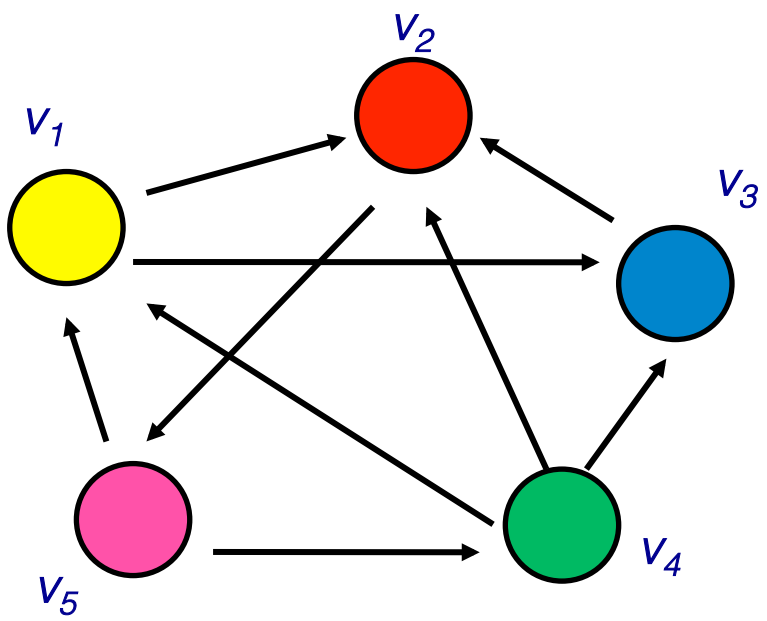
$p_1^0 = 1/5$ $\quad p_1^t = 1/3\, p_4^{t-1} + 1/2\, p_5^{t-1}$

$p_2^0 = 1/5$ $\quad p_2^t = 1/2\, p_1^{t-1} + p_3^{t-1} + 1/3\, p_4^{t-1}$

$p_3^0 = 1/5$ $\quad p_3^t = 1/2\, p_1^{t-1} + 1/3\, p_4^{t-1}$

$p_4^0 = 1/5$ $\quad p_4^t = 1/2\, p_5^{t-1}$

$p_5^0 = 1/5$ $\quad p_5^t = p_2^{t-1}$

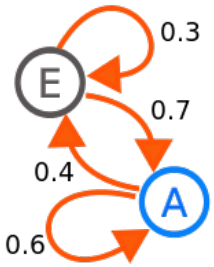


Markov Chains (1/2)

- A Markov chain describes a discrete time stochastic process over a set of states

$$S = \{s_1, s_2, \ldots, s_n\}$$

according to a transition probability matrix $P = \{P_{ij}\}$



- P_{ij} is the probability of moving to state j when at state i

- Matrix P has the property that the entries of all rows sum to 1

$$\sum_j P(i, j) = 1$$

Stochastic matrix

State probability distribution: the vector $p^t = (p_1^t, p_2^t, \ldots, p_n^t)$ that stores the probability of being at state s_i after t steps

Markov Chains (2/2)

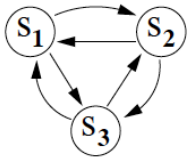
- Memorylessness property
  - The next state of the chain depends only at the current state and not on the past of the process (first order MC)
  - Higher order MCs are also possible

- Markov Chain Theory: After infinite steps the state probability vector converges to a unique distribution if the chain is
  - Irreducible (possible to get from any state to any other state)
  - and aperiodic

These are the PageRank values

Periodicity: Period of occurrence of a state

Period = 2



Aperiodic (period = 1)

