Link Analysis: PageRank and HITS

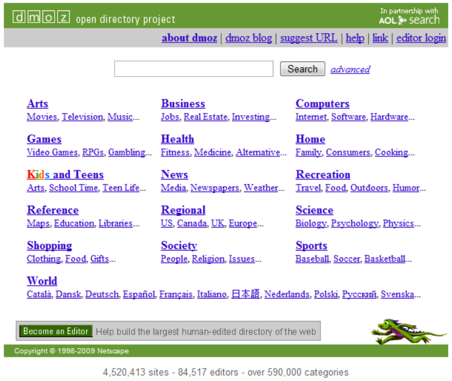
Thanks to Jure Leskovec, Stanford and Panayiotis Tsaparas, Univ. of Ioannina for slides

Agenda

- Web Search: How to Organize the Web?
- Ranking Nodes on Graphs
  - Hubs and Authorities
  - PageRank
    - How to Solve PageRank
  - Personalized PageRank
- Link Prediction in Networks
  - Estimating Scores for Missing Edges
  - Case studies:
    - Facebook: Supervised Random Walks for Link Prediction
    - Twitter: The who to follow service at Twitter

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: 2 Challenges

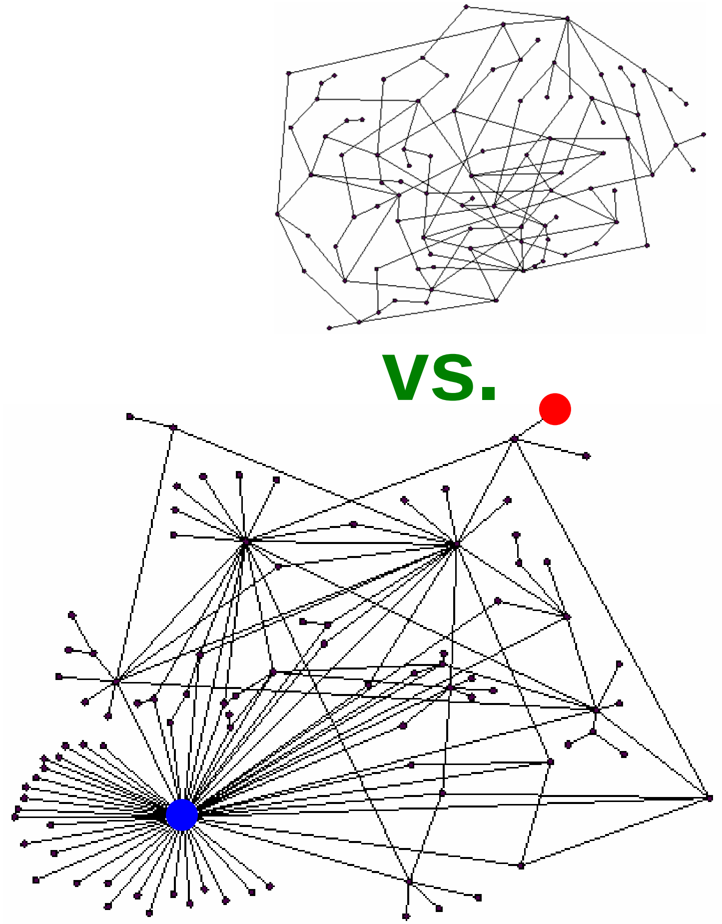
2 challenges of web search:

- (1) Web contains many sources of information
  Who 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 Nodes on the Graph

- **All web pages are not equally "important"**

  www.joe-schmoe.com vs. www.stanford.edu



- **We already know:**
  There is large diversity in the web-graph node connectivity.
- **So, let's rank the pages using the web graph link structure!**

Link Analysis Algorithms

- **We will cover the following Link Analysis approaches** to computing **importance** of nodes in a graph:
  - Hubs and Authorities (HITS)
  - Page Rank
  - Topic-Specific (Personalized) Page Rank

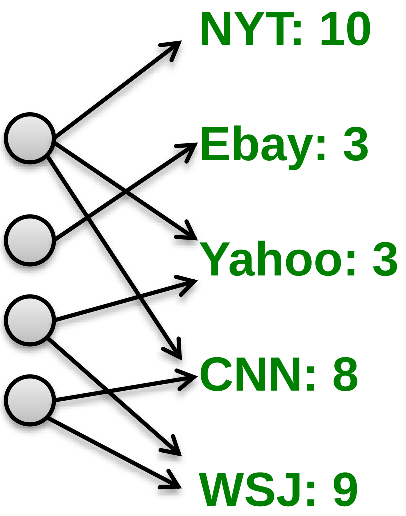
Sidenote: Various notions of **node centrality: Node u**

- **Degree centrality** = degree of u
- **Betweenness centrality** = #shortest paths passing through u
- **Closeness centrality** = avg. length of shortest paths from u to all other nodes of the network
- **Eigenvector centrality** = like PageRank

Hubs and Authorities

Link Analysis

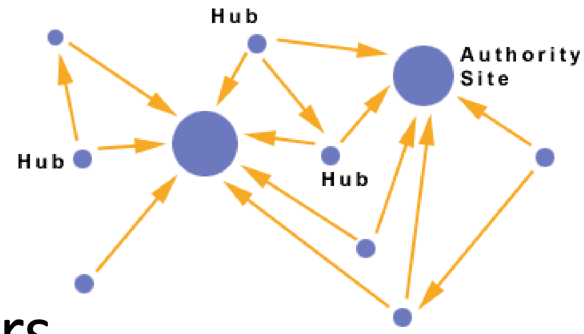
- **Goal** (back to the newspaper example):
  - Don't just find newspapers. Find "experts" – pages that link in a coordinated way to good newspapers
- **Idea: Links as votes**
  - **Page is more important if it has more links**
    - In-coming links? Out-going links?
- **Hubs and Authorities**
  Each page has **2** scores:
  - **Quality as an expert (hub):**
    - Total sum of votes of pages pointed to
  - **Quality as a content (authority):**
    - Total sum of votes of experts
  - **Principle of repeated improvement**



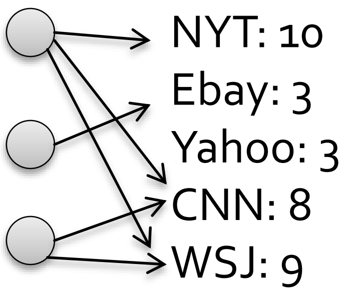
Hubs and Authorities

Interesting pages fall into two classes:

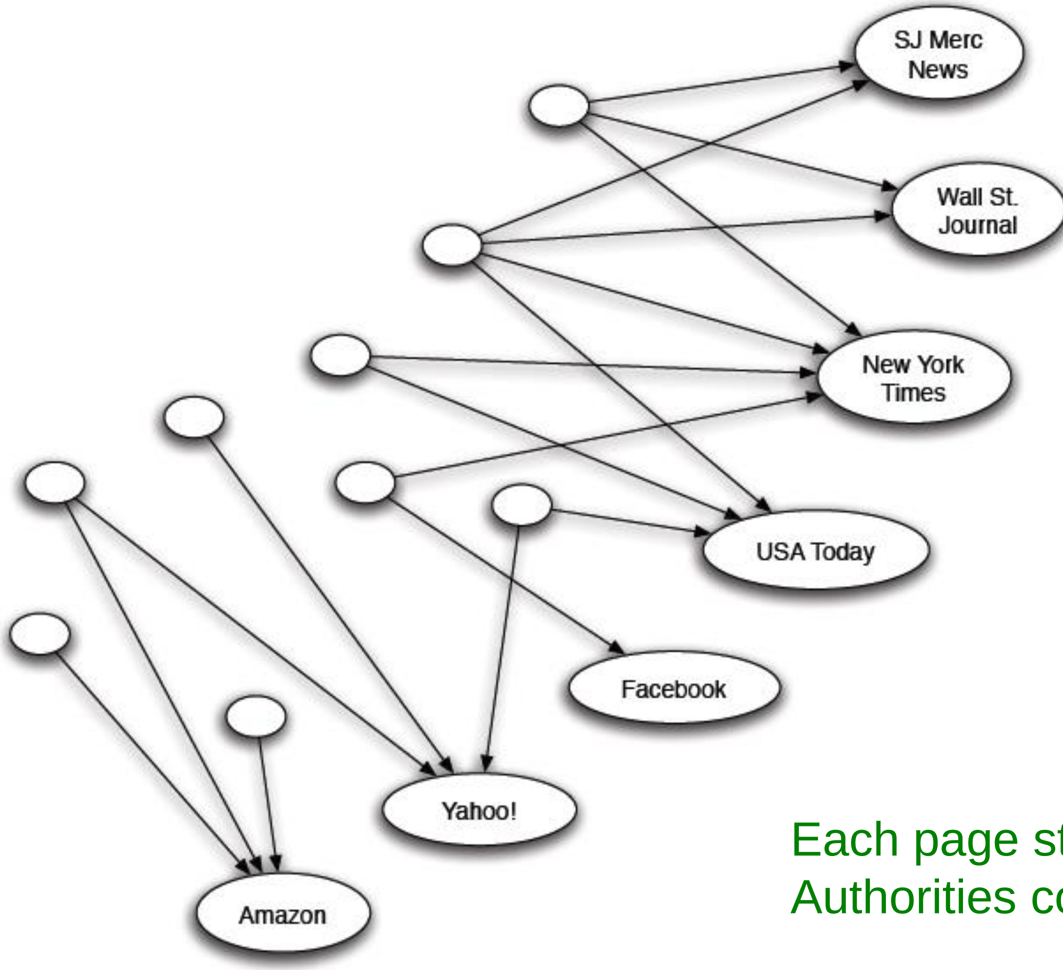
1. **Authorities** are pages containing useful information
   - Newspaper home pages
   - Course home pages
   - Home pages of auto manufacturers



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



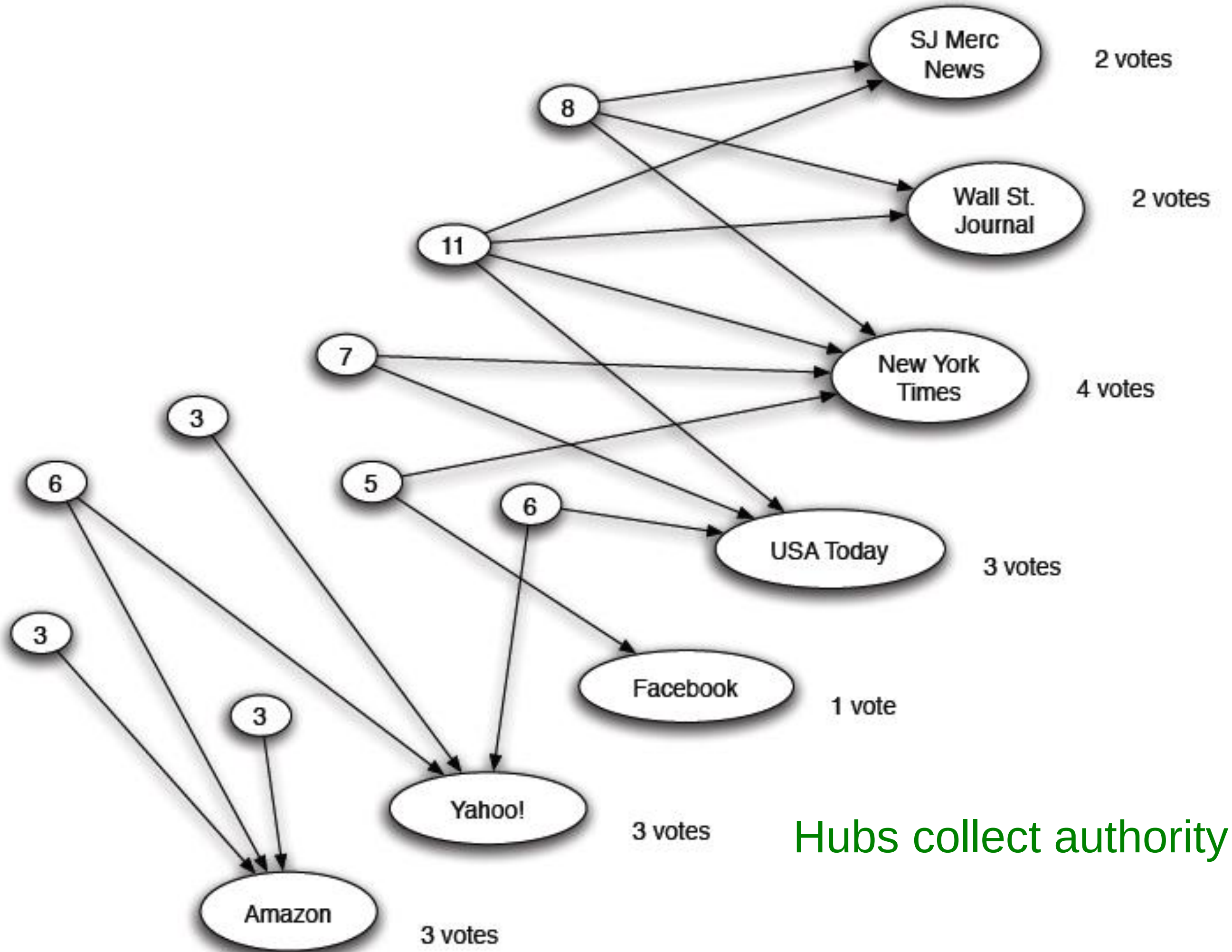
Counting in-links: Authority



Each page starts with **hub score 1**
Authorities collect their votes

(Note this is idealized example. In reality graph is not bipartite and each page has both the hub and authority score)

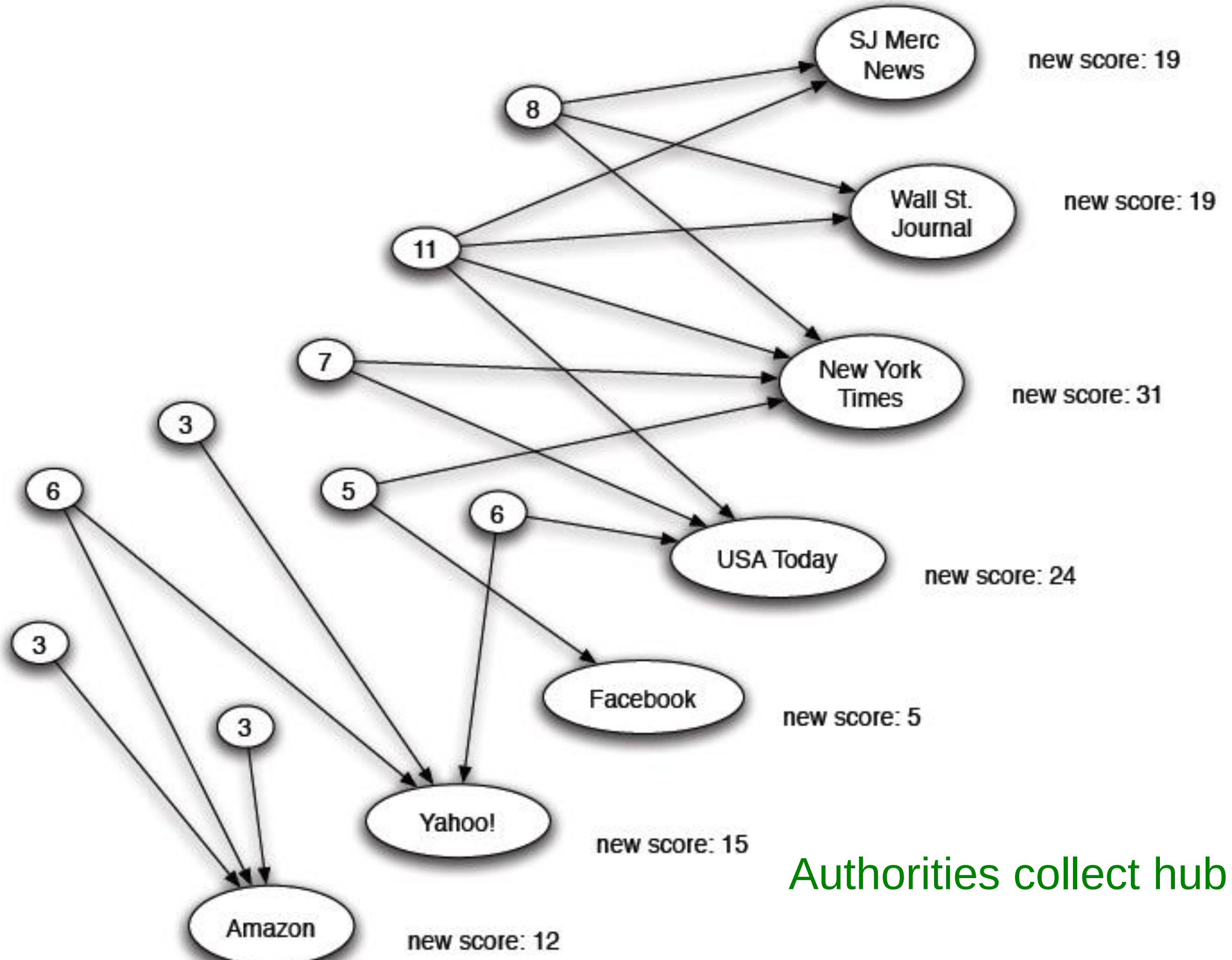
Expert Quality: Hub



Hubs collect authority scores

(Note this is idealized example. In reality graph is not bipartite and each page has both the hub and authority score)

Reweighting



Authorities collect hub scores

(Note this is idealized example. In reality graph is not bipartite and each page has both the hub and authority score)

Mutually Recursive Definition

- **A good hub links to many good authorities**
- **A good authority is linked from many good hubs**
  - Note a self-reinforcing recursive definition
- **Model using two scores for each node:**
  - **Hub** score and **Authority** score
  - Represented as vectors $\boldsymbol{h}$ and $\boldsymbol{a}$, where the ***i*-th** element is the hub/authority score of the ***i*-th** node

[Kleinberg '98]

Hubs and Authorities

- **Each page i has 2 scores:**
  - Authority score: $\boldsymbol{a_i}$
  - Hub score: $\boldsymbol{h_i}$

Convergence criteria:

$$\sum_i \left(h_i^{(t)} - h_i^{(t+1)}\right)^2 < \varepsilon$$

$$\sum_i \left(a_i^{(t)} - a_i^{(t+1)}\right)^2 < \varepsilon$$

HITS algorithm:

- Initialize: $a_j^{(0)} = 1/\sqrt{\mathrm{n}},\ \ \mathrm{h}_\mathrm{j}^{(0)} = 1/\sqrt{\mathrm{n}}$
- Then keep iterating until **convergence**:
  - $\forall \boldsymbol{i}$: Authority: $a_i^{(t+1)} = \sum_{j \to i} h_j^{(t)}$
  - $\forall \boldsymbol{i}$: Hub: $h_i^{(t+1)} = \sum_{i \to j} a_j^{(t)}$
  - $\forall \boldsymbol{i}$: Normalize:

$$\sum_i \left(a_i^{(t+1)}\right)^2 = 1, \sum_j \left(h_j^{(t+1)}\right)^2 = 1$$

Hubs and Authorities

- **Definition:** **Eigenvectors & Eigenvalues**
- Let $\boldsymbol{R} \cdot \boldsymbol{x} = \boldsymbol{\lambda} \cdot \boldsymbol{x}$
  for some scalar $\boldsymbol{\lambda}$, vector $\boldsymbol{x}$, matrix $\boldsymbol{R}$
  - Then $\boldsymbol{x}$ **is an eigenvector**, and $\boldsymbol{\lambda}$ is its **eigenvalue**
- **The steady state (HITS has converged) is:**
  - $\boldsymbol{A^T \cdot A \cdot a = c' \cdot a}$
  - $\boldsymbol{A \cdot A^T \cdot h = c'' \cdot h}$

Note constants *c'*,*c"* don't matter as we normalize them out every step of HITS

- **So,** **authority** $\boldsymbol{a}$ is eigenvector of $\boldsymbol{A^T A}$
  (associated with the largest eigenvalue)
  Similarly: **hub** $\boldsymbol{h}$ is eigenvector of $\boldsymbol{AA^T}$

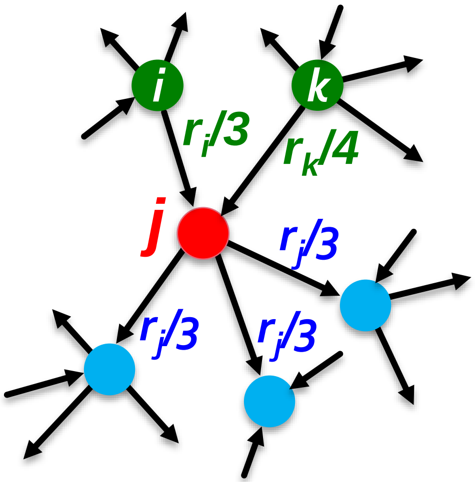
PageRank

Links as Votes

- **Still the same idea: Links as votes**
  - **Page is more important if it has more links**
    - In-coming links? Out-going links?
- **Think of in-links as votes:**
  - www.stanford.edu (many in-links)
  - www.edessacity.gr (few in-link)
- **Are all in-links equal?**
  - Links from important pages count more
  - Recursive question!

