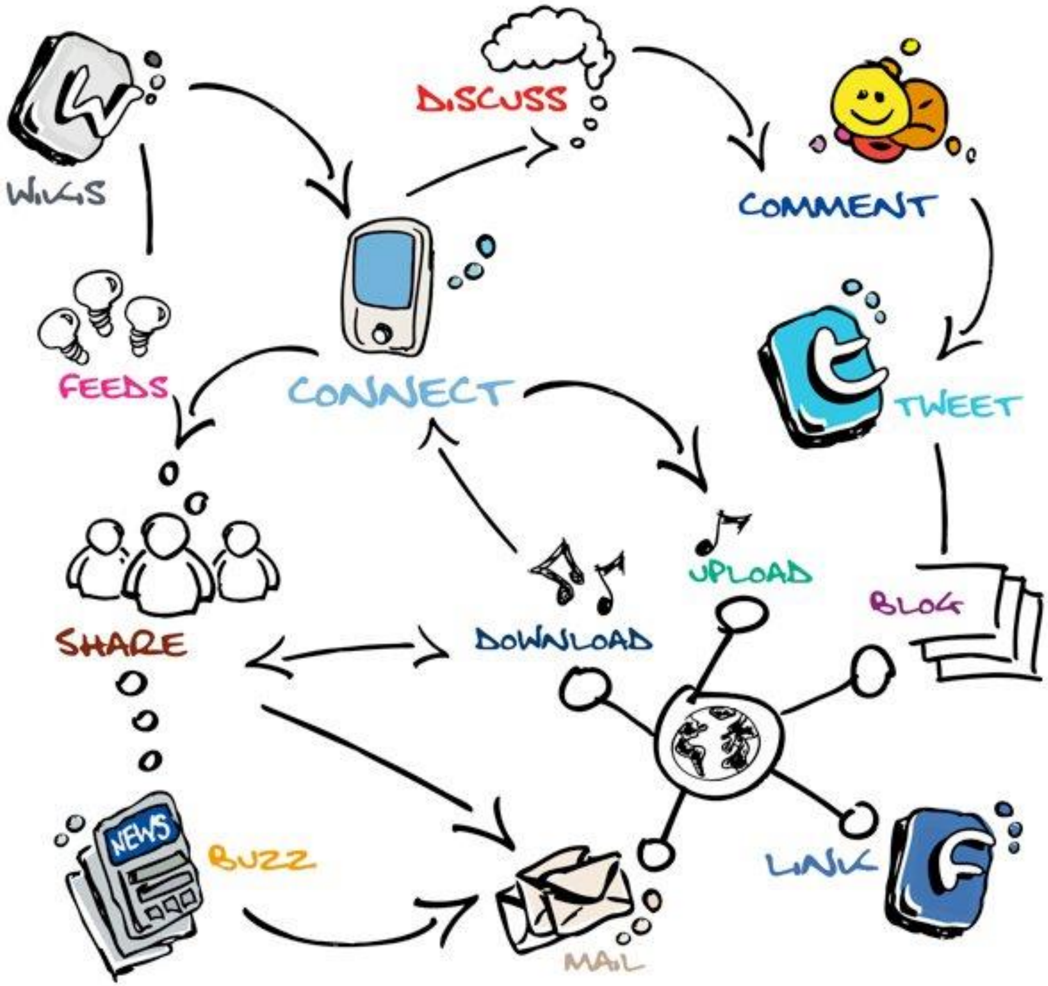


Link Analysis: PageRank and HITS

[Ahmet Onur Durahim]

How the Class Fits Together

Properties	Models	Algorithms
Small diameter, Edge clustering	Small-world model, Erdös-Renyi model	Decentralized search
Scale-free	Preferential attachment, Copying model	PageRank, Hubs and authorities
Strength of weak ties, Core-periphery	Kronecker Graphs	Community detection: Girvan-Newman, Modularity
Densification power law, Shrinking diameters	Microscopic model of evolving networks	Link prediction, Supervised random walks
Complex Graph Structure	Graph Neural Networks	Node Classification Graph Representation Learning
Information virality, Memetracking	Independent cascade model, Game theoretic model	Influence maximization, Outbreak detection, LIM

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!**
    - using not some external source of knowledge but ***information intrinsic to the Web and its structure***

Difficulties in Searches

- ***Diversity in authoring styles*** makes it much harder to rank documents according to a common criterion
  - on a single topic, one can easily find pages written by experts, novices, children, conspiracy theorists
- Rich ***diversity in the set of people issuing queries***, and the ***problem of multiple meanings***
  - when someone issues the single-word query “Cornell”
    - Did the searcher want information about the university?
    - The university’s hockey team?
    - The Nobel-Prize-winning physicist Eric Cornell?