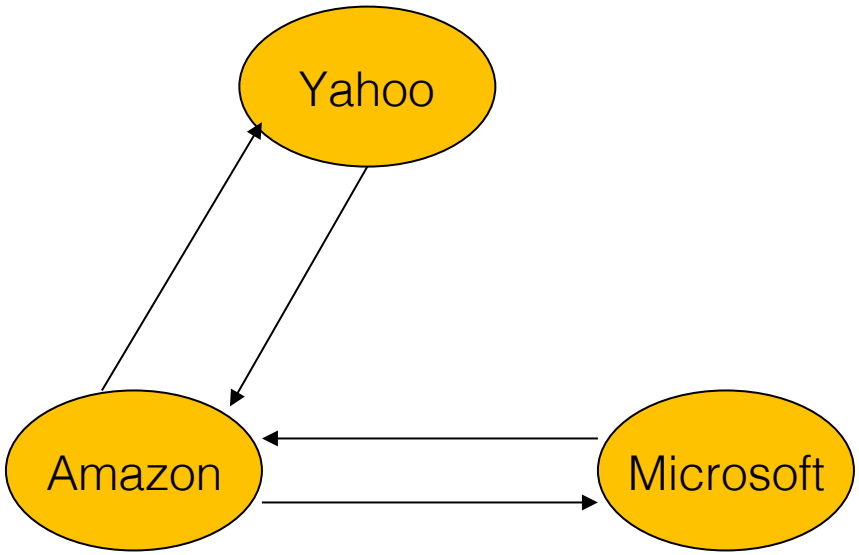
Wikipedia: https://en.wikipedia.org/wiki/Markov_chain

Random Walks

- Random walks on graphs correspond to Markov Chains
  - The set of states *S* is the set of nodes of the graph *G*
  - The transition probability matrix is the probability that we follow an edge from one node to another

$$P(i, j) = \frac{1}{k_{out}(i)}$$

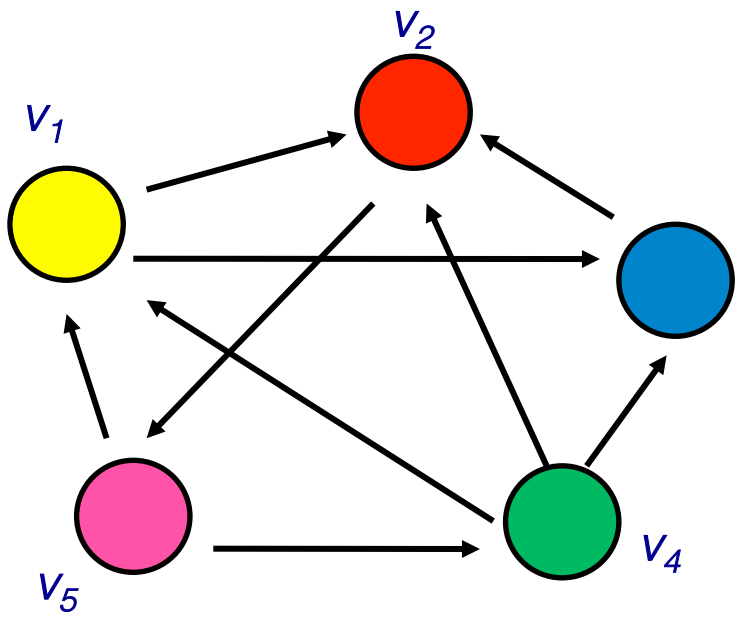
the transition matrix



An Example

the adjacency matrix

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



the transition matrix

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

Node Probability Vector

- The vector $p^t = (p_1^t, p_2^t, \ldots, p_n^t)$ that stores the probability of being at node v_i at step t

- p_i^0 = the probability of starting from state i (usually) set to uniform

- We can compute the vector p^t at step t using a vector-matrix multiplication

$$p^t = p^{t-1} P$$

The transition matrix

Stationary Distribution

- The stationary distribution of a random walk with transition matrix P, is a probability distribution π, such that

$$\pi = \pi P$$

- The stationary distribution is an eigenvector of matrix P
  - The left principal eigenvector of P
  - Stochastic matrices have maximum eigenvalue 1

- The probability π_i is the fraction of times that we visited state i as $t \rightarrow \infty$