PageRank: The "Flow" Model

- **A "vote" from an important page is worth more:**
  - Each link's vote is proportional to the **importance** of its source page
  - If page ***i*** with importance r_i has d_i out-links, each link gets r_i / d_i votes
  - Page ***j***'s own importance r_j is the sum of the votes on its in-links



$$r_j = r_i/3 + r_k/4$$

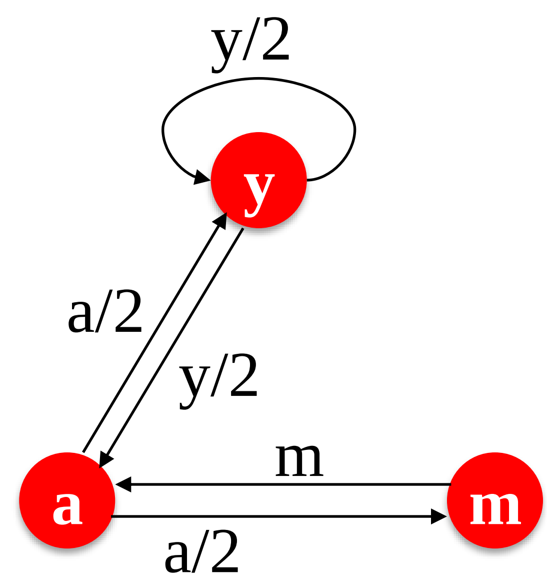
PageRank: The "Flow" Model

- **A page is important if it is pointed to by other important pages**
- **Define a "rank" r_j for node j**

$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_i}$$

d_i **… out-degree of node i**

The web in 1839



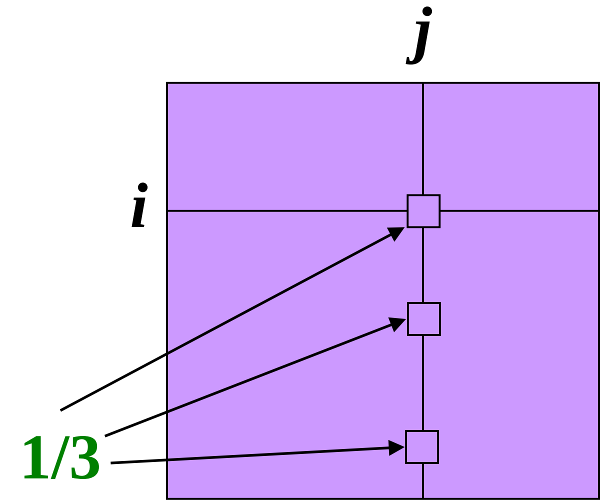
"Flow" equations:

$$\mathbf{r_y = r_y/2 + r_a/2}$$
$$\mathbf{r_a = r_y/2 + r_m}$$
$$\mathbf{r_m = r_a/2}$$

You might wonder: Let's just use Gaussian elimination to solve this system of linear equations. Bad idea!

PageRank: Matrix Formulation

- **Stochastic adjacency matrix $\boldsymbol{M}$**
  - Let page $\boldsymbol{j}$ have $\boldsymbol{d_j}$ out-links
  - If $\boldsymbol{j} \rightarrow \boldsymbol{i}$, then $\boldsymbol{M_{ij}} = \frac{1}{d_j}$
    - $\boldsymbol{M}$ is a **column stochastic matrix**
      - **Columns** sum to **1**



M

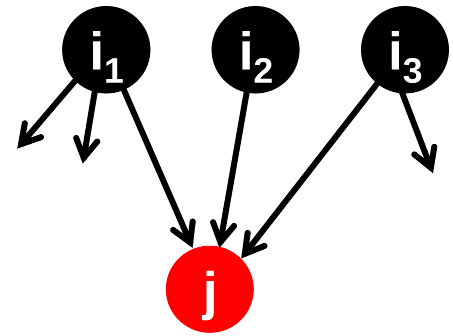
- **Rank vector $\boldsymbol{r}$:** An entry per page
  - $\boldsymbol{r_i}$ is the importance score of page $\boldsymbol{i}$
  - $\sum_i \boldsymbol{r_i} = \mathbf{1}$
- **The flow equations can be written**

$$\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$$

$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_i}$$

Random Walk Interpretation

- **Imagine a random web surfer:**
  - At any time $\boldsymbol{t}$, surfer is on some page i
  - At time $\boldsymbol{t+1}$, the surfer follows an out-link from $\boldsymbol{i}$ uniformly at random
  - Ends up on some page $\boldsymbol{j}$ linked from $\boldsymbol{i}$
  - Process repeats indefinitely
- **Let:**
  - $\boldsymbol{p}(\boldsymbol{t})$ … vector whose i^{th} coordinate is the prob. that the surfer is at page i at time t
  - So, $\boldsymbol{p}(\boldsymbol{t})$ is a probability distribution over pages

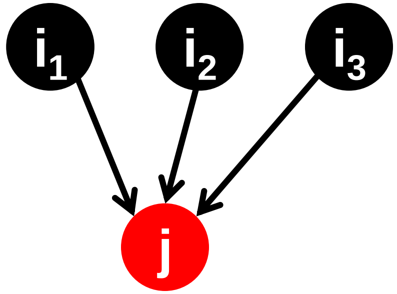


$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_{\text{out}}(\mathrm{i})}$$

The Stationary Distribution

- **Where is the surfer at time *t+1*?**
  - Follows a link uniformly at random

    $$\boldsymbol{p}(\boldsymbol{t}+1) = \boldsymbol{M} \cdot \boldsymbol{p}(\boldsymbol{t})$$



$$p(t+1) = \mathrm{M} \cdot p(t)$$

- Suppose the random walk reaches a state

  $$\boldsymbol{p}(\boldsymbol{t}+1) = \boldsymbol{M} \cdot \boldsymbol{p}(\boldsymbol{t}) = \boldsymbol{p}(\boldsymbol{t})$$

  then $\boldsymbol{p}(t)$ is **stationary distribution** of a random walk
- **Our original rank vector** $\boldsymbol{r}$ satisfies $\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$
  - **So, r is a stationary distribution for the random walk**

PageRank: How to solve?

PageRank: How to solve?

Given a web graph with *n* nodes, where the nodes are pages and edges are hyperlinks

- Assign each node an initial page rank
- Repeat until convergence ($\Sigma_i \, |r_i^{(t+1)} - r_i^{(t)}| < \varepsilon$)
  - Calculate the page rank of each node

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

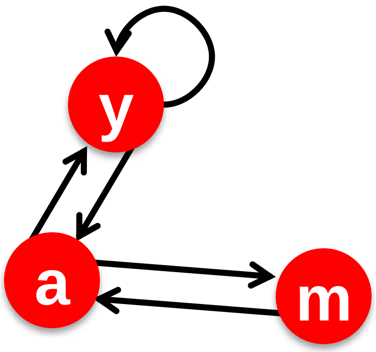
d_i …. out-degree of node i

PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j \leftarrow 1/N$
  - **1:** $r'_j \leftarrow \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r \leftarrow r'$
  - If $|r - r'| > \varepsilon$: goto **1**
- **Example:**

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$

Iteration 0, 1, 2, …



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

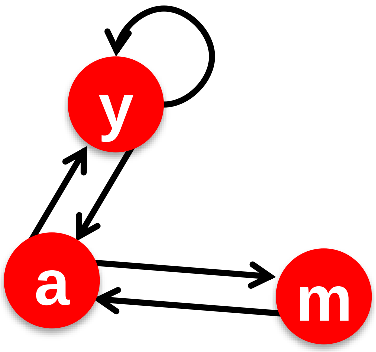
$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j \leftarrow 1/N$
  - **1:** $r'_j \leftarrow \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r \leftarrow r'$
  - If $|r - r'| > \varepsilon$: goto **1**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$\mathbf{r_y = r_y/2 + r_a/2}$

$\mathbf{r_a = r_y/2 + r_m}$

$\mathbf{r_m = r_a/2}$

- **Example:**

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 1/3 & 5/12 & 9/24 & & 6/15 \\ 1/3 & 3/6 & 1/3 & 11/24 & \ldots & 6/15 \\ 1/3 & 1/6 & 3/12 & 1/6 & & 3/15 \end{matrix}$$

Iteration 0, 1, 2, …

PageRank: Three Questions

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

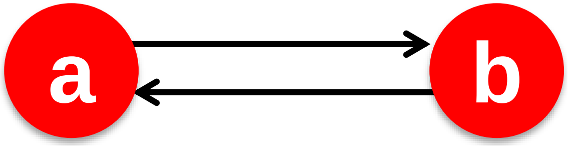
or equivalently

$$r = Mr$$

- **Does this converge?**
- **Does it converge to what we want?**
- **Are results reasonable?**

Does this converge?

- **The "Spider trap" problem:**



$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

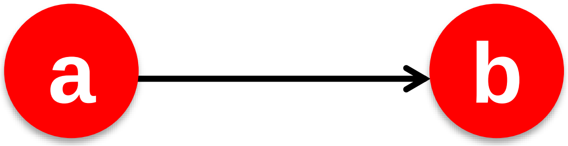
- **Example:**

Iteration: 0, 1, 2, 3…

$$\begin{array}{c} \mathrm{r_a} \\ \mathrm{r_b} \end{array} = \begin{array}{c|c|c|c} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}$$

Does it converge to what we want?

- **The "Dead end" problem:**

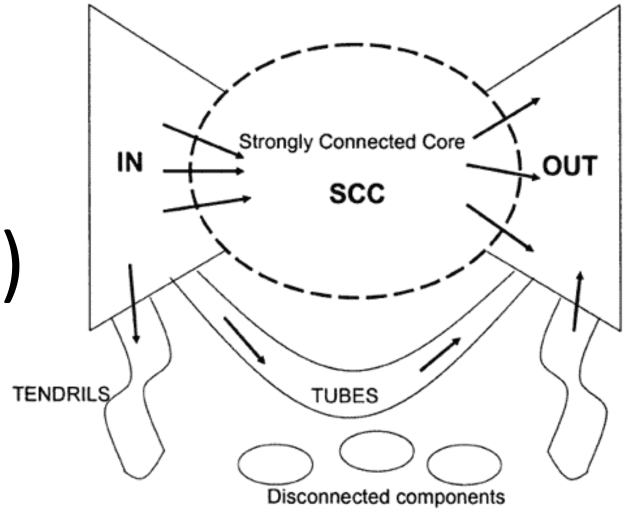


$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_\mathrm{i}}$$

- **Example:**

	Iteration:	0,	1,	2,	3…
r_a	=	1	0	0	0
r_b		0	1	0	0

RageRank: Problems

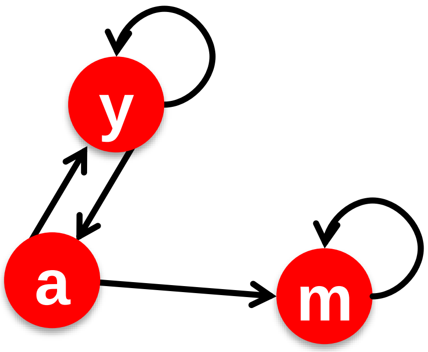


2 problems:

- **(1)** Some pages are **dead ends** (have no out-links)
  - Such pages cause importance to "leak out"
- **(2)** **Spider traps** (all out-links are within the group)
  - Eventually spider traps absorb all importance

Problem: Spider Traps

- **Power Iteration:**
  - Set $r_j = \frac{1}{N}$
  - $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
    - And iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$$\mathbf{r_y = r_y/2 + r_a/2}$$
$$\mathbf{r_a = r_y/2}$$
$$\mathbf{r_m = r_a/2 + r_m}$$

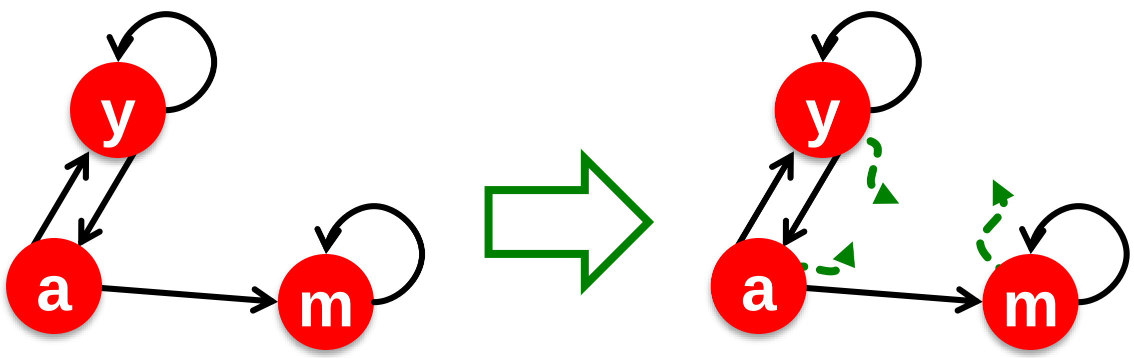
- **Example:**

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{array}{c|c|c|c} 1/3 & 2/6 & 3/12 & 5/24 \\ 1/3 & 1/6 & 2/12 & 3/24 \\ 1/3 & 3/6 & 7/12 & 16/24 \end{array} \quad \ldots \quad \begin{array}{c} 0 \\ 0 \\ 1 \end{array}$$

Iteration 0, 1, 2, …

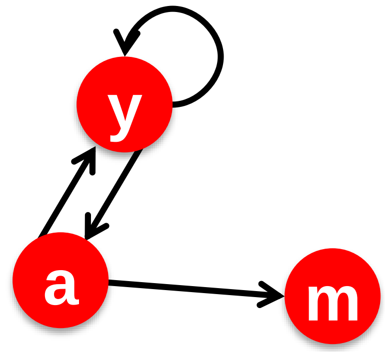
Solution: Random Teleports

- **The Google solution for spider traps: At each time step, the random surfer has two options**
  - With prob. β, follow a link at random
  - With prob. **1-β,** jump to a random page
  - Common values for β are in the range 0.8 to 0.9
- **Surfer will teleport out of spider trap within a few time steps**



Problem: Dead Ends

- **Power Iteration:**
  - Set $r_j = \frac{1}{N}$
  - $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
    - And iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2$$
$$r_m = r_a/2$$

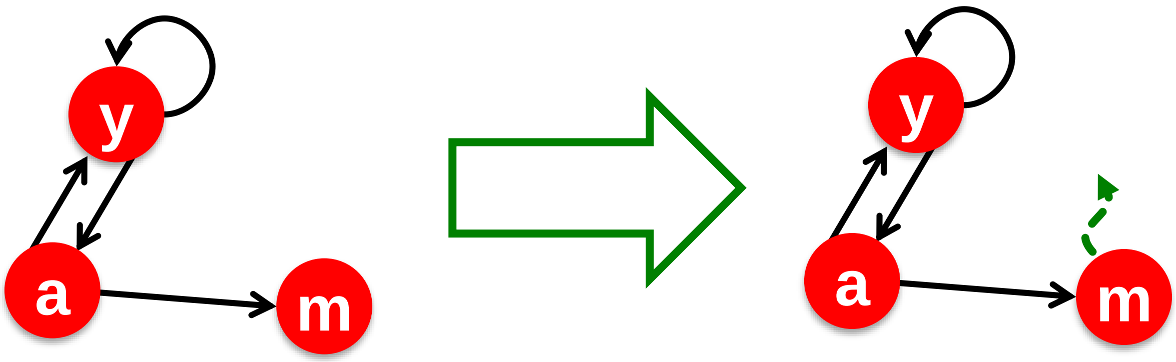
- **Example:**

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{array}{c|c|c|c} 1/3 & 2/6 & 3/12 & 5/24 \\ 1/3 & 1/6 & 2/12 & 3/24 \\ 1/3 & 1/6 & 1/12 & 2/24 \end{array} \quad \begin{array}{c} \\ \dots \\ \\ \end{array} \quad \begin{array}{c} 0 \\ 0 \\ 0 \end{array}$$

Iteration 0, 1, 2, …

Solution: Always Teleport

- **Teleports:** Follow random teleport links with probability **1.0** from dead-ends
  - 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

Final PageRank Equation

- **Google's solution: At each step, random surfer has two options:**
  - With probability β, follow a link at random
  - With probability $1\text{-}\beta$, jump to some random page
- **PageRank equation** [Brin-Page, '98]

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{n}$$

d_i ... out-degree of node i

The above formulation assumes that M has no dead ends. We can either preprocess matrix M **(bad!)** or explicitly follow random teleport links with probability 1.0 from dead-ends. See P. Berkhin, *A Survey on PageRank Computing*, Internet Mathematics, 2005.

PageRank & Eigenvectors

Details!

- **PageRank as a principal eigenvector**

  $\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$ or equivalently $\boldsymbol{r_j} = \sum_i \frac{\boldsymbol{r_i}}{\boldsymbol{d_i}}$

- **But we really want (**):**

  $$\boldsymbol{r_j} = \boldsymbol{\beta} \sum_i \frac{\boldsymbol{r_i}}{\boldsymbol{d_i}} + (\mathbf{1} - \boldsymbol{\beta}) \frac{\mathbf{1}}{\boldsymbol{n}}$$

- **Let's define:**

  $$\boldsymbol{M'_{ij}} = \boldsymbol{\beta}\, \boldsymbol{M_{ij}} + (\mathbf{1} - \boldsymbol{\beta}) \frac{\mathbf{1}}{\boldsymbol{n}}$$

- **Now we get what we want:**

  $$\boldsymbol{r} = \boldsymbol{M'} \cdot \boldsymbol{r}$$

- **What is 1 – β?**
  - In practice *0.15* (Jump approx. every *5-6* links)

d_i … out-degree of node i

Note: M is a sparse matrix but $\boldsymbol{M'}$ is dense (all entries $\neq 0$). In practice we never "materialize" M but rather we use the "sum" formulation (**)