Difficulties in Web Searches

- The ***dynamic*** and ***constantly-changing*** nature of Web content
  - Search for “World Trade Center” on September 11, 2001
    - Google results were all based on pages that were gathered days or weeks earlier
      - the top results were all descriptive pages about the building itself, not about what had occurred that morning
  - Emerging Web sites such as Twitter continue to fill in the spaces that exist between ***static content*** and ***real-time awareness***
- Web has shifted much of the information retrieval question from a problem of ***scarcity*** to a problem of ***abundance***
  - information retrieval in the pre-Web era had a “needle-in-a-haystack” flavour
  - for most Web searches issue is to filter, from among an ***enormous number of relevant documents***, the few that are ***most important***

Web Search: Two Challenges

Two challenges of web search:

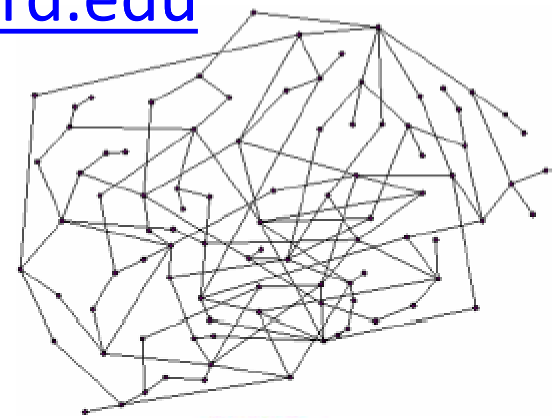
- **(1) Web contains many sources of information Who to "trust"?**
  - **<u>Insight:</u> Trustworthy pages may point to each other!**
- **(2) What is the "best" answer to query "newspaper"?**
  - No single right answer
  - **<u>Insight:</u> Pages that actually know about newspapers might all be pointing to many newspapers**

Ranking Nodes on the Graph

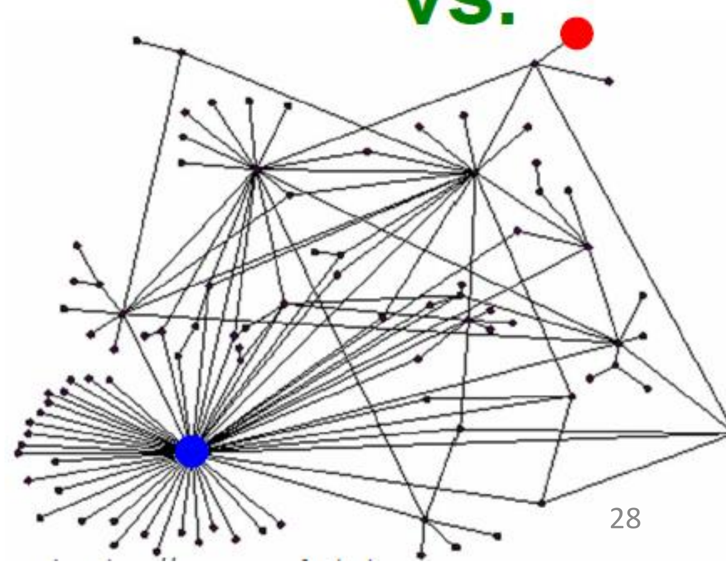
- **All web pages are not equally "important"**

  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***



vs.



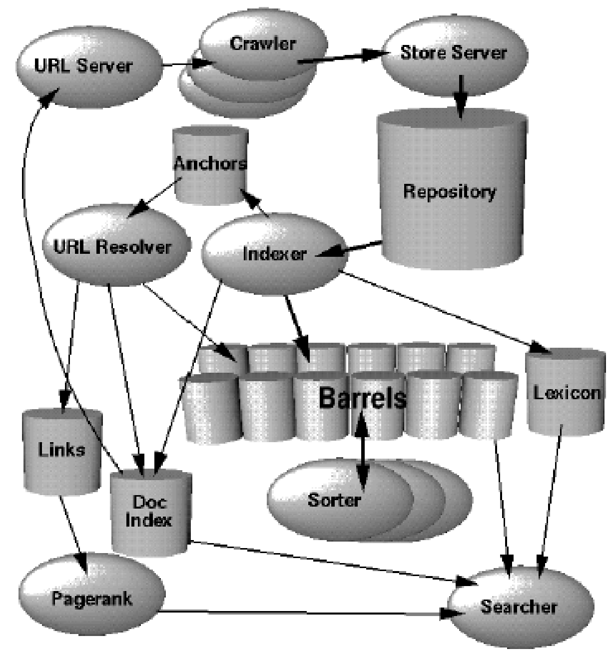
Link Analysis Algorithms

- Following are the ***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

Web Search Engine

"The Anatomy of a Large-Scale Hypertextual Web Search Engine"