- Markov Chain Theory:
  - The random walk converges to a unique stationary distribution independent of the initial vector, if the graph is strongly connected, and not bipartite
  - In our case these are the PageRank values

Computing the Stationary Distribution

- The Power Method

Initialize p^0 to some distribution
Repeat:

$$p^t = p^{t-1} P$$

Until convergence

- After many iterations $p^t \rightarrow \pi$ regardless of the initial vector p^0
- Rate of convergence
  - Determined by the second eigenvalue $|\lambda_2| / |\lambda_1|$

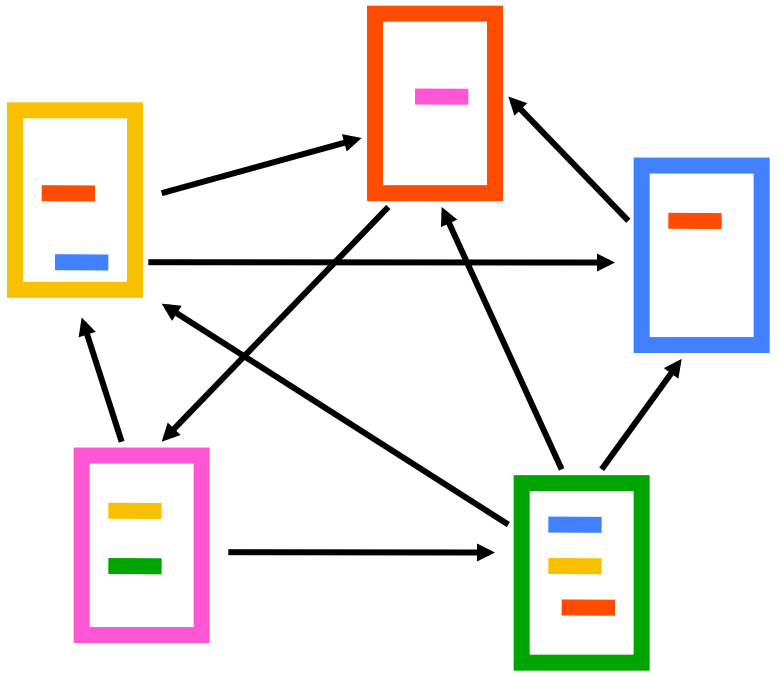
The Stationary Distribution

- What is the meaning of the stationary distribution π of a random walk?
- $\pi(i)$: the probability of being at node i after very large (infinite) number of steps
- $\pi = p_0 P^{\infty}$, where P is the transition matrix, p_0 is the original vector
  - $P(i,j)$: probability of going from i to j in one step
  - $P^2(i,j)$: probability of going from i to j in two steps (probability of all paths of length 2)
  - $P^{\infty}(i,j)$: probability of going from i to j in infinite steps – the starting point does not matter

The PageRank Random Walk

- Make the adjacency matrix stochastic and run a random walk

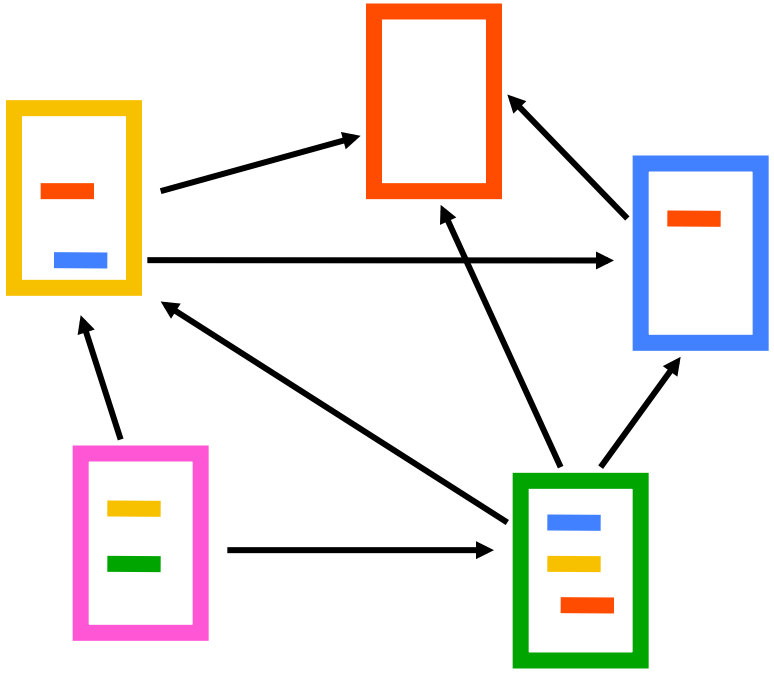
$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