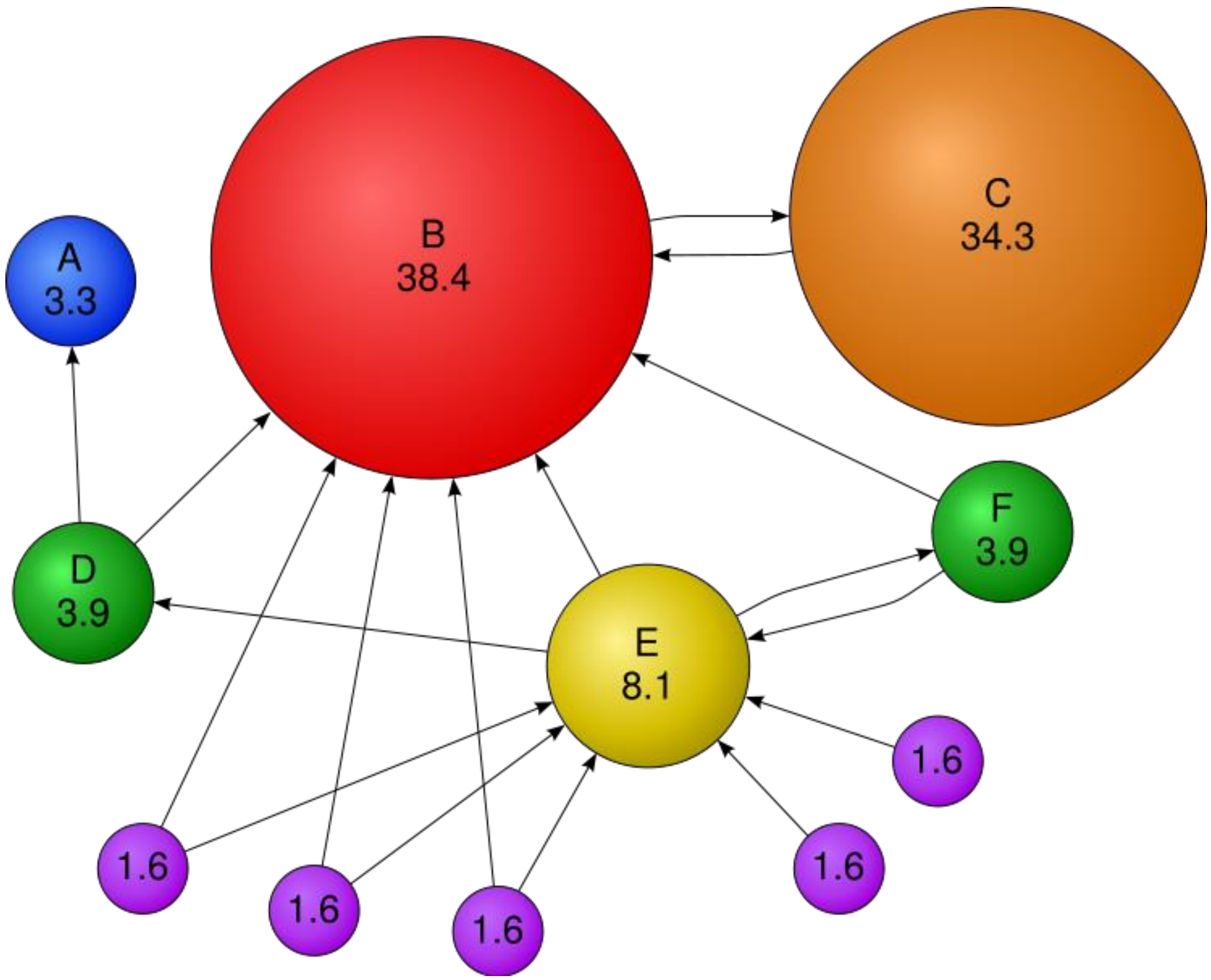
The PageRank Algorithm

- **Input: Graph G and parameter β**
  - Directed graph G with **spider traps** and **dead ends**
  - Parameter β
- **Output: PageRank vector r**
  - **Set:** $r_j^{(0)} = \frac{1}{N}, \quad t = 1$
  - **do:**
    - $\forall j: \ \boldsymbol{r'}_j^{(t)} = \sum_{i \to j} \boldsymbol{\beta} \frac{r_i^{(t-1)}}{d_i}$

      $\boldsymbol{r'}_j^{(t)} = \mathbf{0}$ if in-deg. of $\boldsymbol{j}$ is **0**
    - **Now re-insert the leaked PageRank:**

      $\forall \boldsymbol{j}: \ \boldsymbol{r}_j^{(t)} = \boldsymbol{r'}_j^{(t)} + \frac{\mathbf{1}-\boldsymbol{S}}{\boldsymbol{N}}$ **where:** $S = \sum_j r'^{(t)}_j$
    - $\boldsymbol{t} = \boldsymbol{t} + \mathbf{1}$
  - **while** $\sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| > \varepsilon$

Example



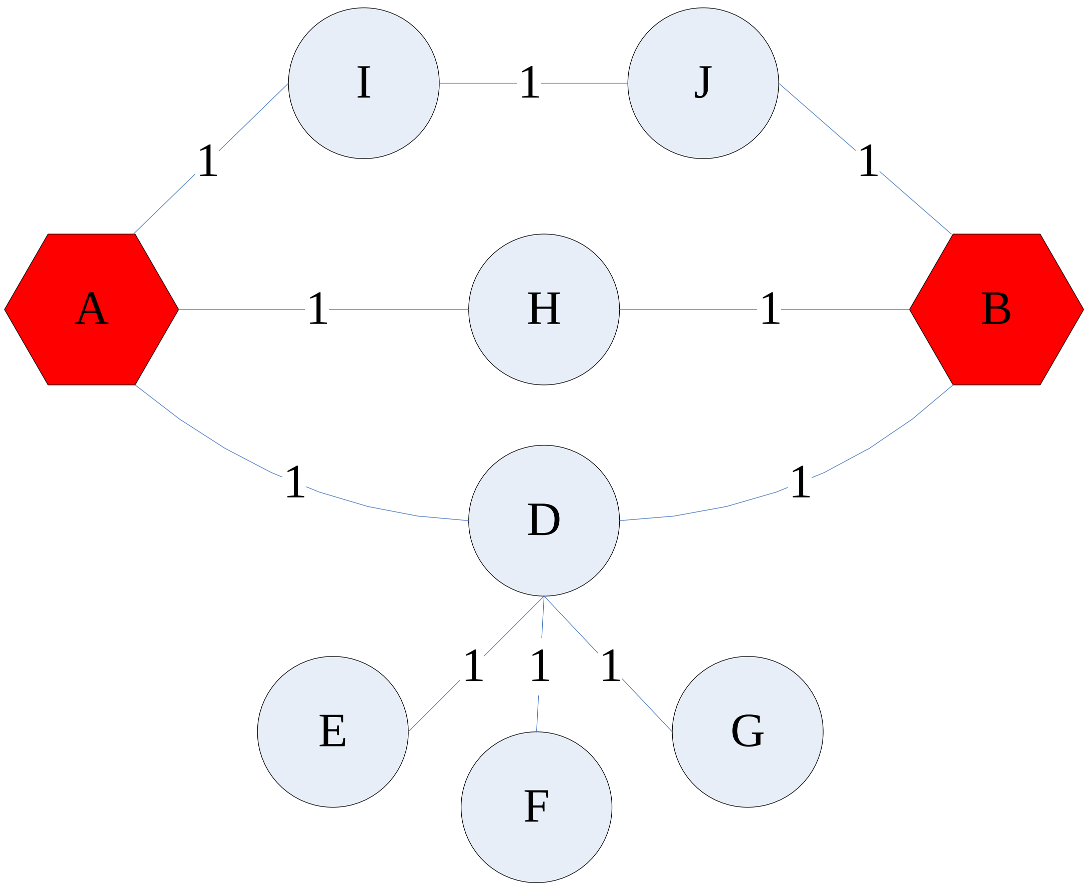
PageRank and HITS

- **PageRank and HITS are two solutions to the same problem:**
  - **What is the value of an in-link from *u* to *v*?**
  - In the PageRank model, the value of the link depends on the links **into** *u*
  - In the HITS model, it depends on the value of the other links **out of** *u*
- **The destinies of PageRank and HITS post-1998 were very different**

Personalized PageRank, Random Walk with Restarts

Proximity on Graphs

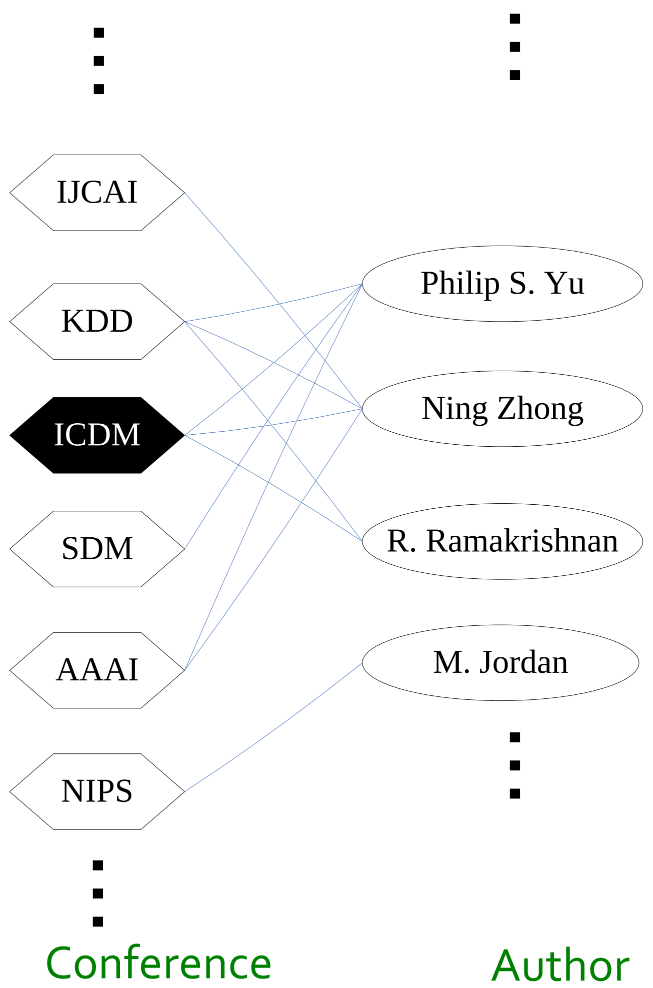
[Tong-Faloutsos, '06]



a.k.a.: Relevance, Closeness, 'Similarity'…

Example Application: Graph Search

- **Given:**
  Conferences-to-authors graph
- **Goal:**
  Proximity on graphs
  - Q: What is most related conference to ICDM?



[Tong et al. '08]

Automatic Image Captioning

{Sea Sun Sky Wave}

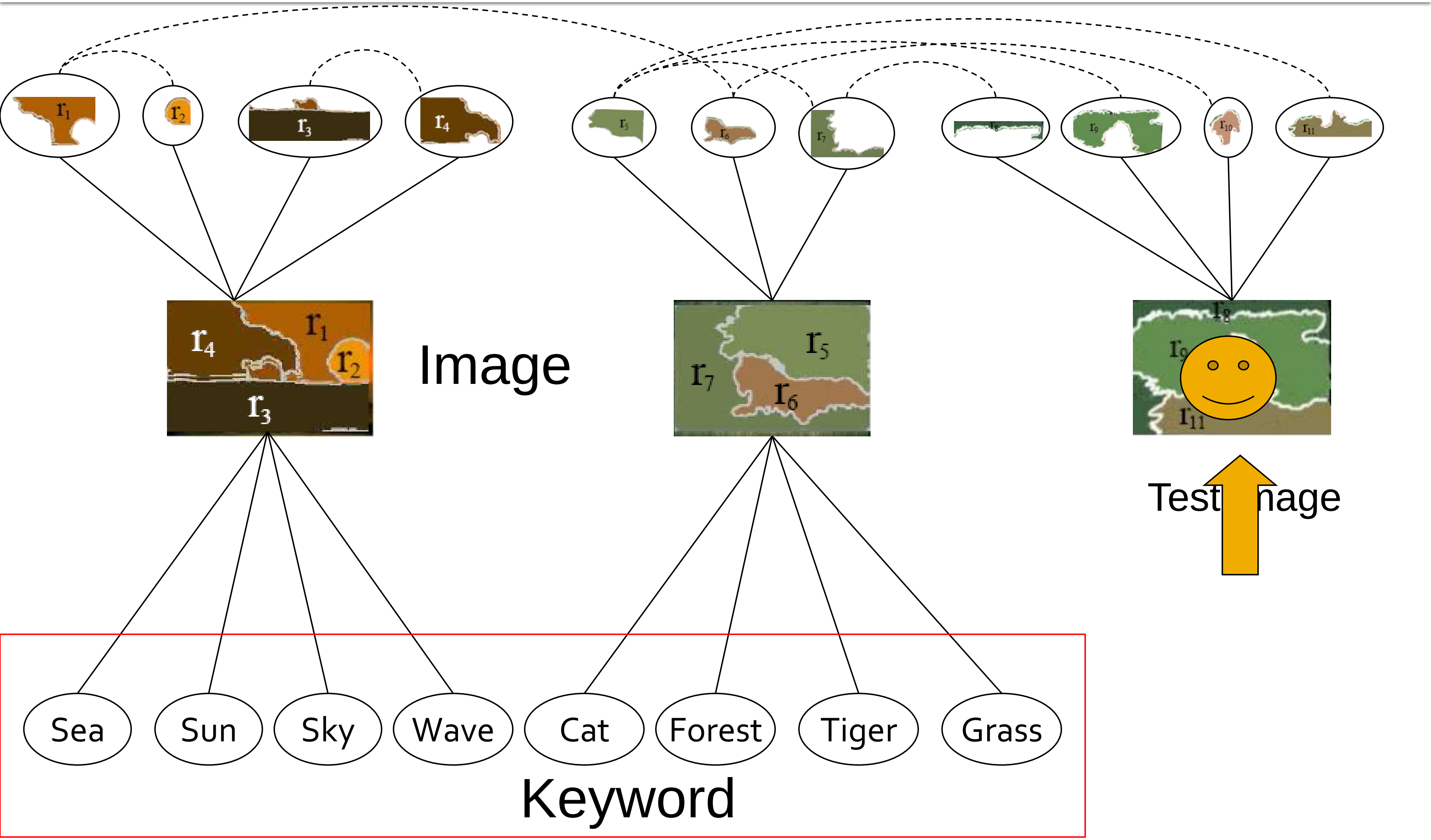
…

{Cat Forest Grass Tiger}

?

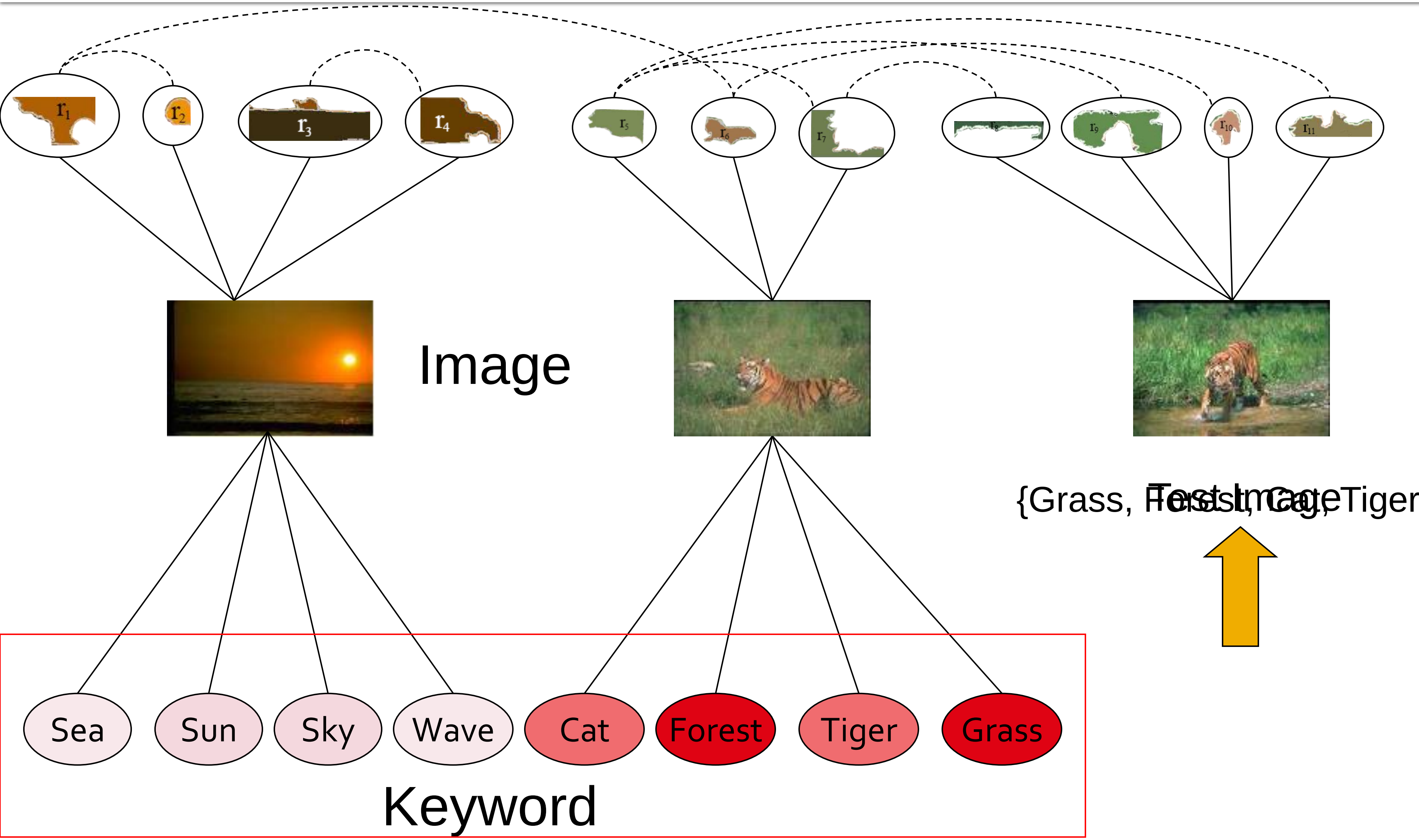
{?, ?, ?,}

[Tong et al. '08]



r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
Image
Test Image
Sea
Sun
Sky
Wave
Cat
Forest
Tiger
Grass
Keyword

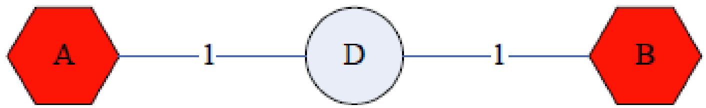
[Tong et al. ‘08]

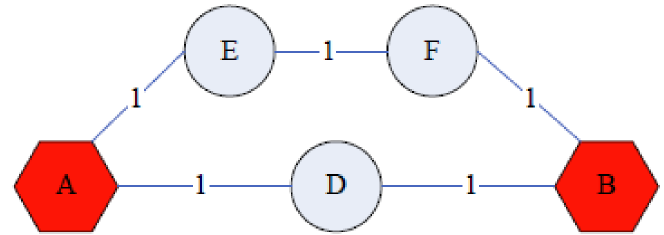


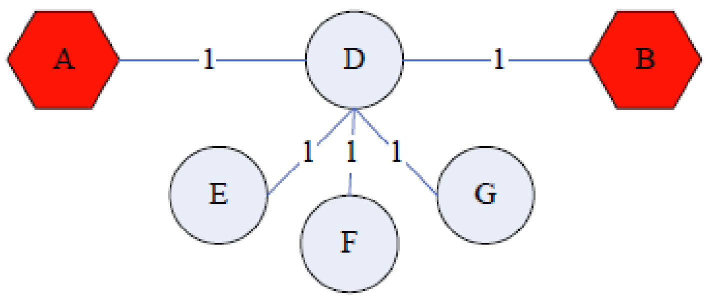
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
Image
Test Image
{Grass, Forest, Cat, Tiger}
Sea
Sun
Sky
Wave
Cat
Forest
Tiger
Grass
Keyword

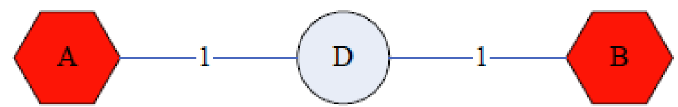
Good proximity measure?

- **Shortest path is not good:**





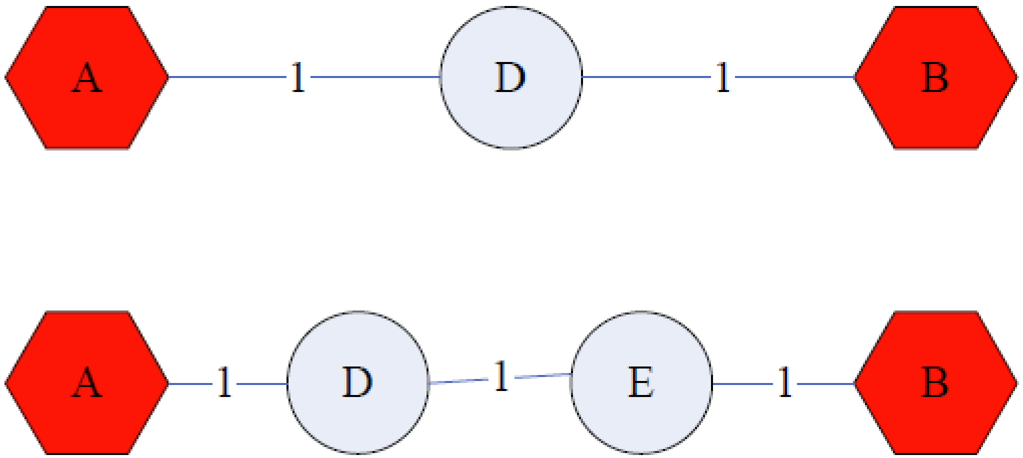




- **No influence for degree-1 nodes (E, F, G)!**
- Multi-faceted relationships

Good proximity measure?