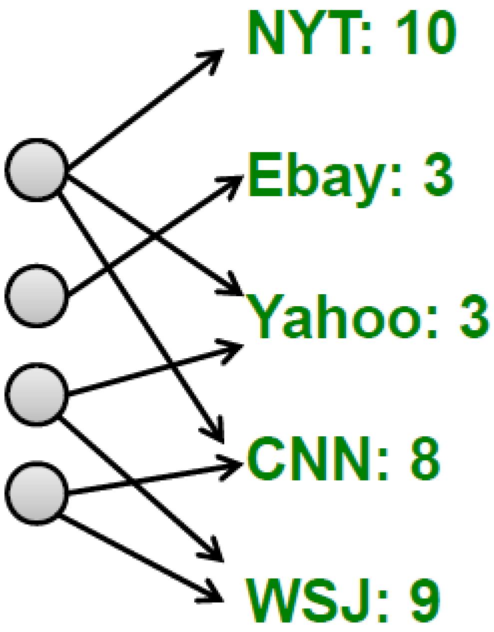
Sergey Brin and Lawrence Page, 1998

Hubs and Authorities
-
Hyperlink Induced Topic Search (HITS) Algorithm

Link Analysis

- **Goal** (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 an content (authority):**
    - Total sum of votes of experts
  - **Principle of repeated improvement**

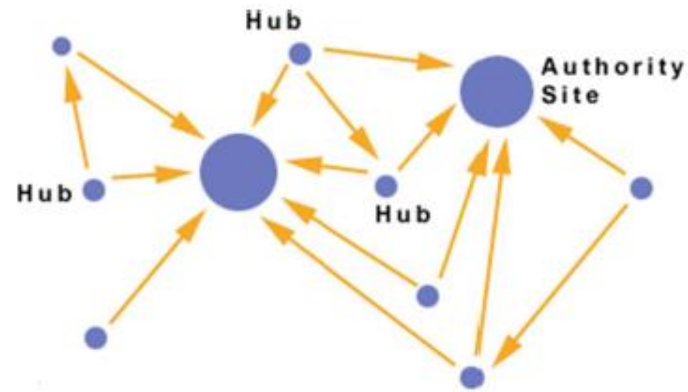


Hubs and Authorities

Interesting pages fall into two classes:

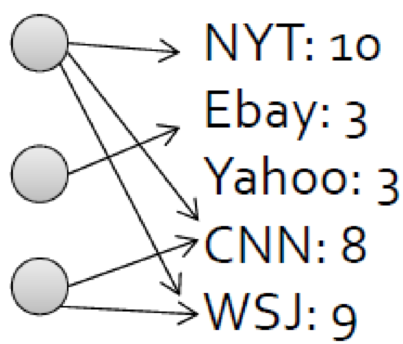
1. Authorities are pages containing useful information

- The prominent, highly endorsed answers to queries
- Newspaper home pages
- Course home pages
- Home pages of auto manufacturers