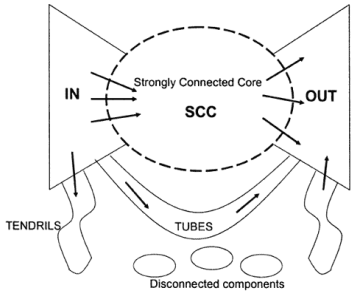


Sink Nodes (1/2)

- What about sink nodes?
  - What is happening when the random walk moves to a node without any outgoing links?

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

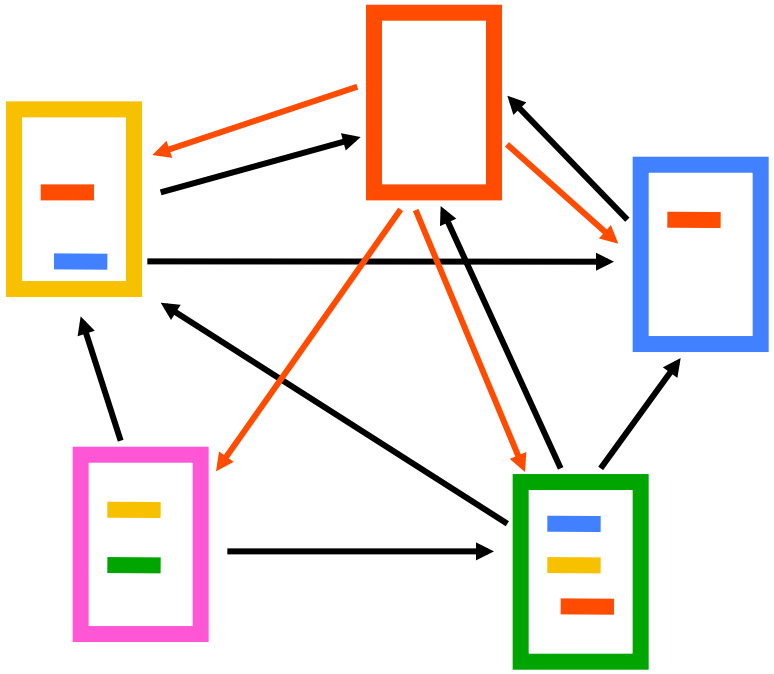




Sink Nodes (2/2)