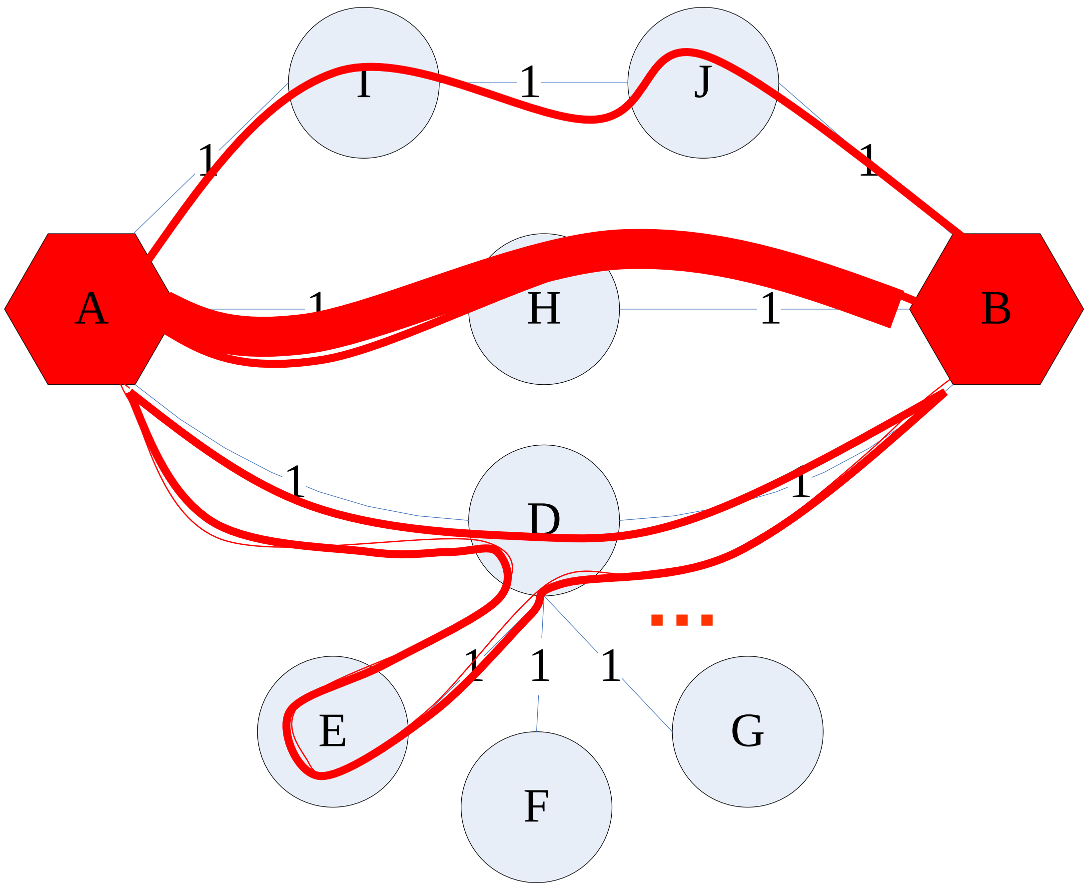
- Network Flow is not good:



- Does not punish long paths

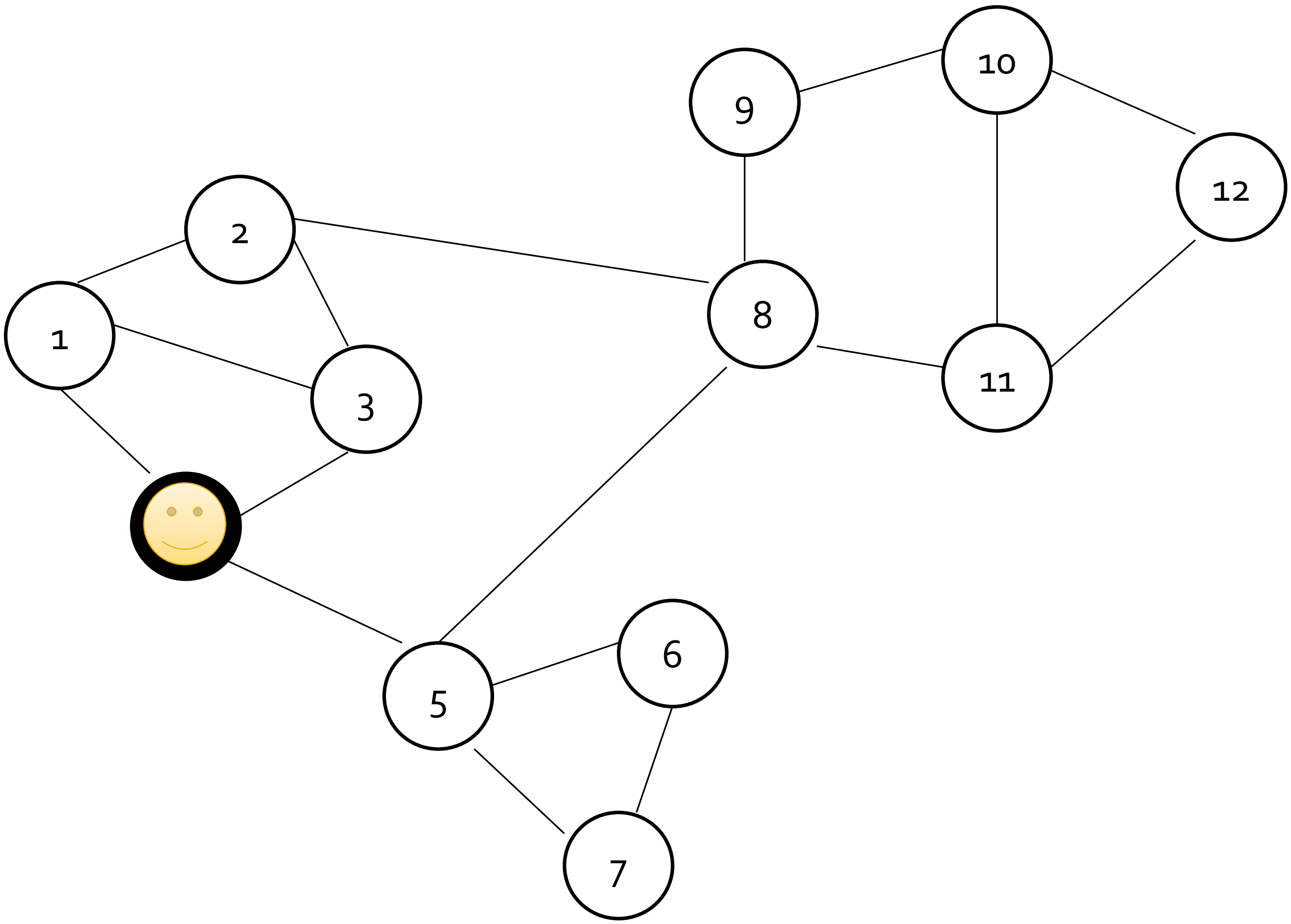
[Tong-Faloutsos, '06]

What is good notion of proximity?



- **Multiple Connections**
- **Quality of connection**
  - **Direct & In-direct connections**
  - **Length, Degree, Weight…**

Random Walk with Restarts



Personalized PageRank

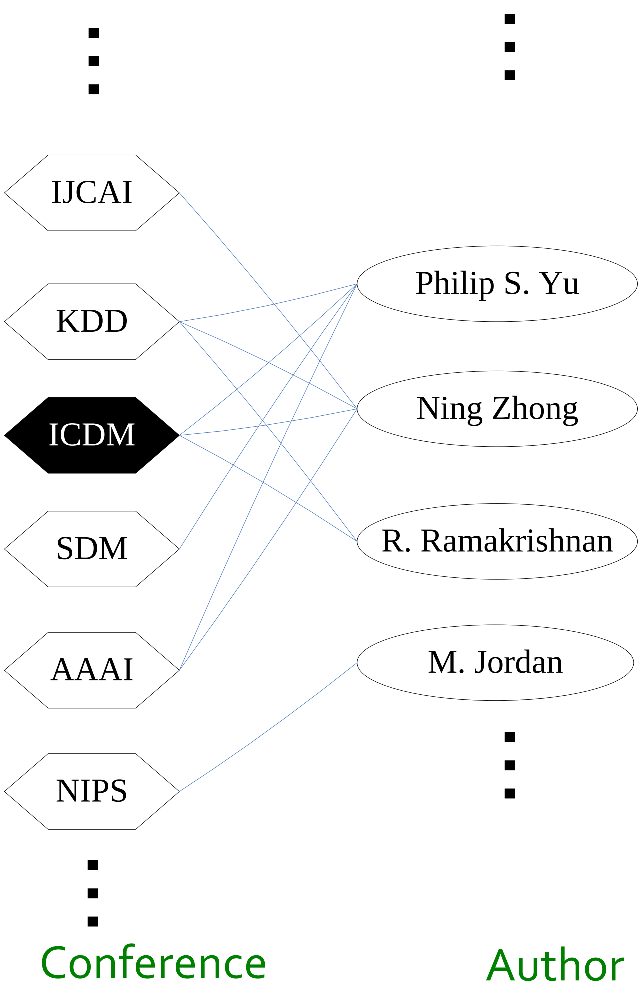
- **Goal:** Evaluate pages not just by popularity but by how close they are to the topic
- **Teleporting can go to:**
  - **Any page with equal probability**
    - (we used this so far)
  - **A topic-specific set of "relevant" pages**
    - Topic-specific (personalized) PageRank (***S*** ...teleport set)

$$M'_{ij} = \beta M_{ij} + (1-\beta)/|S| \quad \text{if } i \in S$$
$$= \beta M_{ij} \quad \text{otherwise}$$

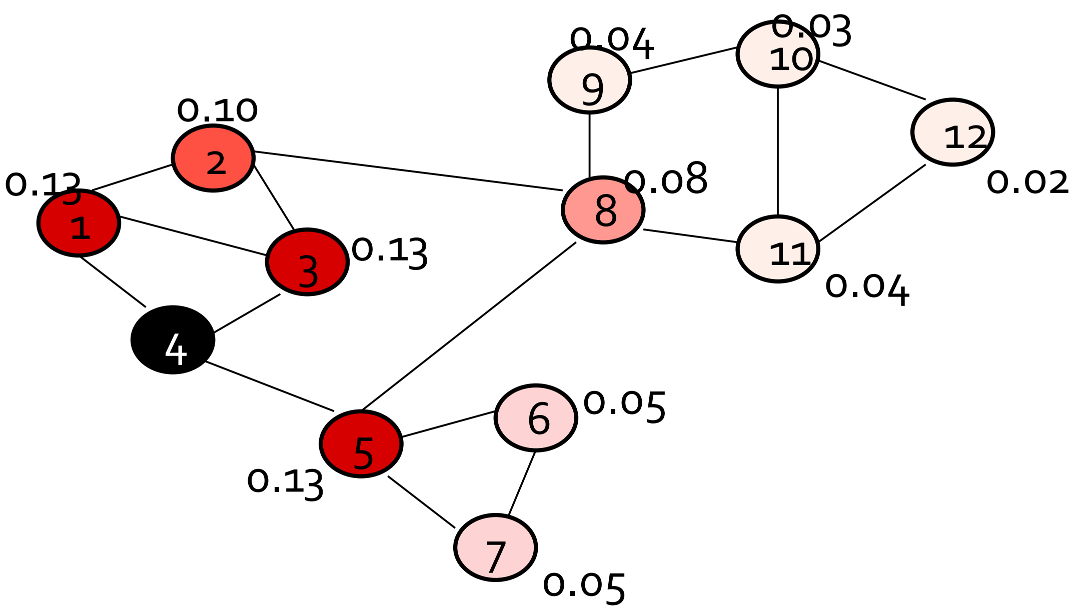
  - **Random Walk with Restart: S** is a single element

PageRank: Applications

- **Graphs and web search:**
  - Ranks nodes by "importance"
- **Personalized PageRank:**
  - Ranks proximity of nodes to the teleport nodes S
- **Proximity on graphs:**
  - **Q:** What is most related conference to **ICDM**?
  - **Random Walks with Restarts**
    - Teleport back to the starting node: S = { **single node** }



Random Walk with Restarts



	Node 4
Node 1	0.13
Node 2	0.10
Node 3	0.13
Node 4	0.22
Node 5	0.13
Node 6	0.05
Node 7	0.05
Node 8	0.08
Node 9	0.04
Node 10	0.03
Node 11	0.04
Node 12	0.02

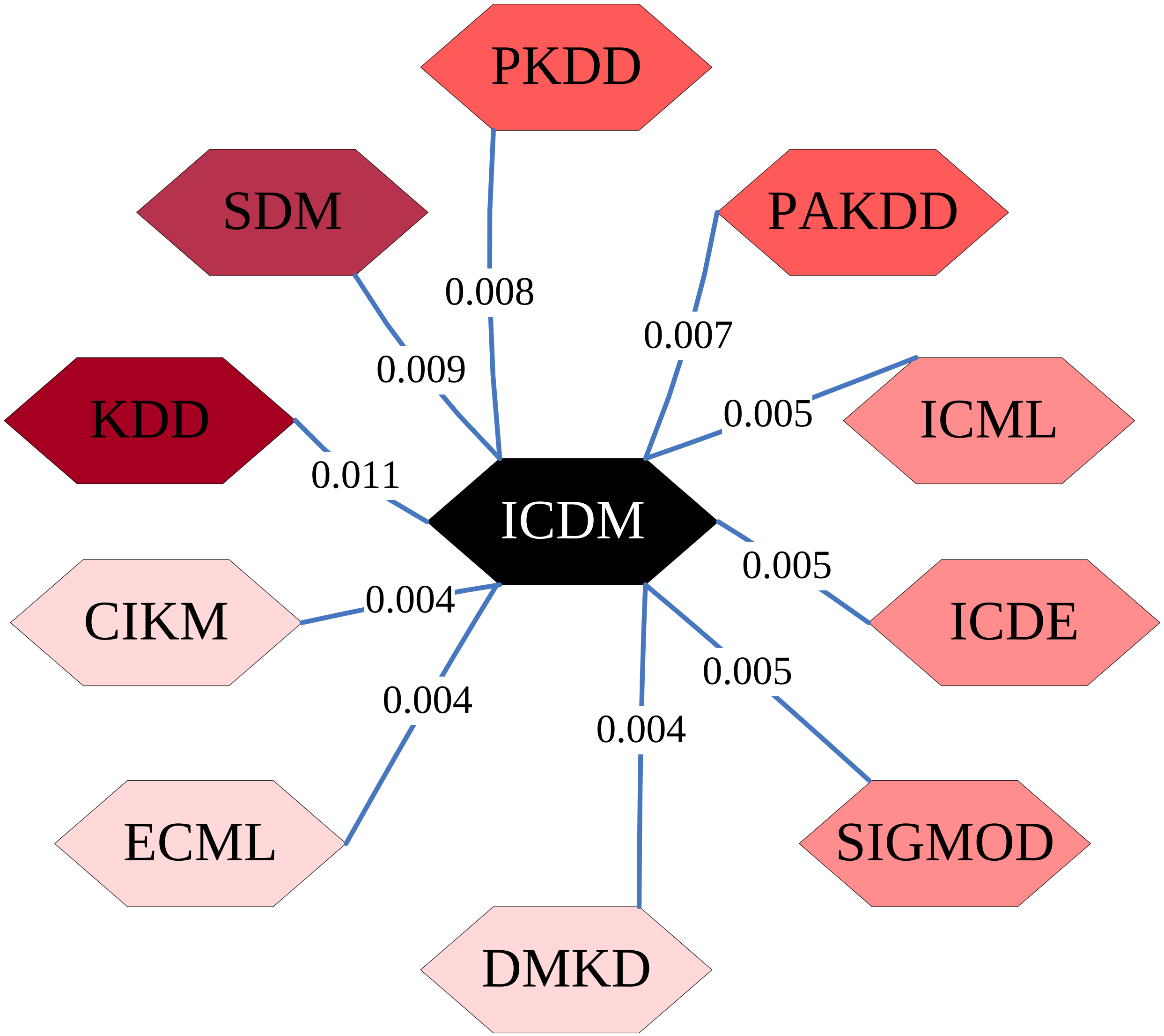
Nearby nodes, higher scores

More red, more relevant

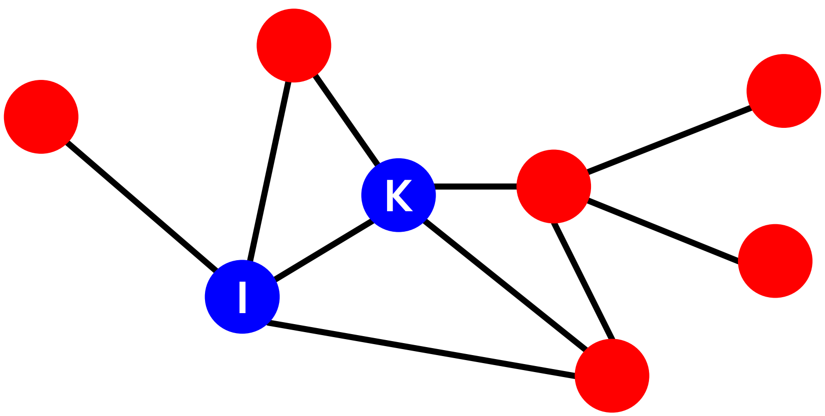
Ranking vector

$\vec{r}_4$

Most related conferences to ICDM



Personalized PageRank



Graph of CS conferences

Q: Which conferences are closest to KDD & ICDM?

A: Personalized PageRank with teleport set *S*={KDD, ICDM}

Link Prediction in Networks

Thanks to Jure Leskovec, Stanford and Panayiotis Tsaparas, Univ. of Ioannina for slides

Agenda

- Link Prediction in Networks
  - Estimating Scores for Missing Edges
  - Classification Approach (Omitted)
- Case studies:
  - Facebook: Supervised Random Walks for Link Prediction
  - Twitter: The who to follow service at Twitter

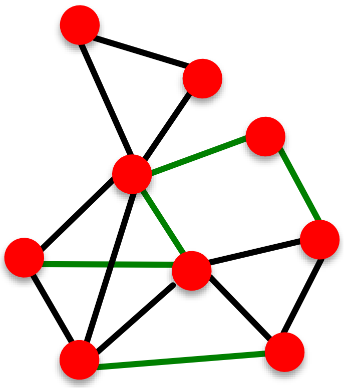
Link Prediction Motivation

- Recommending *new friends* in online social networks
- Predicting the participation of *actors* in events
- Suggesting *interactions* between the members of a company/organization that are external to the hierarchical structure of the organization itself
- Predicting *connections* between members of communities/organizations who have not been directly observed together
- Suggesting *collaborations* between researchers based on co-authorship
- Overcoming the data-sparsity problem *in recommender systems* using collaborative filtering

[LibenNowell-Kleinberg '03]

Link Prediction in Networks

- **The link prediction task:**
  - Given $G[t_0, t_0']$ a graph on edges up to time t_0', **output a ranked list L** of links (not in $G[t_0, t_0']$) that are predicted to appear in $G[t_1, t_1']$



$G[t_0, t_0']$
$G[t_1, t_1']$

- **Evaluation:**
  - $n = |E_{new}|$: # new edges that appear during the test period $[t_1, t_1']$
  - Take top n elements of L and count correct edges

[LibenNowell-Kleinberg '03]

Link Prediction

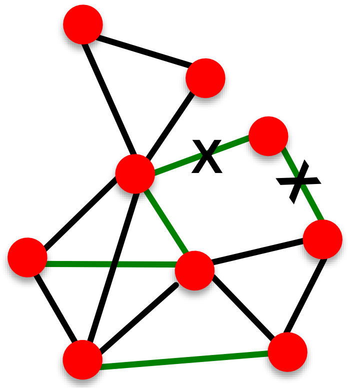
- **Predict links in a evolving collaboration network**

	training period			Core		
	authors	papers	collaborations[1]	authors	$\lvert E_{old} \rvert$	$\lvert E_{new} \rvert$
astro-ph	5343	5816	41852	1561	6178	5751
cond-mat	5469	6700	19881	1253	1899	1150
gr-qc	2122	3287	5724	486	519	400
hep-ph	5414	10254	47806	1790	6654	3294
hep-th	5241	9498	15842	1438	2311	1576

- **Core:** Because network data is very sparse
  - Consider only nodes with degree of at least 3
    - Because we don't know enough about these nodes to make good inferences

Link Prediction

- **Methodology:**
  - For each pair of nodes *(x,y)* compute score *c(x,y)*
    - For example, *c(x,y)* could be the # of common neighbors of *x* and *y*
  - Sort pairs *(x,y)* by the decreasing score *c(x,y)*
    - **Note:** Only consider/predict edges where both endpoints are in the core (*deg.* ≥ *3*)
  - **Predict top n pairs as new links**
  - **See which of these links actually appear in $G[t_1, t_1']$**