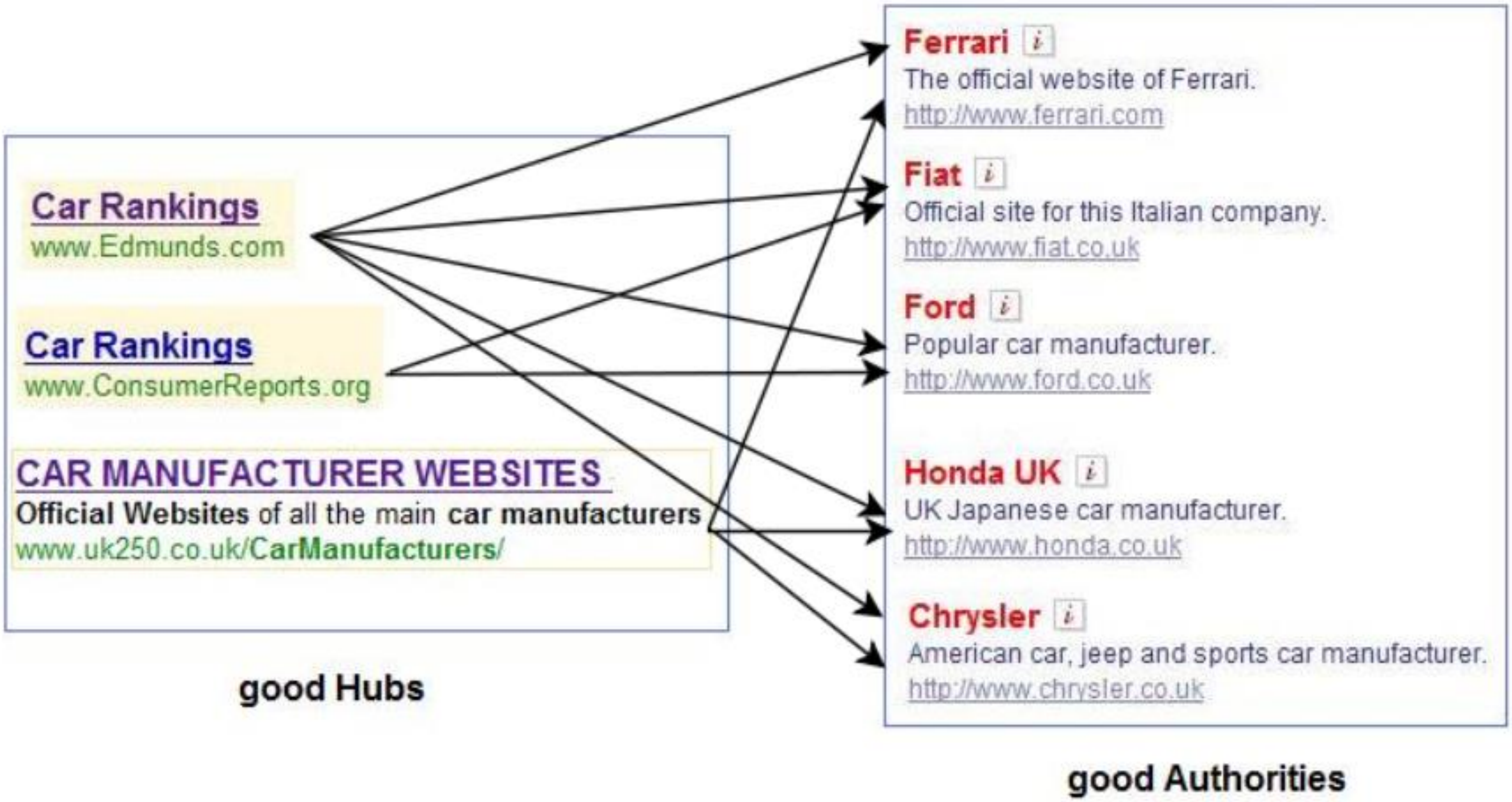


2. Hubs are pages that link to authorities

- High-value lists for queries
- List of newspapers
- Course bulletin
- List of U.S. auto manufacturers