- Replace these row vectors with a vector *v*
  - Typically, the uniform vector

$$P' = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

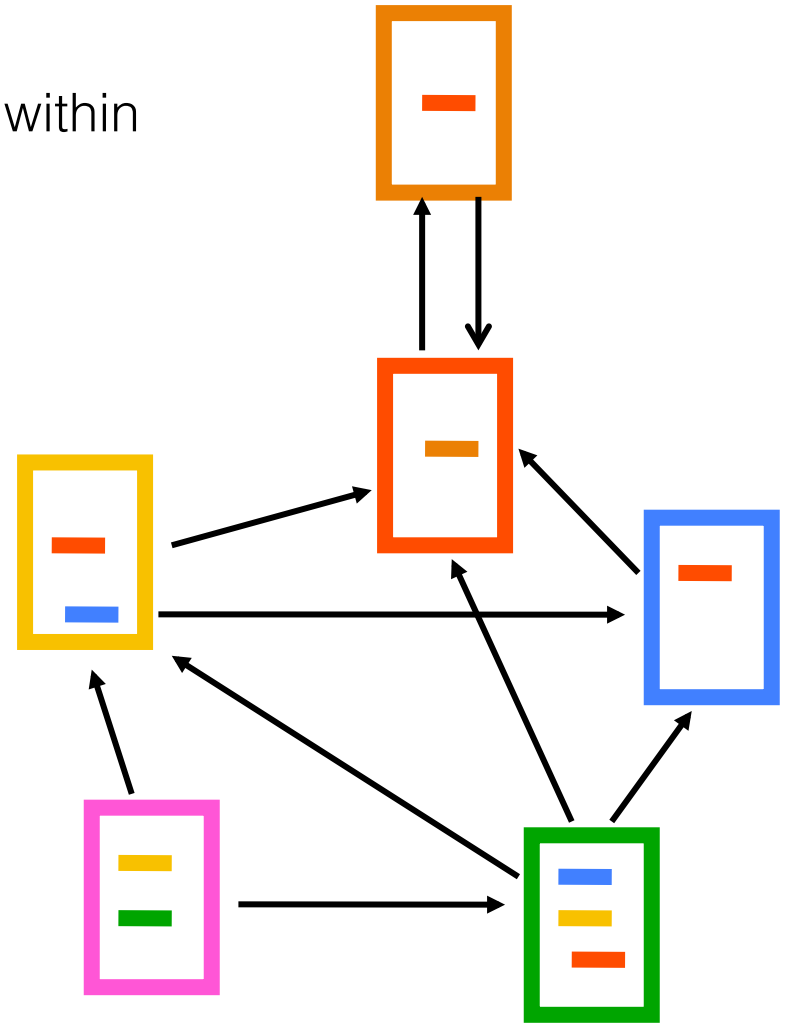


We can jump to any other node

$$P' = P + dv^T \qquad d = \begin{cases} 1 & \text{if i is sink} \\ 0 & \text{otherwise} \end{cases}$$

Spider Traps

- What about loops?
  - Spider traps: all out-links are within the group



Solution: Random Teleports

- Add a random jump to any other node (vector *v*) with prob *1-α*
  - Typically, to a uniform vector
- Restarts after *1/(1-α)* steps in expectation
  - Guarantees convergence

$$P'' = \alpha \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix} + (1-\alpha) \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}$$