[LibenNowell-Kleinberg '03]

Link Prediction

- **Different scoring functions** $c(x,y) =$
  - **Graph distance:** (negated) Shortest path length
  - **Common neighbors:** $|\Gamma(x) \cap \Gamma(y)|$
  - **Jaccard's coefficient:** $|\Gamma(x) \cap \Gamma(y)| / |\Gamma(x) \cup \Gamma(y)|$
  - **Adamic/Adar:** $\sum_{z \in \Gamma(x) \cap \Gamma(y)} 1/\log|\Gamma(z)|$
  - **Preferential attachment:** $|\Gamma(x)| \cdot |\Gamma(y)|$
  - **PageRank:** $r_x(y) + r_y(x)$
    - $r_x(y)$ … stationary distribution score of *y* under the random walk:
      - with prob. 0.15, jump to *x*
      - with prob. 0.85, go to random neighbor of current node

Γ(x) … neighbors of node *x*

- **Then, for a particular choice of *c(·)***
  - For every pair of nodes *(x,y)* compute *c(x,y)*
  - Sort pairs *(x,y)* by the decreasing score *c(x,y)*
  - **Predict top *n* pairs as new links**

Link Prediction Methods

Thanks to Jure Leskovec, Stanford and Panayiotis Tsaparas, Univ. of Ioannina for slides

Link Prediction Methods

- How to assign the score **c(x, y)** for each pair *(x, y)*?
  - Some form of **similarity** between **x** and **y**
  - Some form of **node proximity** between **x** and **y**

- Methods
  - Neighborhood-based (shared neighbors)
  - Network proximity based (paths between **x** and **y**)
  - Other

Methods for Link Prediction

Neighborhood-based

Neighborhood-based Methods

Let $\Gamma(x)$ be the set of neighbors of x in G_{old}

- Methods
  - Common Neighbors Overlap
  - Jaccard
  - Adamic/Adar
  - Preferential Attachment

Neighborhood-based Methods

Intuition: The **larger the overlap** of the neighbors of two nodes, the **more likely** the nodes to be linked in the future

- **Common neighbors**
  - **A:** adjacency matrix, $\mathbf{A}_{x,y}^{2}$: #paths of length 2

$$\mathrm{score}(x,y) = |\Gamma(x) \cap \Gamma(y)|$$

- **Jaccard coefficient**
  - The probability that both **x** and **y** have a feature for a randomly selected feature that either **x** or **y** has

$$\mathrm{score}(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

Neighborhood-based Methods

- **Adamic/Adar**
  - Assigns large weights to common neighbors **z** of **x** and **y** which themselves have few neighbors (weight rare features more heavily)

$$\text{score}(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}$$

- **Preferential attachment**
  - Based on the premise that the probability that a new edge has node **x** as its endpoint is proportional to **|Γ(x)|**, i.e., nodes like to form ties with 'popular' nodes

$$\text{score}(x, y) = |\Gamma(x)||\Gamma(y)|$$

Methods for Link Prediction

Network proximity based

Network Proximity Methods

Intuition: The **“closer” two nodes are** in the network, the **more likely** are to be linked in the future

- Methods
  - based on **shortest path length** between **x** and **y**
  - based on **all paths** between **x** and **y**
    - Katz_β measure (unweighted, weighted)
    - Random walk-based
      - hitting time
      - commute time
      - Rooted PageRank
      - SimRank

Shortest Path Based

For **x**, **y** $\in$ **V×V–E$_{old}$**,

score(x, y) = (negated) **length of shortest path** between **x** and **y**

If there are more than **n** pairs of nodes tied for the shortest path length, order them at random

Ensemble of All Paths

- **Katz$_\beta$ measure**

$$\text{score}(x,y) := \sum_{\ell=1}^{\infty} \beta^{\ell} \cdot |\text{paths}_{x,y}^{\langle \ell \rangle}|$$

  - Sum over all paths of length ***ℓ***
  - *0*<***β***<1: a parameter of the predictor, exponentially damped to count short paths more heavily

Ensemble of All Paths

- **Katz$_\beta$ measure**

$$\text{score}(x, y) := \sum_{\ell=1}^{\infty} \beta^{\ell} \cdot |\text{paths}_{x,y}^{\langle \ell \rangle}|$$

$$\sum_{l=1}^{\infty} \beta^{l} \cdot |\text{paths}_{xy}^{\langle l \rangle}| = \beta A_{xy} + \beta^2 (A^2)_{xy} + \beta^3 (A^3)_{xy} + \cdots$$

  - Unweighted version: **path$_{x,y}$(1)** = **1**, if **x** and **y** have collaborated, **0** otherwise
  - Weighted version: **path$_{x,y}$(1)** = **#times x** and **y** have collaborated

Random Walk Based

Consider a random walk on $\mathbf{G}_{\mathbf{old}}$ that starts at **x** and iteratively moves to a neighbor of **x** chosen uniformly at random from **Γ(x)**

- **Hitting** $\mathbf{H_{x,y}}$ (from **x** to **y**): the expected number of steps it takes for the random walk starting at **x** to reach **y**

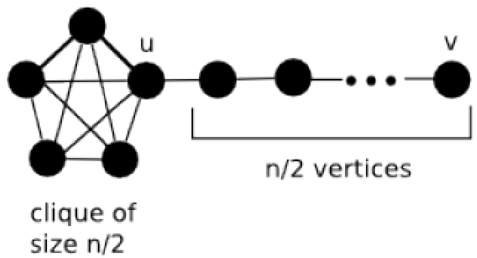
  score(x, y) $= -H_{x,y}$

- **Commute Time** $\mathbf{C_{x,y}}$ (from x to y): the expected number of steps to travel from **x** to **y** and from **y** to **x**

  score(x, y) $= -(H_{x,y} + H_{y,x})$

Not symmetric, can be shown

$$h_{vu} = \Theta(n^2)$$
$$h_{uv} = \Theta(n^3)$$



Random Walk Based

- The hitting time and commute time measures are sensitive to parts of the graph far away from x and y → periodically **reset the walk**
- Random walk on $\mathbf{G_{old}}$ that starts at $\boldsymbol{x}$ and has a probability $\boldsymbol{\alpha}$ of returning to **x** at each step
- **Rooted PageRank**
  - Starts from **x**
  - with probability $(1 - \alpha)$ moves to a random neighbor
  - with probability α returns to **x**

score(x, y) = stationary probability of **y** in a **rooted PageRank**

SimRank

Intuition: Two objects are *similar*, if they are *related to similar objects*

- Two objects ***x*** and ***y*** are *similar*, if they are related to objects ***a*** and ***b*** respectively and ***a*** and ***b*** are themselves similar

$$\text{similarity}(x, y) := \gamma \cdot \frac{\sum_{a \in \Gamma(x)} \sum_{b \in \Gamma(y)} \text{similarity}(a, b)}{|\Gamma(x)| \cdot |\Gamma(y)|}$$

- Expresses the average similarity between neighbors of **x** and neighbors of **y**

score(x, y) = similarity(x, y)

Methods for Link Prediction

Other Methods

Other Methods

- Low-rank Approximations
- Unseen bigrams
- High-level Clustering

Low Rank Approximations

Intuition: represent the adjacency matrix **M** with a lower rank matrix $\mathbf{M_k}$

- Method
  - Apply SVD (singular value decomposition)
  - Obtain the **rank-k** matrix that best approximates **M**

Singular Value Decomposition

$$A = \underset{[n\times r]}{U}\ \underset{[r\times r]}{\Sigma}\ \underset{[r\times n]}{V^T} = \begin{bmatrix}\vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r\end{bmatrix}\begin{bmatrix}\sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r\end{bmatrix}\begin{bmatrix}\vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_r\end{bmatrix}$$

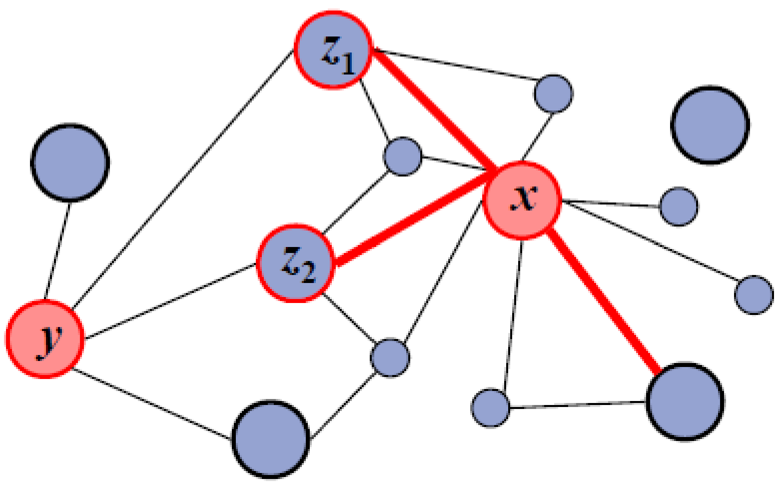
- r: rank of matrix A
- $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$: singular values (square roots of eig-vals AA^T, A^TA)
- $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_r$: left singular vectors (eig-vectors of AA^T)
- $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_r$: right singular vectors (eig-vectors of A^TA)
- $A = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_r \vec{u}_r \vec{v}_r^T$
- $A_k = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_k \vec{u}_k \vec{v}_k^T, k \in \{1,2,\ldots,r\}$

Unseen bigrams

- **Unseen bigrams**: Predict pairs of words that co-occur in a test corpus, but not in the corresponding training corpus
- Not just **score(x, y)** but **score(z, y)** for nodes **z** that are similar to **x** --- $S_x^{(\delta)}$: the **δ** nodes *most related to x*

$$\text{score}^*_{unweighted}(x, y) := \left|\{z : z \in \Gamma(y) \cap S_x^{(\delta)}\}\right|$$

$$\text{score}^*_{weighted}(x, y) := \sum_{z \in \Gamma(y) \cap S_x^{(\delta)}} \text{score}(x, z)$$



Clustering

- Compute **score(x, y)** for all edges in $\mathbf{E_{old}}$
- Delete the **(1-p)** fraction of the edges whose score is the lowest, for some parameter **p**
- Re-compute **score(x, y)** for all pairs in the subgraph

Evaluation & Results

How to Evaluate the Prediction

- Each link predictor ***p*** outputs a ranked list L_p of pairs in $V \times V - E_{old}$ in decreasing order of confidence
  - focus on **Core** network, (d > 3)

$$E^*_{new} = E_{new} \cap (Core \times Core) = |E^*_{new}|$$

- Evaluation method: Size of intersection of
  - the first **n** edge predictions from $\mathbf{L_p}$ that are in Core × Core, and
  - the actual set $\mathbf{E^*_{new}}$

How many of the (relevant) top-n predictions are correct (precision?)

Evaluation: Baseline Predictor

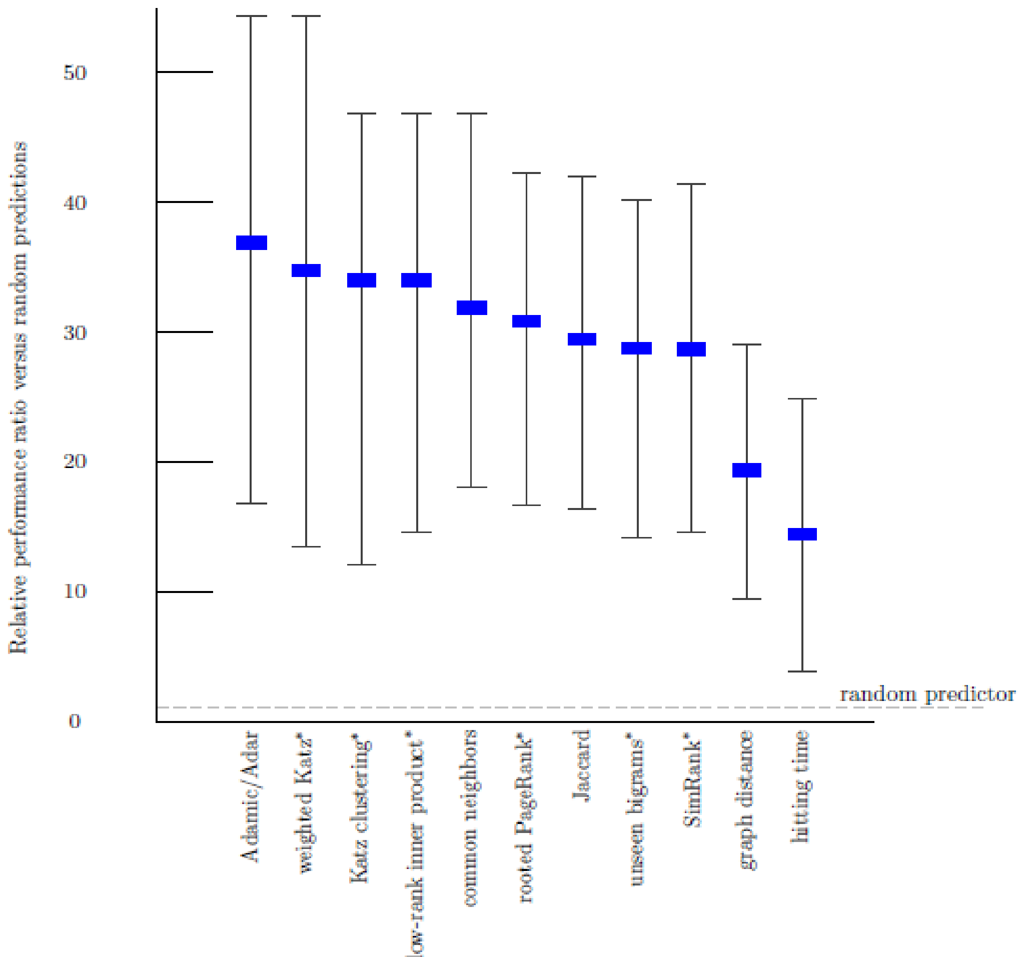
- **Random Predictor:** Randomly select pairs of authors who did not collaborate in the training interval
  - Probability that a random prediction is correct:

$$\frac{|E_{new}|}{\binom{|\text{Core}|}{2} - |E_{old}|}$$

In the datasets, from 0.15% (cond-mat) to 0.48% (astro-ph)

Average Relevance Performance

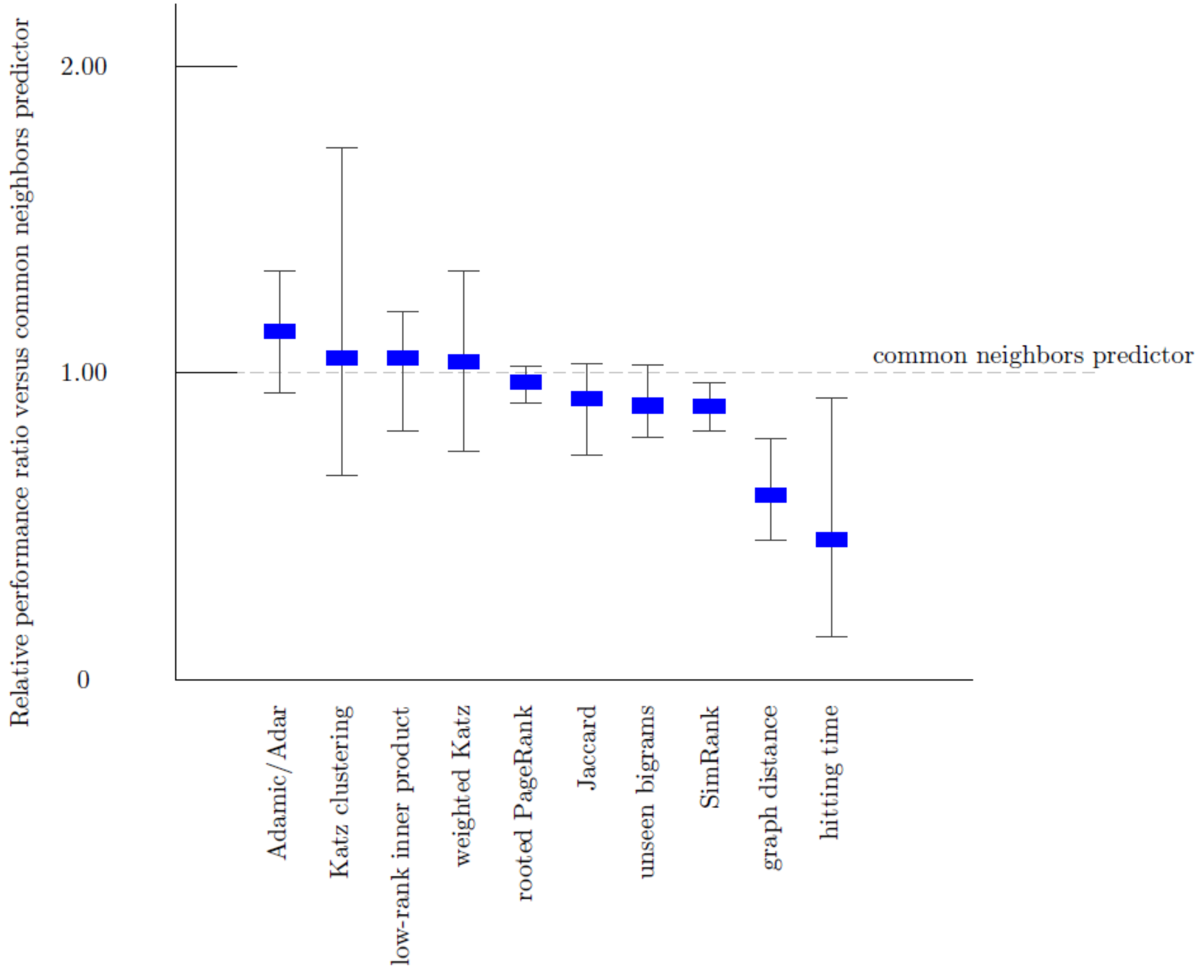
- **Improvement over random predictor**



- average ratio over the five datasets of the given predictor's performance *versus a baseline* predictor's performance.
- the error bars indicate the minimum and maximum of this ratio over the five datasets.
- the parameters for the starred predictors are: (1) for weighted Katz, β= 0.005; (2) for Katz clustering, β1 = 0.001; ρ = 0.15; β2 = 0.1; (3) for low-rank inner product, rank = 256; (4) for rooted Pagerank, α = 0.15; (5) for unseen bigrams, unweighted, common neighbors with δ = 8; and (6) for SimRank, C (γ) = 0.8.