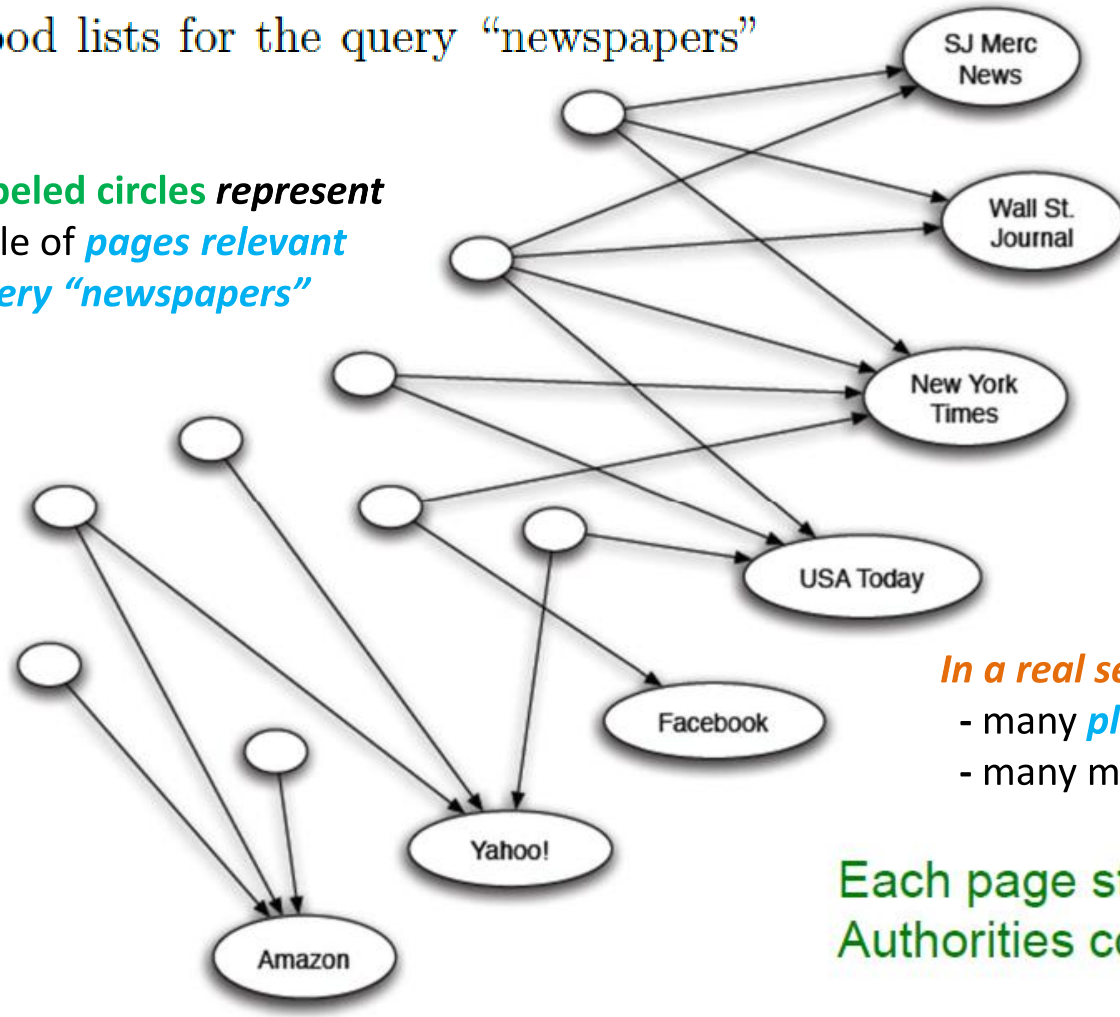
Hubs and Authorities



Query: Top automobile makers

Counting in-links: Authority

Finding good lists for the query "newspapers"



The **unlabeled circles** ***represent*** our sample of ***pages relevant to the query "newspapers"***

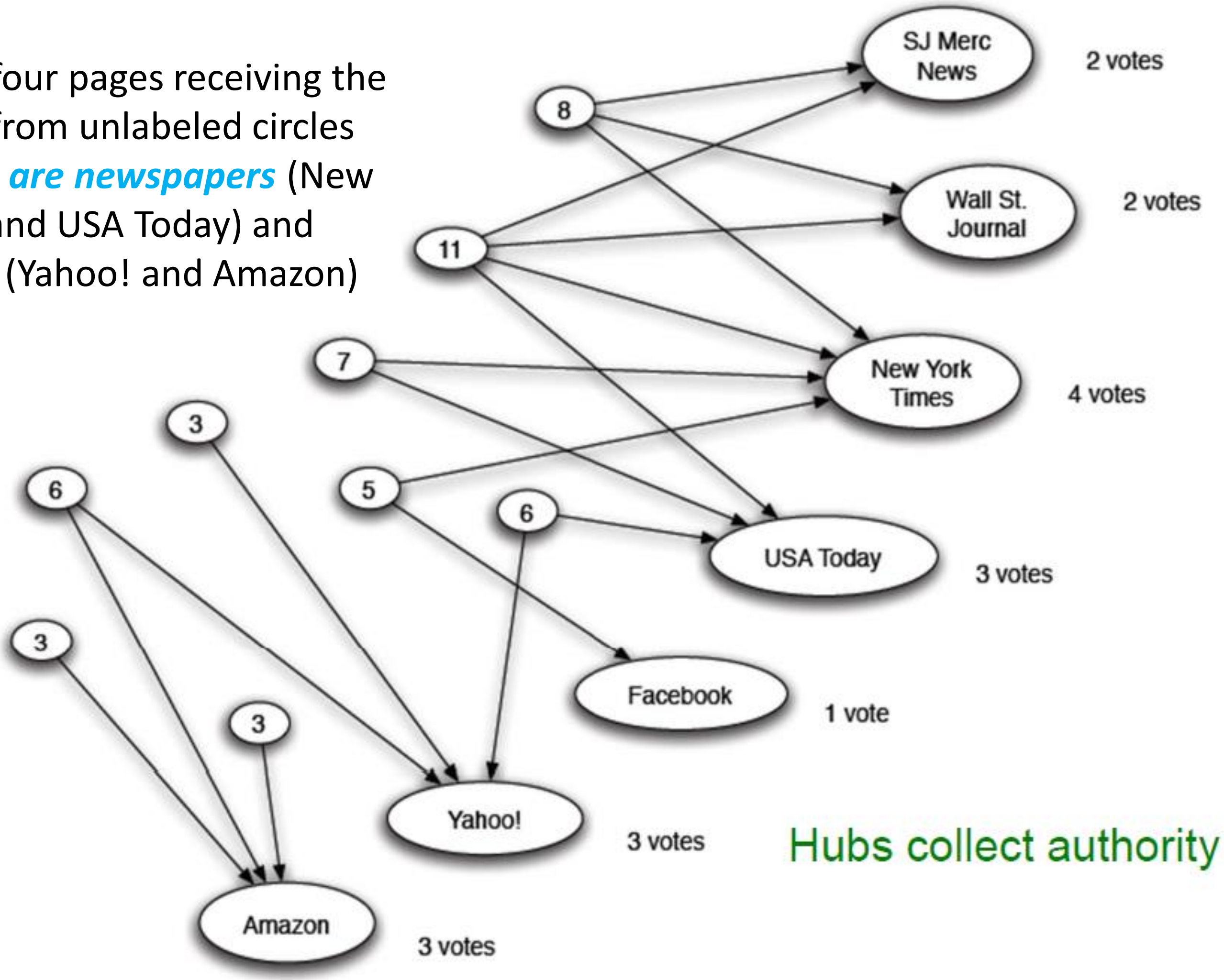
In a real setting, there would be
- many ***plausible newspaper pages***
- many more ***off-topic pages***