With prob. **α** follow a random outgoing link

With prob. **1-α** jump to a random node

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

P'' = αP' + (1-α)uv^T, where *u* is the vector of all 1s

Google's PageRank Algorithm

- The Random Surfer model: 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

- Rank according to the stationary distribution

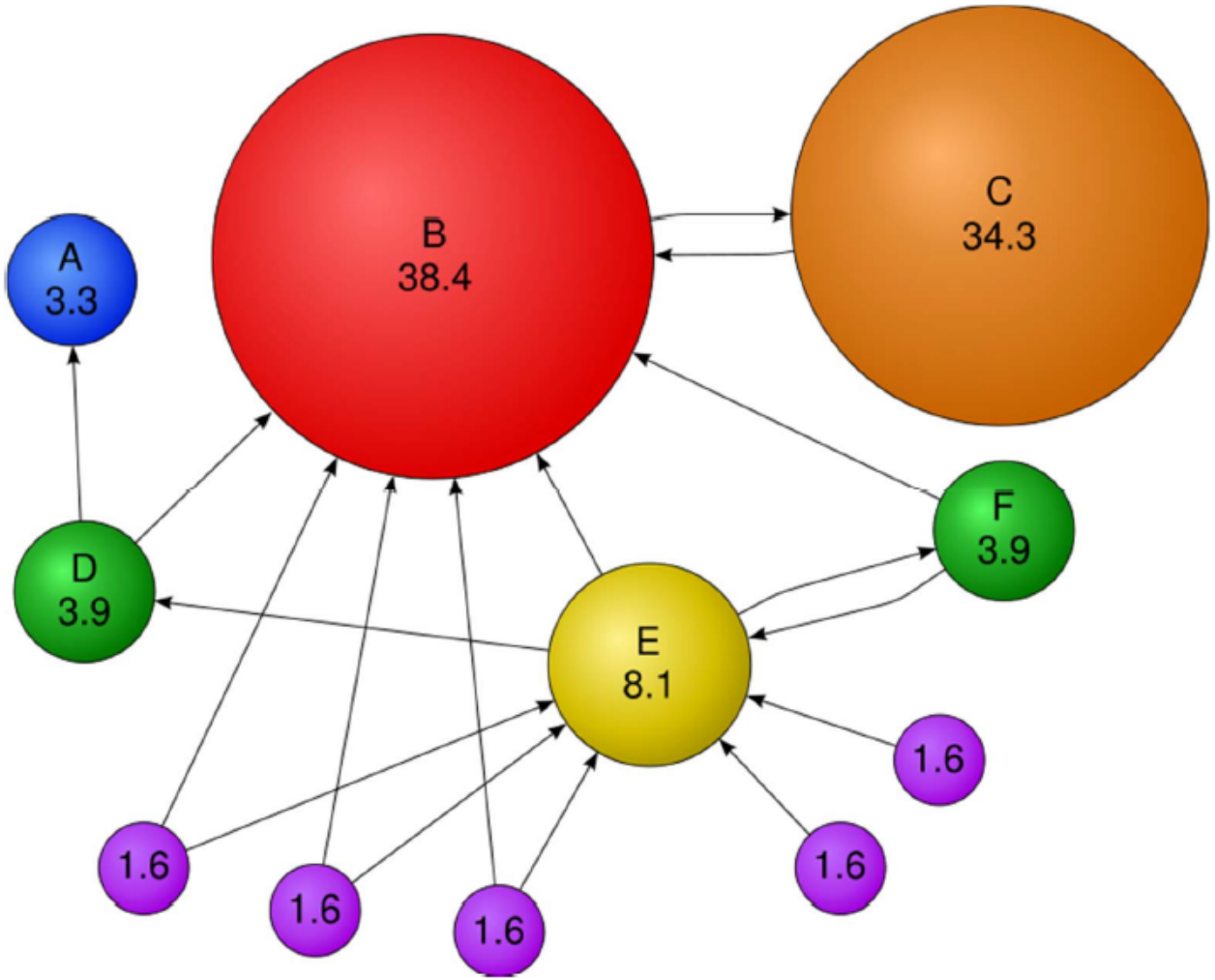
$$\pi_v = \alpha \sum_{\forall (u,v) \in E} \frac{1}{k_{out}(u)} \pi_u \quad + \quad (1-\alpha)\frac{1}{n}$$

Typically, $\alpha = 0.85$

- We repeat this computation until convergence

[Page and Brin '98]

Example



Random Walks with Restarts

- If the jump vector v is not uniform, we can bias the random walk towards the nodes that are close to v
- Personalized PageRank
  - Restart the random walk from a specific node x
  - All nodes are ranked according to their closeness to x
- Topic-Specific Pagerank
  - Restart the random walk from a specific set of nodes (e.g., nodes about a topic)
  - All nodes are ranked according to their closeness to the topic
- Random Walks with restarts is a general technique for measuring proximity on graphs

More on that at the Node Similarity lecture

PageRank History

- Huge advantage for Google in the early days
  - It gave a way to get an idea for the value of a page, which was useful in many different ways
    - Put an order to the web
  - After a while it became clear that the anchor text was probably more important for ranking
  - Also, link spam became a new (dark) art
- Flood of research
  - Numerical analysis got revived
  - Huge number of variations
  - Efficiency became a great issue
  - A plethora of applications in different fields
    - Random walk is often referred to as PageRank

Thank You!

http://www.travelandleisure.com/