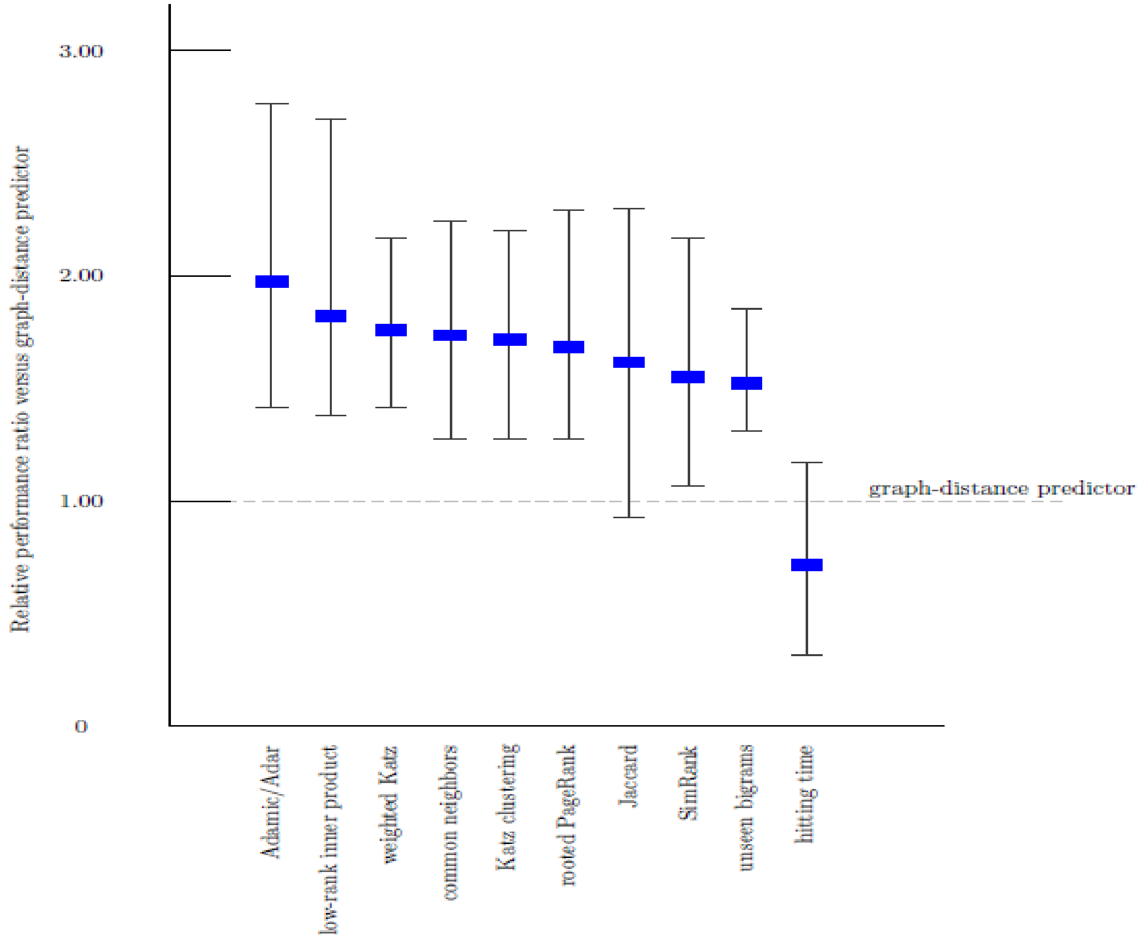
Results: Common Neighbors

- **Improvement over #common neighbors**



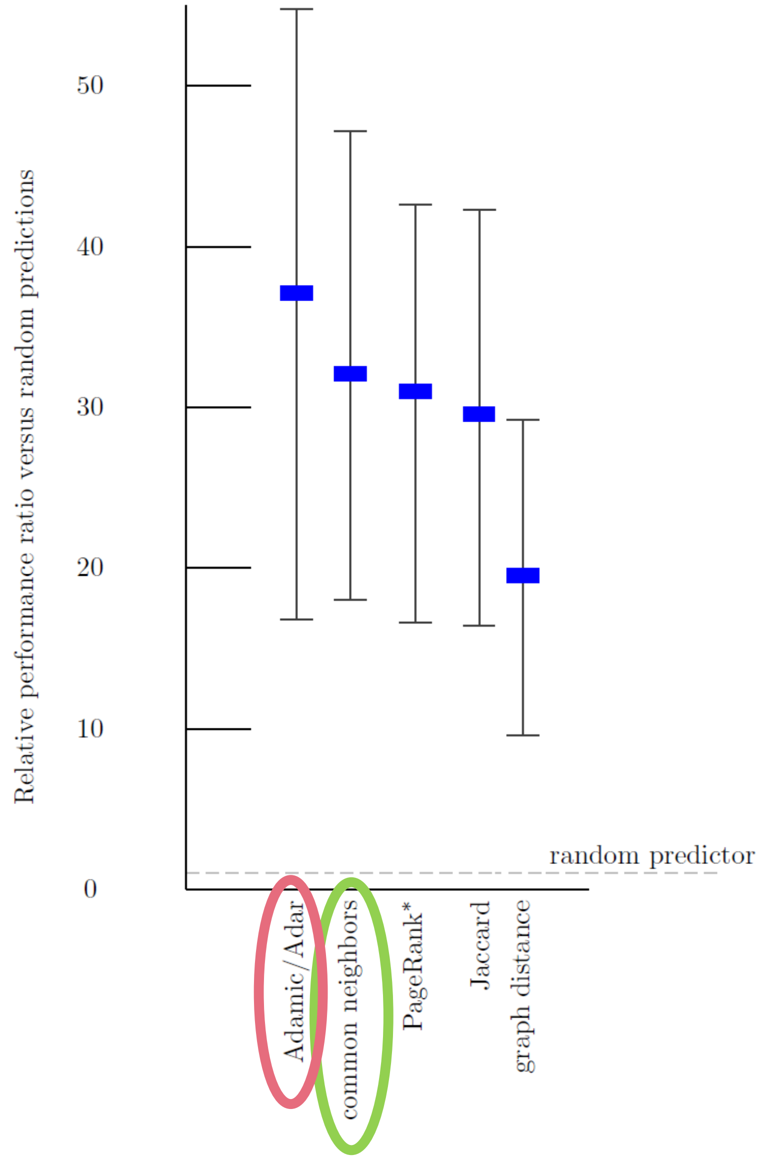
Results: Common Neighbors

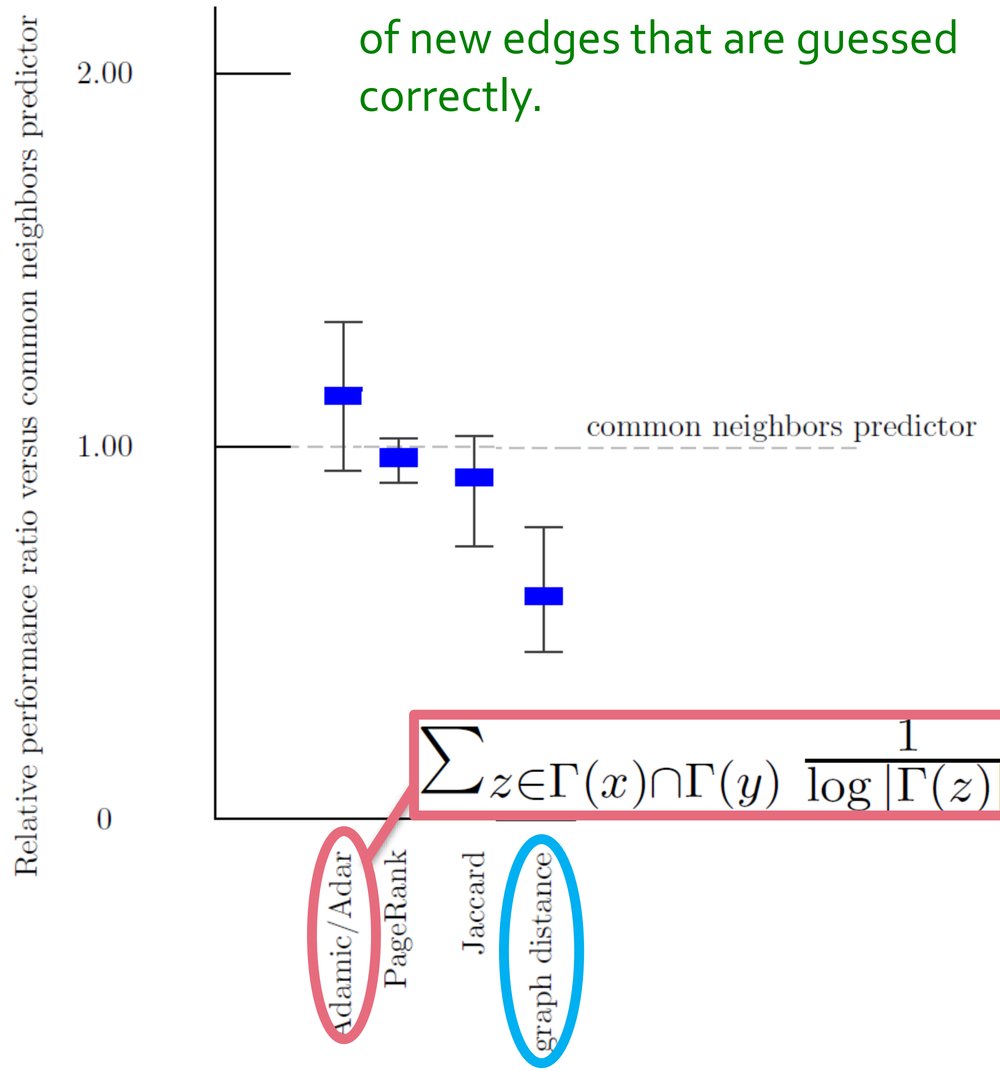
- **Improvement over graph distance predictor**



[LibenNowell-Kleinberg ' 03]

Results: Improvement





Performance score: Fraction of new edges that are guessed correctly.

$$\boxed{\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}}$$

Factor Improvement Over Random

predictor		astro-ph	cond-mat	gr-qc	hep-ph	hep-th
probability that a random prediction is correct		0.475%	0.147%	0.341%	0.207%	0.153%
graph distance (all distance-two pairs)		*9.4*	*25.1*	*21.3*	*12.0*	*29.0*
common neighbors		**18.0**	**40.8**	**27.1**	**26.9**	**46.9**
preferential attachment		4.7	6.0	7.5	*15.2*	7.4
Adamic/Adar		*16.8*	**54.4**	**30.1**	**33.2**	**50.2**
Jaccard		*16.4*	**42.0**	19.8	**27.6**	*41.5*
SimRank	$\gamma = 0.8$	*14.5*	*39.0*	*22.7*	*26.0*	*41.5*
hitting time		6.4	23.7	*24.9*	3.8	13.3
hitting time—normed by stationary distribution		5.3	23.7	11.0	11.3	21.2
commute time		5.2	15.4	**33.0**	*17.0*	23.2
commute time—normed by stationary distribution		5.3	16.0	11.0	11.3	16.2
rooted PageRank	$\alpha = 0.01$	*10.8*	*27.8*	**33.0**	*18.7*	*29.1*
	$\alpha = 0.05$	*13.8*	*39.6*	**35.2**	*24.5*	*41.1*
	$\alpha = 0.15$	*16.6*	**40.8**	**27.1**	**27.5**	*42.3*
	$\alpha = 0.30$	*17.1*	**42.0**	*24.9*	**29.8**	*46.5*
	$\alpha = 0.50$	*16.8*	**40.8**	*24.2*	**30.6**	*46.5*
Katz (weighted)	$\beta = 0.05$	3.0	21.3	19.8	2.4	12.9
	$\beta = 0.005$	*13.4*	**54.4**	**30.1**	*24.0*	**51.9**
	$\beta = 0.0005$	*14.5*	**53.8**	**30.1**	**32.5**	**51.5**
Katz (unweighted)	$\beta = 0.05$	*10.9*	**41.4**	**37.4**	*18.7*	**47.7**
	$\beta = 0.005$	*16.8*	**41.4**	**37.4**	*24.1*	**49.4**
	$\beta = 0.0005$	*16.7*	**41.4**	**37.4**	*24.8*	**49.4**

Factor Improvement Over Random

predictor		astro-ph	cond-mat	gr-qc	hep-ph	hep-th
probability that a random prediction is correct		0.475%	0.147%	0.341%	0.207%	0.153%
graph distance (all distance-two pairs)		*9.4*	*25.1*	*21.3*	*12.0*	*29.0*
common neighbors		**18.0**	**40.8**	**27.1**	**26.9**	**46.9**
Low-rank approximation: Inner product	rank $= 1024$	*15.2*	**53.8**	**29.3**	**34.8**	**49.8**
	rank $= 256$	*14.6*	**46.7**	**29.3**	**32.3**	**46.9**
	rank $= 64$	*13.0*	**44.4**	**27.1**	**30.7**	**47.3**
	rank $= 16$	*10.0*	21.3	**31.5**	**27.8**	*35.3*
	rank $= 4$	8.8	15.4	**42.5**	*19.5*	22.8
	rank $= 1$	6.9	5.9	**44.7**	*17.6*	14.5
Low-rank approximation: Matrix entry	rank $= 1024$	8.2	16.6	6.6	*18.5*	21.6
	rank $= 256$	*15.4*	*36.1*	8.1	*26.2*	*37.4*
	rank $= 64$	*13.7*	**46.1**	16.9	**28.1**	*40.7*
	rank $= 16$	9.1	21.3	*26.4*	*23.1*	*34.0*
	rank $= 4$	8.8	15.4	**39.6**	*20.0*	22.4
	rank $= 1$	6.9	5.9	**44.7**	*17.6*	14.5
Low-rank approximation: Katz ($\beta = 0.005$)	rank $= 1024$	*11.4*	*27.2*	**30.1**	**27.0**	*32.0*
	rank $= 256$	*15.4*	**42.0**	11.0	**34.2**	*38.6*
	rank $= 64$	*13.1*	**45.0**	19.1	**32.2**	*41.1*
	rank $= 16$	9.2	21.3	**27.1**	*24.8*	*34.9*
	rank $= 4$	7.0	15.4	**41.1**	*19.7*	22.8
	rank $= 1$	0.4	5.9	**44.7**	*17.6*	14.5
unseen bigrams (weighted)	common neighbors, $\delta = 8$	*13.5*	*36.7*	**30.1**	*15.6*	**46.9**
	common neighbors, $\delta = 16$	*13.4*	*39.6*	**38.9**	*18.5*	**48.6**
	Katz ($\beta = 0.005$), $\delta = 8$	*16.8*	*37.9*	*24.9*	*24.1*	**51.1**
	Katz ($\beta = 0.005$), $\delta = 16$	*16.5*	*39.6*	**35.2**	*24.7*	**50.6**
unseen bigrams (unweighted)	common neighbors, $\delta = 8$	*14.1*	*40.2*	**27.9**	*22.2*	*39.4*
	common neighbors, $\delta = 16$	*15.3*	*39.0*	**42.5**	*22.0*	*42.3*
	Katz ($\beta = 0.005$), $\delta = 8$	*13.1*	*36.7*	**32.3**	*21.6*	*37.8*
	Katz ($\beta = 0.005$), $\delta = 16$	*10.3*	*29.6*	**41.8**	*12.2*	*37.8*
clustering: Katz ($\beta_1 = 0.001, \beta_2 = 0.1$)	$\rho = 0.10$	7.4	*37.3*	**46.9**	**32.9**	*37.8*
	$\rho = 0.15$	*12.0*	**46.1**	**46.9**	*21.0*	*44.0*
	$\rho = 0.20$	4.6	*34.3*	19.8	*21.2*	*35.7*
	$\rho = 0.25$	3.3	*27.2*	20.5	*19.4*	17.4

Evaluation: Prediction Overlap

	Adamic/Adar	Katz clustering	common neighbors	hitting time	Jaccard's coefficient	weighted Katz	low-rank inner product	rooted Pagerank	SimRank	unseen bigrams
Adamic/Adar	1150	638	520	193	442	1011	905	528	372	486
Katz clustering		1150	411	182	285	630	623	347	245	389
common neighbors			1150	135	506	494	467	305	332	489
hitting time				1150	87	191	192	247	130	156
Jaccard's coefficient					1150	414	382	504	845	458
weighted Katz						1150	1013	488	344	474
low-rank inner product							1150	453	320	448
rooted Pagerank								1150	678	461
SimRank									1150	423
unseen bigrams										1150

How similar are the predictions made by the different methods?

 # common predictions

 # correct predictions

	Adamic/Adar	Katz clustering	common neighbors	hitting time	Jaccard's coefficient	weighted Katz	low-rank inner product	rooted Pagerank	SimRank	unseen bigrams
Adamic/Adar	92	65	53	22	43	87	72	44	36	49
Katz clustering		78	41	20	29	66	60	31	22	37
common neighbors			69	13	43	52	43	27	26	40
hitting time				40	8	22	19	17	9	15
Jaccard's coefficient					71	41	32	39	51	43
weighted Katz						92	75	44	32	51
low-rank inner product							79	39	26	46
rooted Pagerank								69	48	39
SimRank									66	34
unseen bigrams										68

Extensions

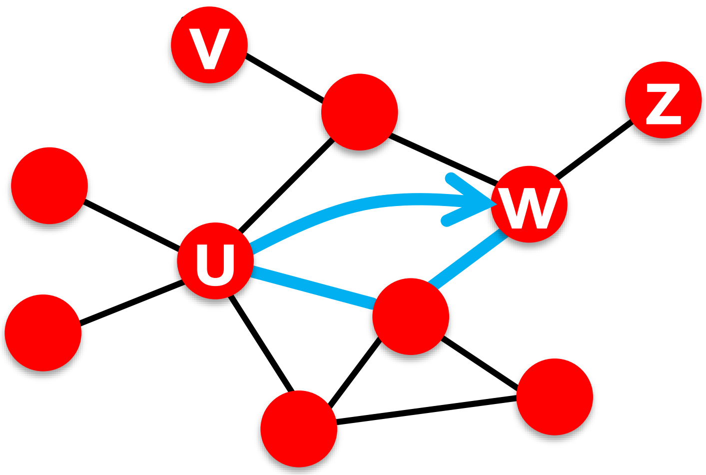
- Improve performance. Even the best (Katz clustering on gr-qc) correct on only about 16% of its prediction
- Improve efficiency on very large networks (approximation of distances)
- Treat more recent links (e.g., collaborations) as more important
- Additional information (paper titles, author institutions, etc) latently present in the graph

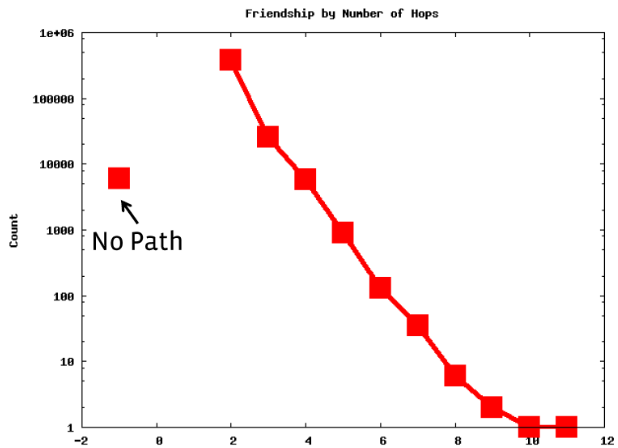
Facebook: Supervised Random Walks for Link Prediction

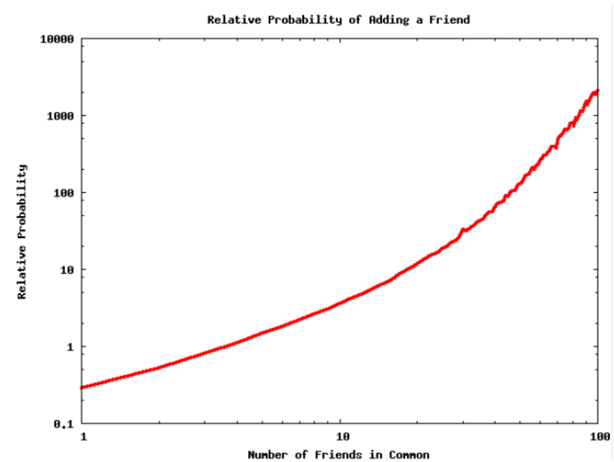
Supervised Link Prediction

[WSDM '11]

- **Can we learn to predict new friends?**
  - **Facebook's People You May Know**
  - **Let's look at the FB data:**
    - 92% of new friendships on FB are friend-of-a-friend
    - More mutual friends helps

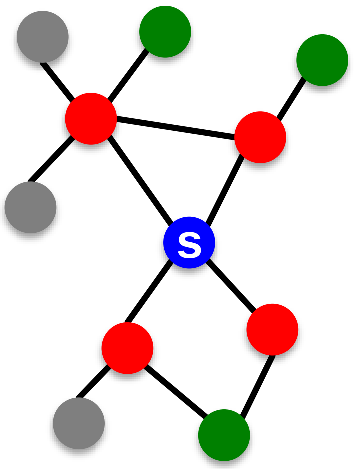






Supervised Link Prediction

- **Goal:** **Recommend a list of possible friends**
- **Supervised machine learning setting:**
  - **Labeled training examples:**
    - For every user s have a list of others she will create links to $\{d_1 \ \dots \ d_k\}$ **in the future**
      - *Use* FB network from May 2012 and $\{d_1 \ \dots \ d_k\}$ are the new friendships you created since then
      - These are the "positive" training examples
    - Use all other users as "negative" example
  - **Task:**
    - For a given node s, **score** nodes $\{d_1 \ \dots \ d_k\}$ **higher** than any other node in the network

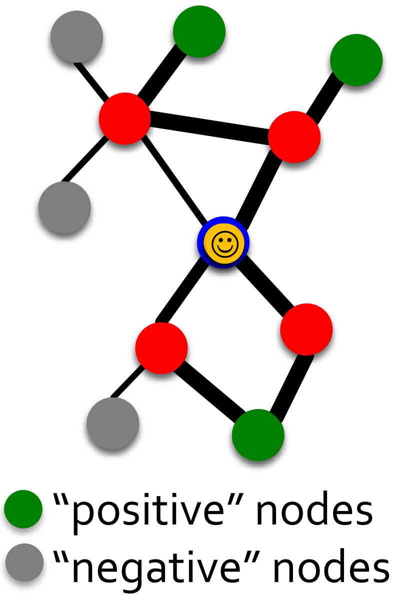


"positive" nodes
"negative" nodes

Green nodes are the nodes to which **s** creates links in the future

Supervised Link Prediction

- **How to combine node/edge features and the network structure?**
  - Estimate **strength** of each friendship (u, v) using:
    - Profile of user u, profile of user v
    - Interaction history of users u and v
  - This creates a **weighted graph**
  - Do **Personalized PageRank from s** and measure the "**proximity**" (the visiting prob.) of any other node w from s
  - Sort nodes w by decreasing **"proximity"**