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 the authority score)

Expert Quality: Hub

Among the four pages receiving the most votes from unlabeled circles initially, ***two are newspapers*** (New York Times and USA Today) and ***two are not*** (Yahoo! and Amazon)

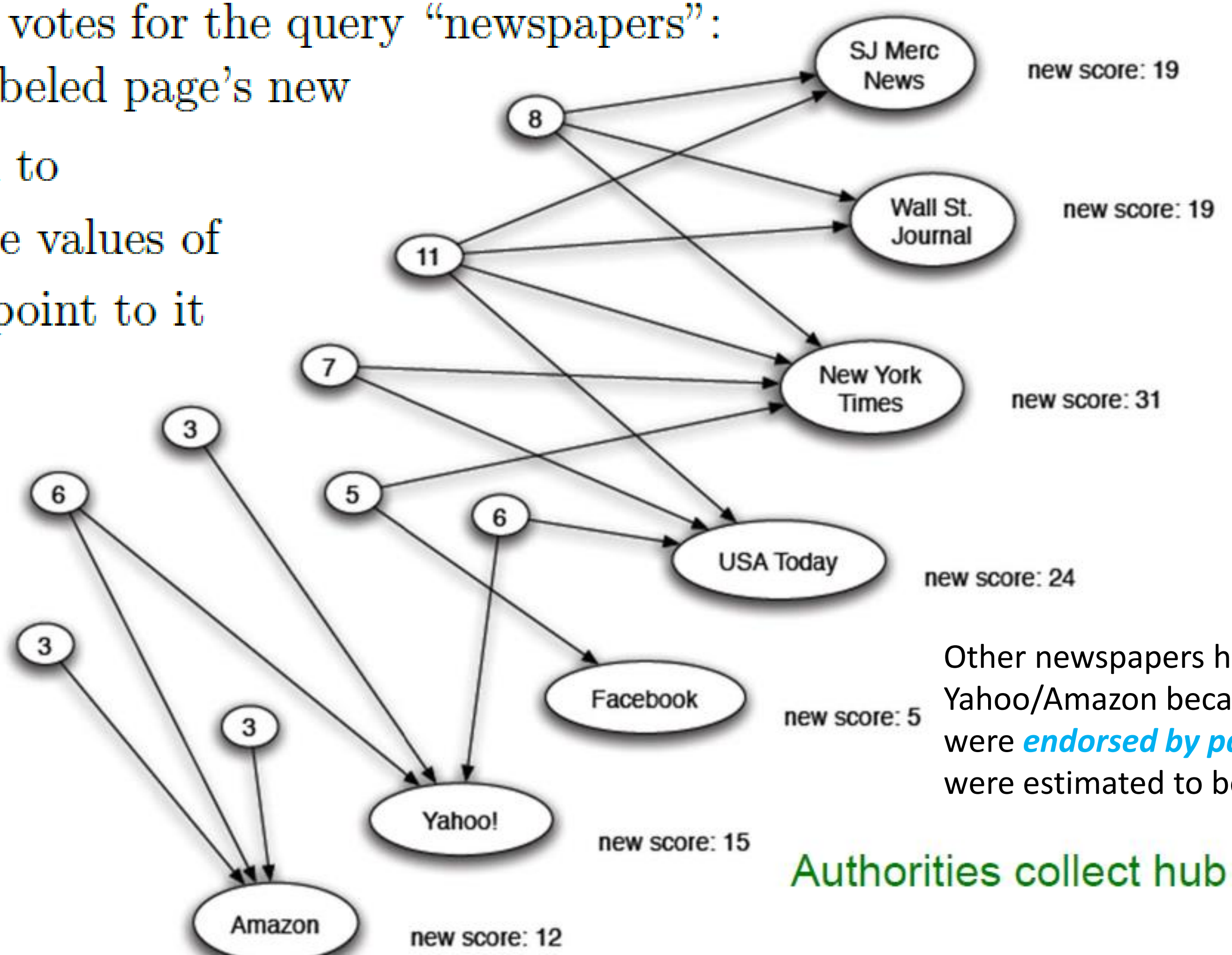


Hubs collect authority scores

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

Reweighting

Re-weighting votes for the query "newspapers": each of the labeled page's new score is equal to the sum of the values of all lists that point to it