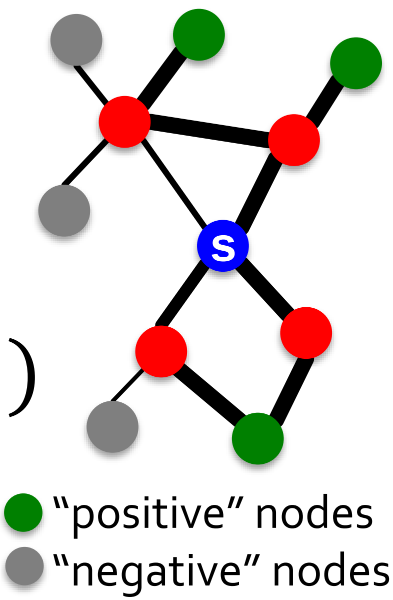


Supervised Random Walks

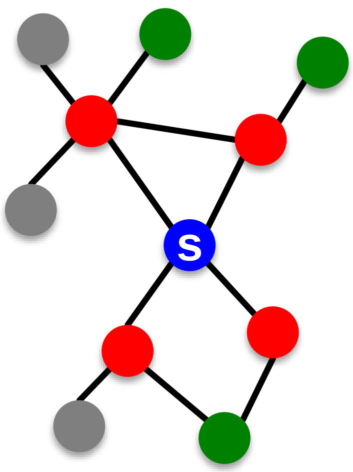
- Let s be the starting node
- Let $f_\beta(u, v)$ be a function that assigns **strength a_{uv} to edge (u, v)**

$$a_{uv} = f_\beta(u, v) = \exp(-\sum_i \beta_i \cdot \mathrm{x}_{uv}[i])$$

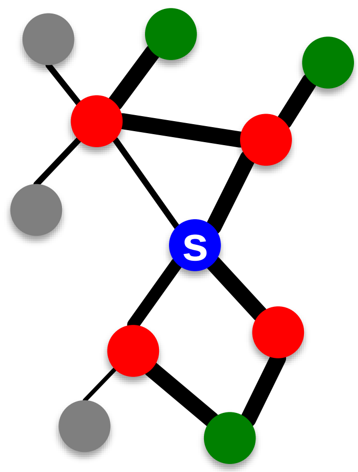
  - **x_{uv} is a feature vector of (u, v)**
    - Features of node u
    - Features of node v
    - Features of edge (u, v)
  - **Note: β is the weight vector we will later estimate!**
- Do **Random Walk with Restarts** from s where transitions are according to edge strengths a_{uv}



SRW: Prediction



Network



Set edge strengths
$a_{uv} = f_\beta(u,v)$

Random Walk with Restarts on the weighted graph.
Each node w has a PageRank proximity p_w

Sort nodes w by the decreasing PageRank score p_w

Recommend top k nodes with the highest proximity p_w to node s

- **How to estimate edge strengths?**
  - **How to set parameters β of $f_\beta(u,v)$?**
- **Idea:** Set β such that it (correctly) predicts the known future links

Personalized PageRank

- a_{uv} .... Strength of edge (u, v)
- **Random walk transition matrix:**

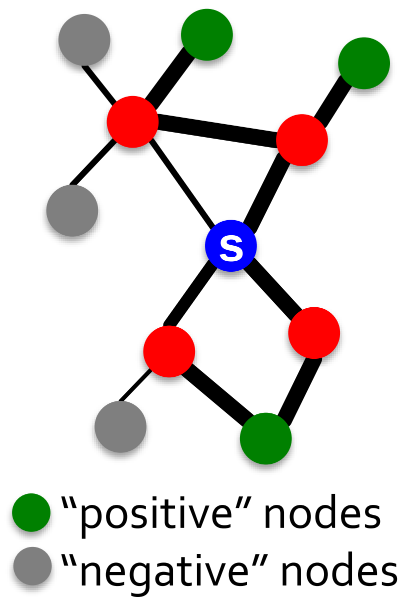
$$Q'_{uv} = \begin{cases} \frac{a_{uv}}{\sum_w a_{uw}} & \text{if } (u,v) \in E, \\ 0 & \text{otherwise} \end{cases}$$

- **PageRank transition matrix:**

$$Q_{ij} = (1-\alpha)Q'_{ij} + \alpha \mathbf{1}(j = s)$$

  - Where with prob. α we jump back to node s

- **Compute PageRank vector:** $p = p^T Q$
- **Rank** nodes w by decreasing p_w



The Optimization Problem

- **Positive** examples
  $D = \{d_1, \ldots, d_k\}$
- **Negative** examples
  $L = \{other\ nodes\}$
- **What do we want?**

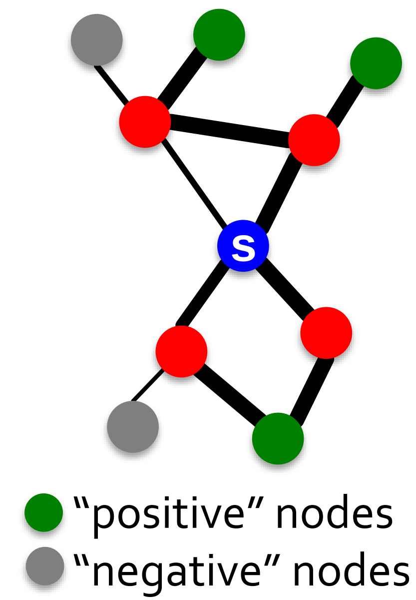
$$\min_{\beta} F(\beta) = ||\beta||^2$$

We prefer small weights β to prevent overfitting

such that

$$\forall\ d \in D, l \in L:\ p_l < p_d$$

Every positive example has to have higher PageRank score than every negative example



- **Note:**
  - Exact solution to this problem may not exist
  - So we make the constraints "soft" (i.e., optional)

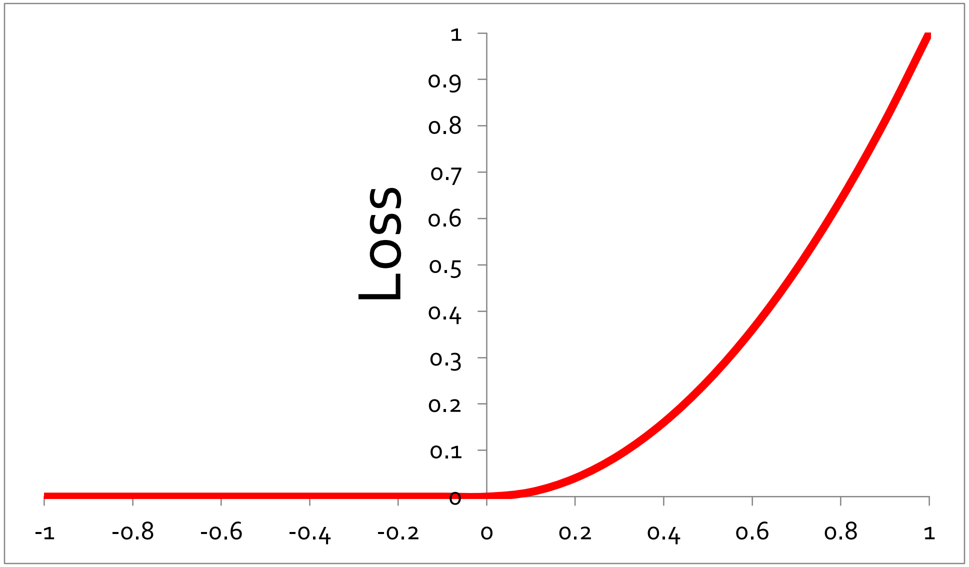
Making Constraints "Soft"

- **Want to minimize:**

$$\min_{\beta} F(\beta) = \sum_{d \in D, l \in L} h(p_l - p_d) + \lambda ||\beta||^2$$

Penalty for violating the constraint that $p_d > p_l$

- **Loss:** $h(x) = 0$ if $x < 0$, or x^2 else



$p_l < p_d$ $\quad$ $p_l = p_d$ $\quad$ $p_l > p_d$

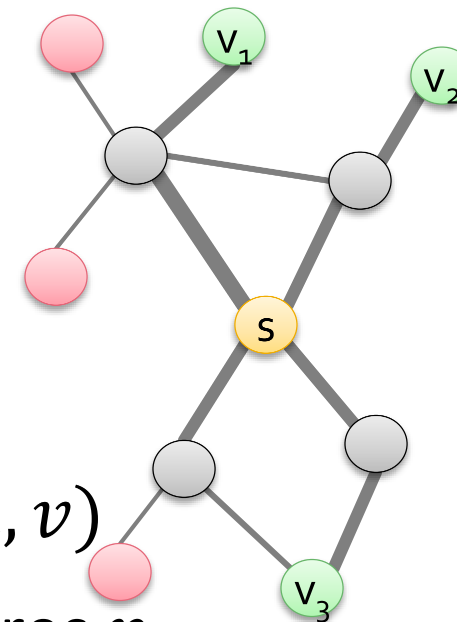
Solving the Problem: Intuition

[WSDM '11]

- **How to minimize *F*?**

$$\min_{\beta} F(\beta) = \sum_{d \in D, l \in L} h(p_l - p_d) + \lambda ||\beta||^2$$

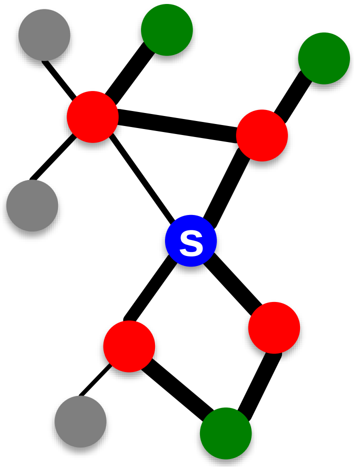
- **Both p_l and p_d depend on β**
  - Given β assign edge weights $a_{uv} = f_\beta(u, v)$
  - Using $Q = [a_{uv}]$ compute PageRank scores p_β
  - Rank nodes by the decreasing score



- **Goal: Want to find β such that $p_l < p_d$**

[WSDM '11]

Solving the Problem: Intuition



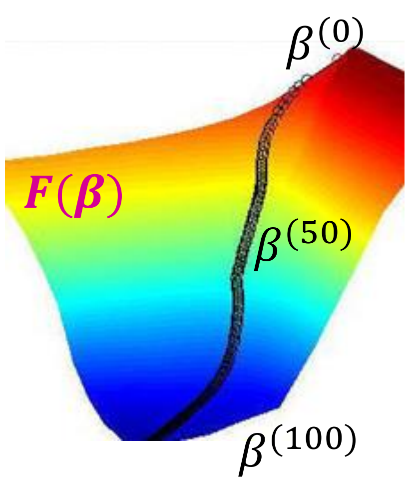
- **How to minimize $F(\beta)$?**

$$\min_{\beta} F(\beta) = \sum_{d \in D, l \in L} h(p_l - p_d) + \lambda ||\beta||^2$$

- **Idea:**
  - Start with some random $\beta^{(0)}$
  - Evaluate the derivative of $F(\beta)$ and do a small step in the opposite direction

$$\beta^{(t+1)} = \beta^{(t)} - \eta \frac{\partial F(\beta^{(t)})}{\partial \beta}$$

  - Repeat until convergence



Gradient Descent

Details!

$$F(\beta) = \sum_{d \in D, l \in L} h(p_l - p_d) + \lambda ||\beta||^2$$

- **What's the derivative $\dfrac{\partial F(\beta^{(t)})}{\partial \beta}$?**

$$\frac{\partial F(\beta)}{\partial \beta} = \sum_{l,d} \frac{\partial h(p_l - p_d)}{\partial \beta} + 2\lambda\beta$$

$$= \sum_{l,d} \frac{\partial h(p_l - p_d)}{\partial (p_l - p_d)} \left(\frac{\partial p_l}{\partial \beta} - \frac{\partial p_d}{\partial \beta}\right) + 2\lambda\beta$$

$$h(x) = \max\{x, 0\}^2$$

Easy!

- **We know:**

$p = p^T Q$ ***that is*** $p_u = \sum_j p_j Q_{ju}$

- **So:**

$$\boxed{\frac{\partial p_u}{\partial \beta} = \sum_j Q_{ju} \frac{\partial p_j}{\partial \beta} + p_j \frac{\partial Q_{ju}}{\partial \beta}}$$

Gradient Descent

Details!

- **We just got:** $\frac{\partial p_u}{\partial \beta} = \sum_j Q_{ju} \frac{\partial p_j}{\partial \beta} + p_j \frac{\partial Q_{ju}}{\partial \beta}$
  - **Few details:**
    - Computing $\partial Q_{ju} / \partial \beta$ is easy. **Remember:** $Q'_{uv} = \begin{cases} \frac{a_{uv}}{\sum_w a_{uw}} & \text{if } (u,v) \in E, \\ 0 & \text{otherwise} \end{cases}$
    - We want $\frac{\partial p_j}{\partial \beta}$ but it appears on both sides of the equation. Notice the whole thing looks like a PageRank equation: $x = Q \cdot x + z$

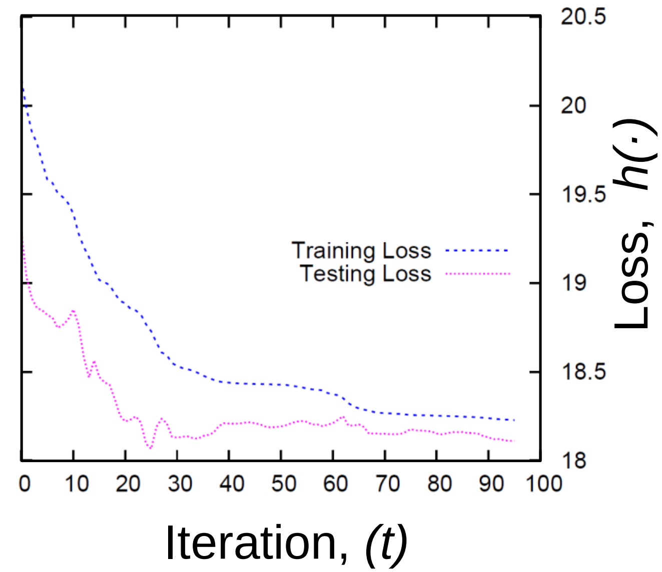
$$a_{uv} = f_\beta(u, v)$$
$$= \exp\left(-\sum_i \beta_i \cdot \mathrm{x}_{uv}[i]\right)$$

- **As with PageRank we can use the power-iteration to solve it:**
  - Start with a random $\frac{\partial p^{(0)}}{\partial \beta}$
  - Then iterate: $\frac{\partial p^{(t+1)}}{\partial \beta} = Q \cdot \frac{\partial p^{(t)}}{\partial \beta} + \frac{\partial Q_{ju}}{\partial \beta} \cdot p$

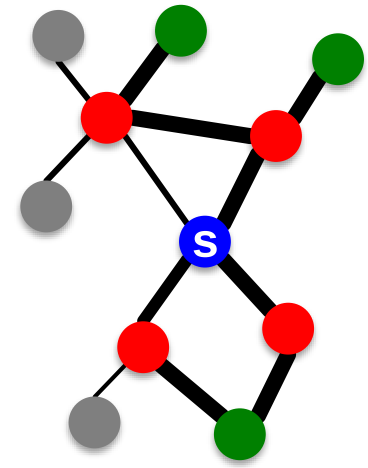
Optimizing $F(\beta)$

Details!

- **To optimize $F(\beta)$, use gradient descent:**
  - Pick a random starting point $\beta^{(0)}$
  - Using current $\beta^{(t)}$ compute edge strenghts and the transition matrix Q
  - Compute PageRank scores p
  - Compute the gradient with respect to weight vector $\beta^{(t)}$
  - Update $\beta^{(t+1)}$



Data: Facebook



- **Facebook Iceland network**
  - 174,000 nodes (55% of population)
  - Avg. degree 168
  - Avg. person added 26 friends/month
- **For every node s:**
  - **Positive examples:**
    - $D = \{$ new friendships s created in Nov '09 $\}$
  - **Negative examples:**
    - $L = \{$ other nodes s did not create new links to $\}$
  - **Limit to friends of friends:**
    - On avg. there are 20,000 FoFs (maximum is 2 million)!

Experimental Setting

- **Node and Edge features for learning:**
  - **Node:** Age, Gender, Degree
  - **Edge:** Age of an edge, Communication, Profile visits, Co-tagged photos
- **Evaluation:**
  - **Precision at top 20**
    - We produce a list of 20 candidates
      - By taking top 20 nodes x with highest PageRank score p_x
    - Measure to what fraction of these nodes s actually links to

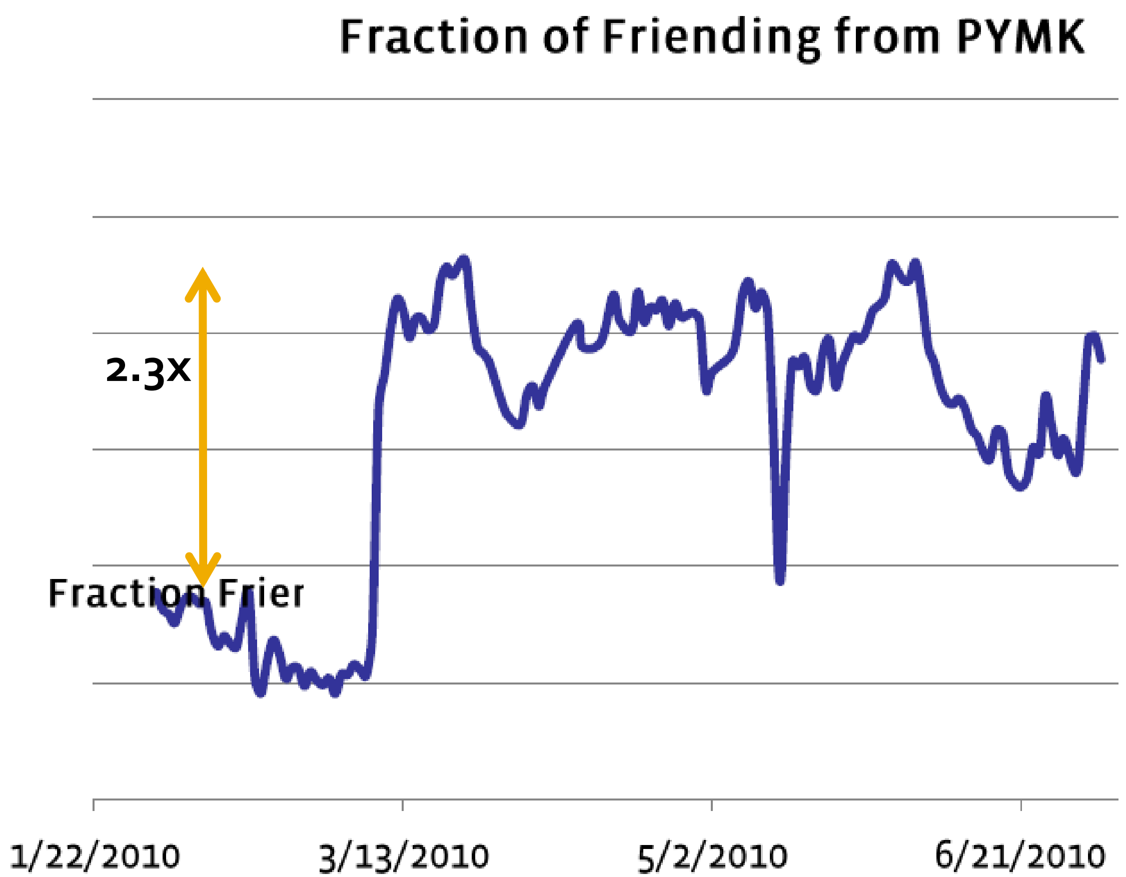
Results: Facebook Iceland

- **Facebook:** Predict future friends
  - Adamic-Adar already works great
  - Supervised Random Walks (SRW) gives slight improvement

Learning Method	Prec@Top20
Random Walk with Restart	6.80
Adamic-Adar	7.35
Common Friends	7.35
Degree	3.25
SRW: one edge type	6.87
SRW: multiple edge types	**7.57**

Results: Facebook

- **2.3x improvement over previous FB-PYMK (People You May Know)**



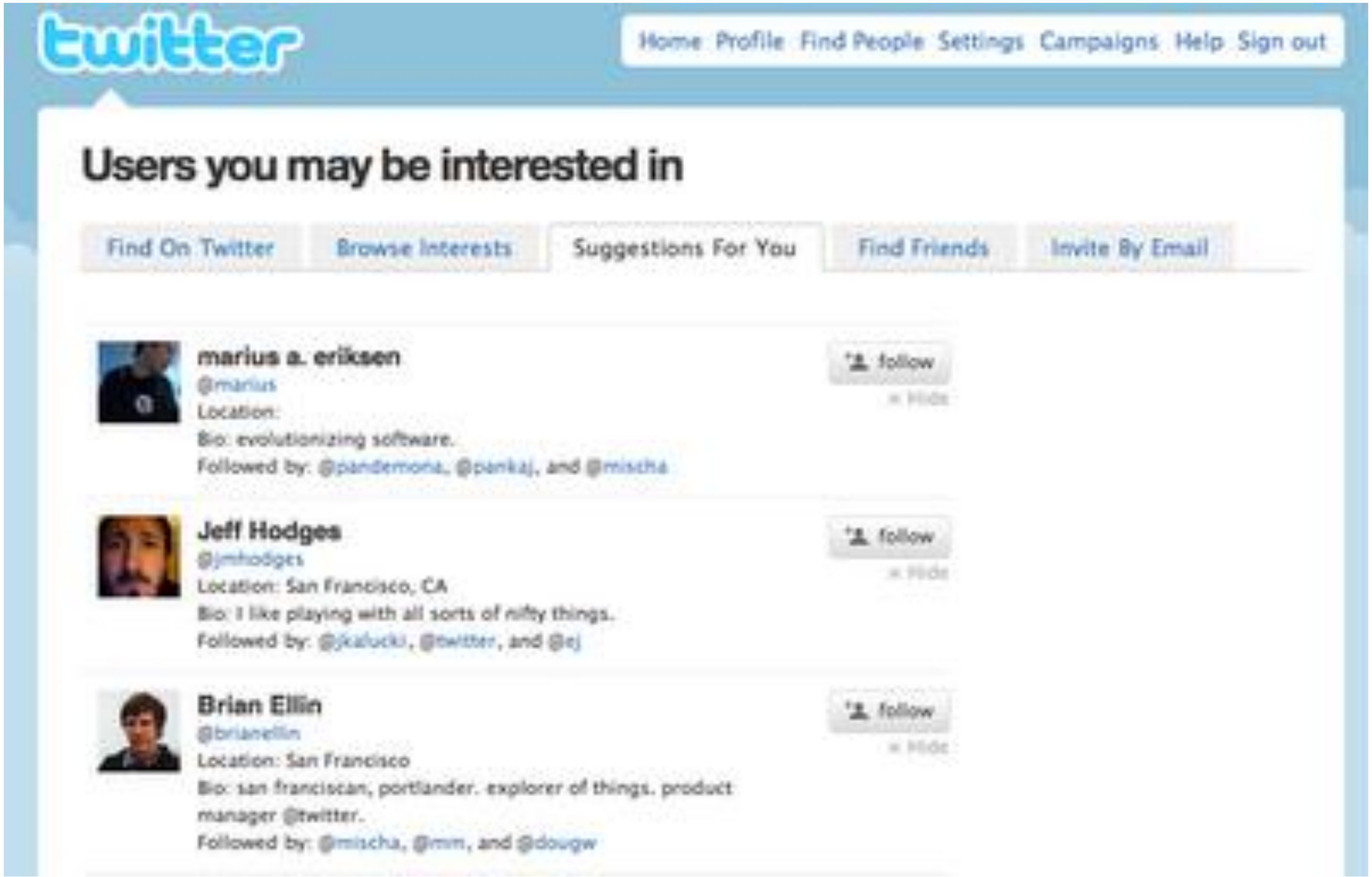
Results: Co-Authorship

- **Arxiv Hep-Ph collaboration network:**
  - Poor performance of unsupervised methods
  - SRW gives a boost of 25%!

Learning Method	Prec@Top20
Random Walk with Restart	3.41
Adamic-Adar	3.13
Common Friends	3.11
Degree	3.05
SRW: one edge type	4.24
SRW: multiple edge types	**4.25**

Wtf: The Who to Follow Service at Twitter

Introduction



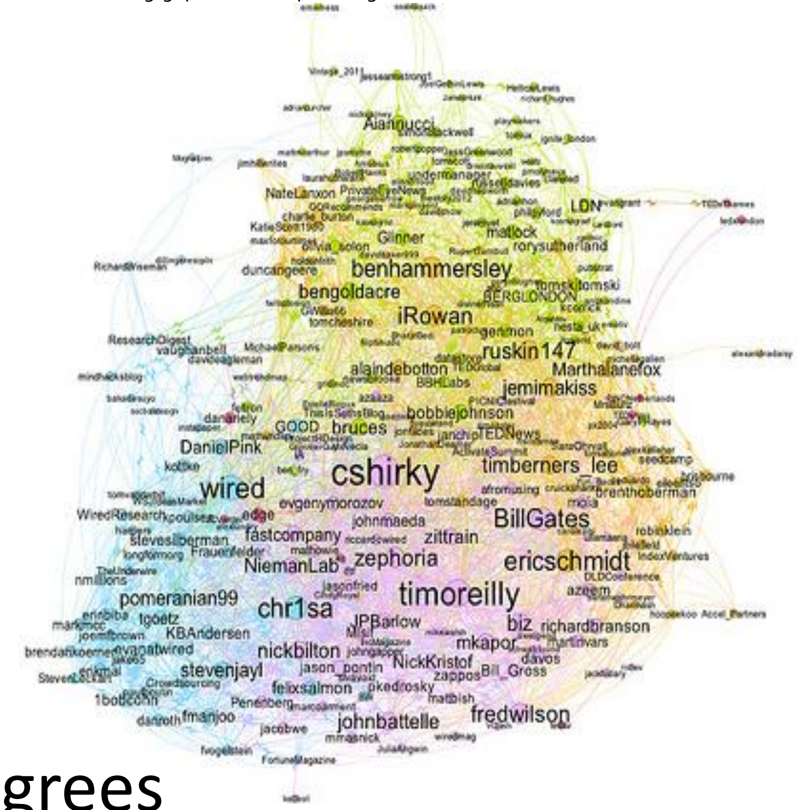
Semantic differences between "interested in" and "similar to"

WtF ("Who to Follow")

- **WtF ("Who to Follow")**: the Twitter user recommendation service
  - help existing and new users to **discover connections** to sustain and grow
  - used for **search relevance**, **content discovery**, **promoted products**, etc.
  - **Twitter Data**:
    - 200 million users
    - 400 million tweets every day (as of early 2013)
    - http://www.internetlivestats.com/twitter-statistics/

The Twitter Graph

http://blog.ouseful.info/2011/07/07/visualising-twitter-friend-connections-using-gephi-an-example-using-wireduk-friends-network/



- **Graph**
  - **Node**: user
  - **(Directed) Edge**: follows
- **Graph Statistics** (Aug'12)
  - Over 20 billion edges
  - Power law of in- and out-degrees
  - Over 1000 with more than 1 million followers
  - 25 users with more than 10 million followers

Algorithms: Circle of Trust

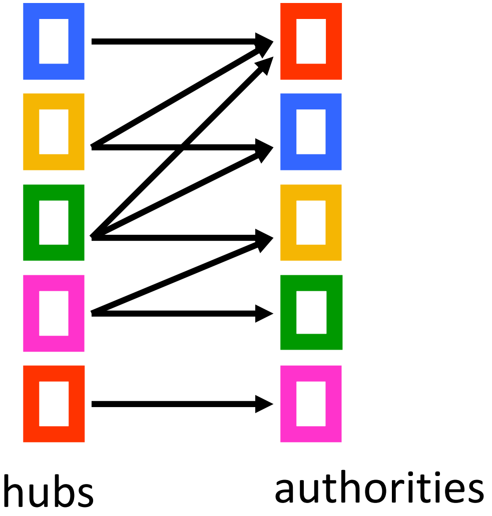
- **Circle of Trust**
  - Based on an **egocentric random walk** (similar to **personalized (rooted) PageRank**)
- Computed in an **online fashion** (from scratch each time) given **a set of parameters**
  - # of random walk steps
  - reset probability
  - pruning settings to discard low probability vertices
  - parameters to control sampling of outgoing edges at vertices with large out-degrees

Algorithms

- Directed edge
  - **Asymmetric** nature of the **follow relationship**
    - Friendships in other social networks such as Facebook or LinkedIn are symmetric/reciprocal
  - Similar to the **user-item recommendations** problem where the "item" is also a user

Algorithms: SALSA

- **SALSA** (Stochastic Approach for Link-Structure Analysis)
  - a variation of HITS
- **HITS**
  - Intuition:
    - Good **hubs** point to good **authorities**
    - Good **auth.** are pointed by good **hubs**
  - Recurs. comput. of **hub score**
  - Recurs. comput. of **auth. score**



$$h_i = \sum_{j:i \to j} a_j$$

$$a_i = \sum_{j:j \to i} h_j$$

Algorithms: SALSA

- Random walks to rank **hubs** and **authorities**
  - Two different random walks (Markov chains): **a chain of hubs** and **a chain of authorities**
  - Each walk traverses nodes only in one side by traversing two links in each step **h→a→h**, **a→h→a**

Transition matrices of each chain: **H** and **A**

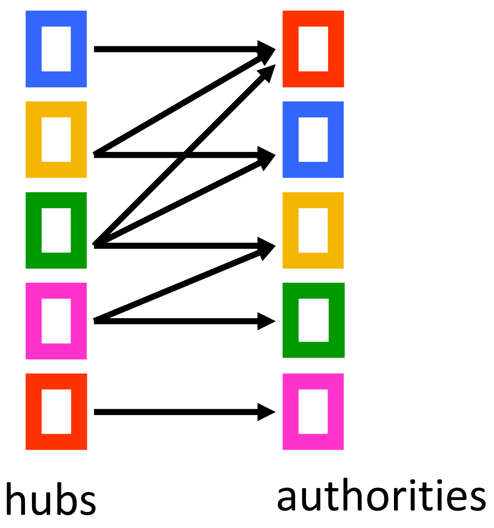
W: the adjacency of the directed graph

$\mathbf{W_r}$: divide each entry by the sum of its row

$\mathbf{W_c}$: divide each entry by the sum of its column

$\mathbf{H} = W_r W_c^T$

$\mathbf{A} = W_c^T W_r$

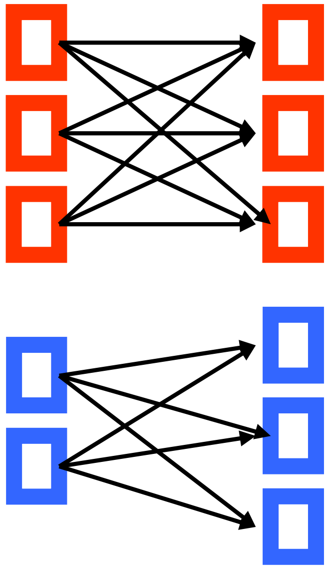


Algorithms: SALSA

- Reduces to the problem of HITS with **tightly knit communities**
  - **TKC effect**
- Better for single-topic communities
- More efficient implementation

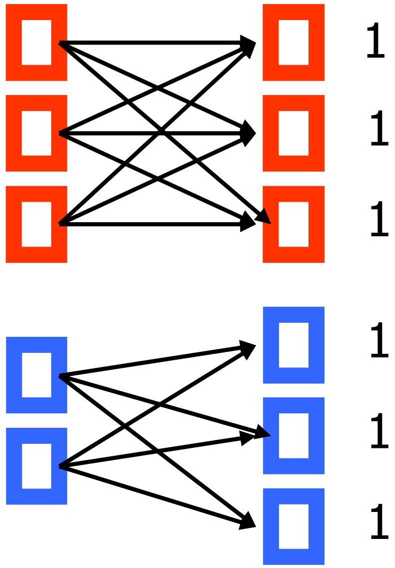
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



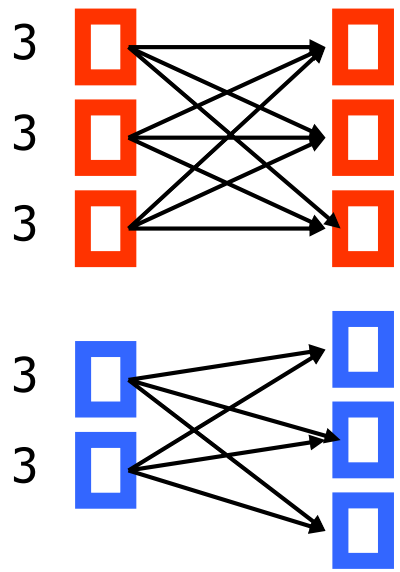
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



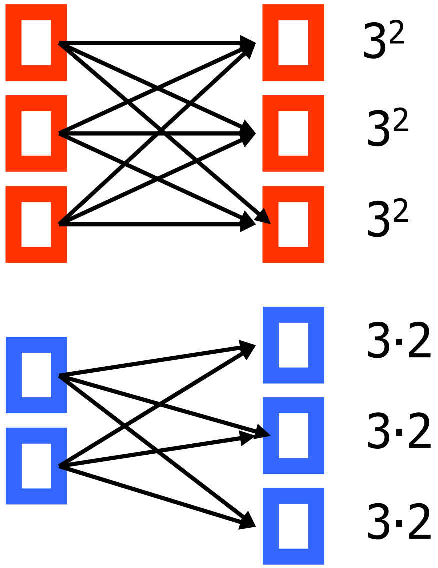
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



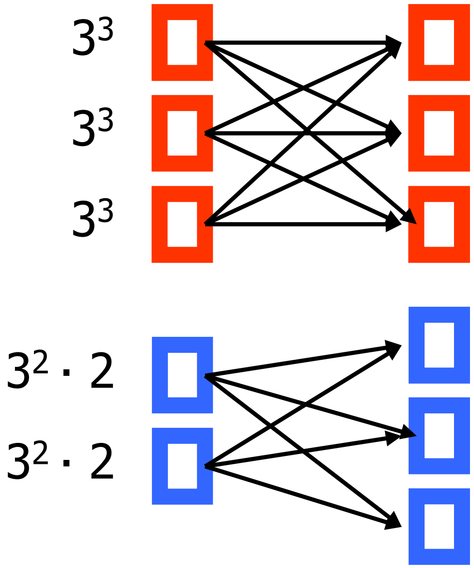
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



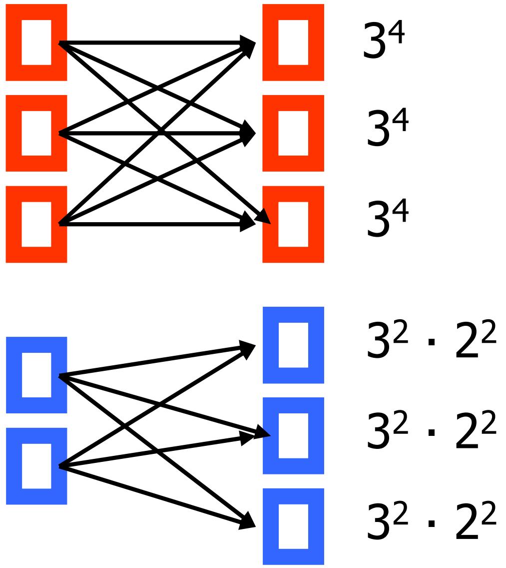
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



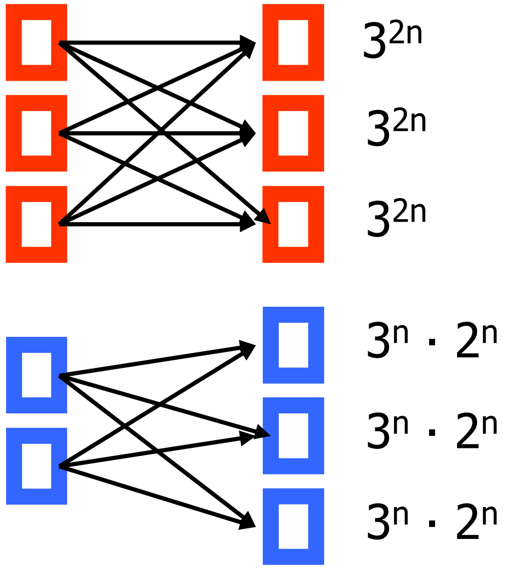
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



HITS and the TKC effect

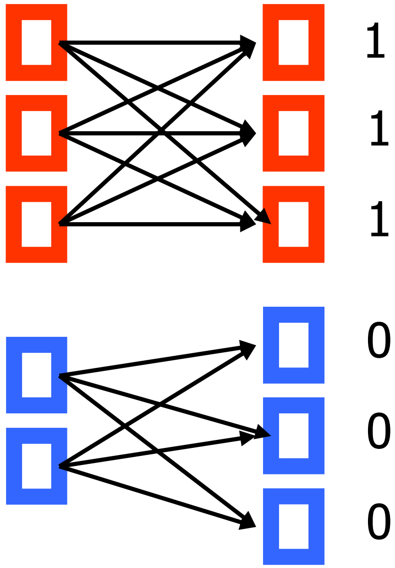
- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



after **n** iterations

HITS and the TKC effect

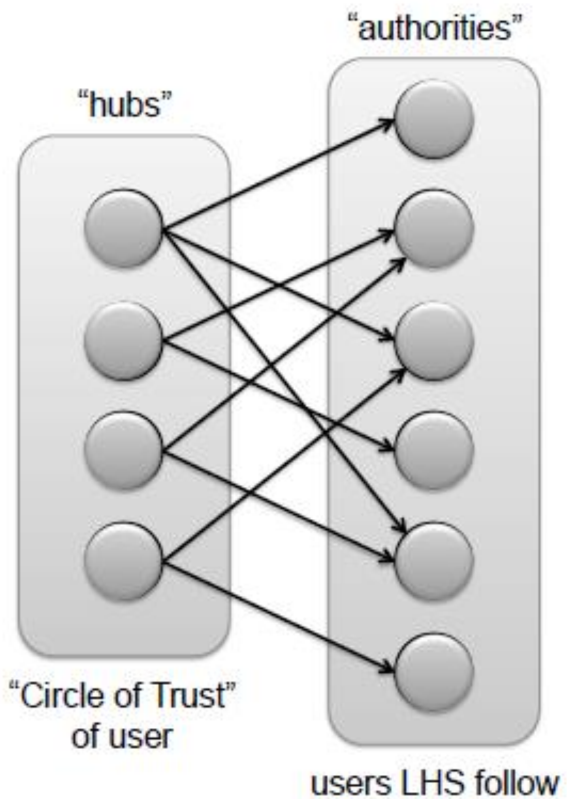
- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



after normalization with the max element as $n \rightarrow \infty$

Algorithms: SALSA

- **Hubs**:
  - 500 top-ranked nodes from a user's circle of trust
  - user similarity (based on homophily, also useful)
- **Authorities**:
  - users that the hubs follow
  - “interested in” user recommendations



Algorithms: SALSA

- **SALSA**'s recursive nature
  - Two users are similar if they follow the same (or similar) users (**LHS**)
  - A user **u** is likely to follow those who are followed by users that are similar to **u** (**RHS**)
- The random walk ensures **fair distribution** of scores in both directions
- Similar users are selected from the circle of trust of a user (via **personalized PageRank**)

Evaluation

- Approaches
  - Offline experiments on retrospective data
  - Online A/B testing on live traffic
- Various parameters may interfere:
  - How the results are rendered
  - Platform (mobile, etc.)
  - New vs old users

Extensions

- Add **metadata to vertices** (e.g., user profile information) and **edges** (e.g., edge weights, timestamp, etc.)
- Consider **interaction graphs** (e.g., graphs defined in terms of retweets, favorites, replies, etc.)

Extensions

- Two phase algorithm
  - **1st - Candidate generation**: produce a list of promising recommendations for each user, using any algorithm
  - **2nd - Rescoring**: apply a machine-learned model to the candidates, binary classification problem (logistic regression)
- Evaluation
  - **1st Phase**: recall + diversity
  - **2nd Phase**: precision + maintain diversity