Other newspapers have surpassed Yahoo/Amazon because they were ***endorsed by pages*** that were estimated to be ***good lists***

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

Hubs and Authorities

[Kleinberg '98]

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

Convergence criteria:

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

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

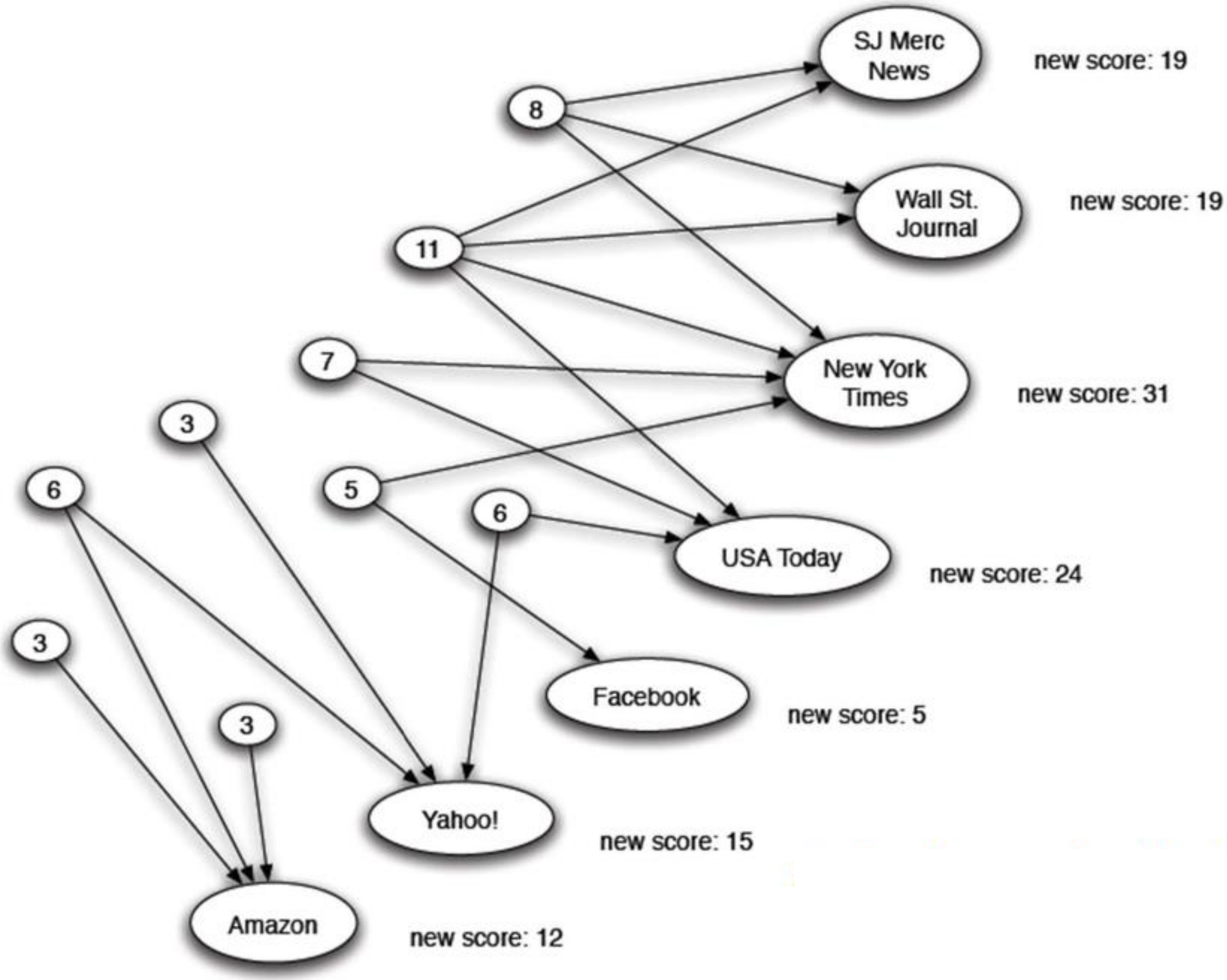
HITS algorithm:

- 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:

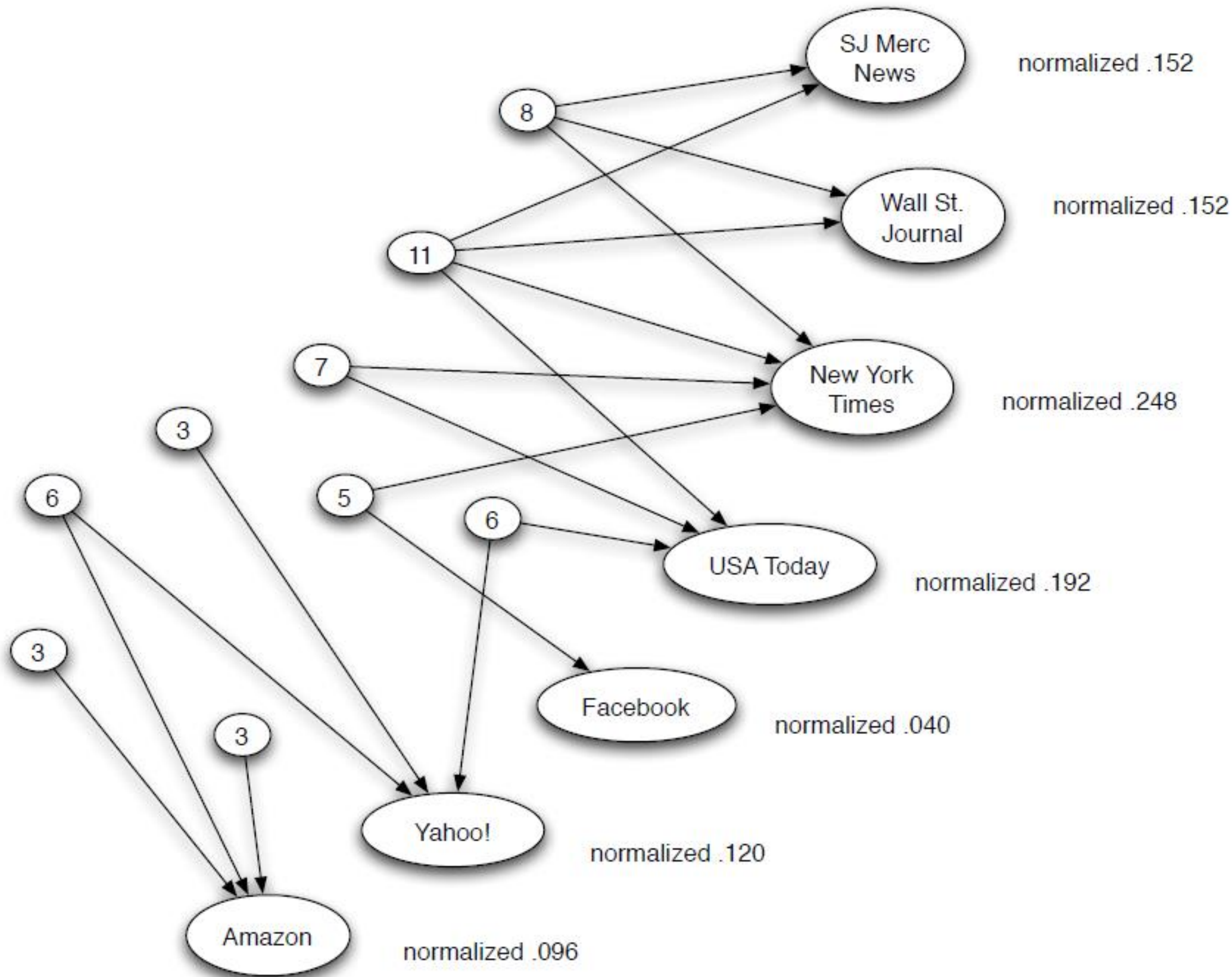
$$\sum_i \left(a_i^{(t+1)}\right)^2 = 1, \sum_j \left(h_j^{(t+1)}\right)^2 = 1$$

We choose a number of steps k

Reweighting



Reweighting and Normalization

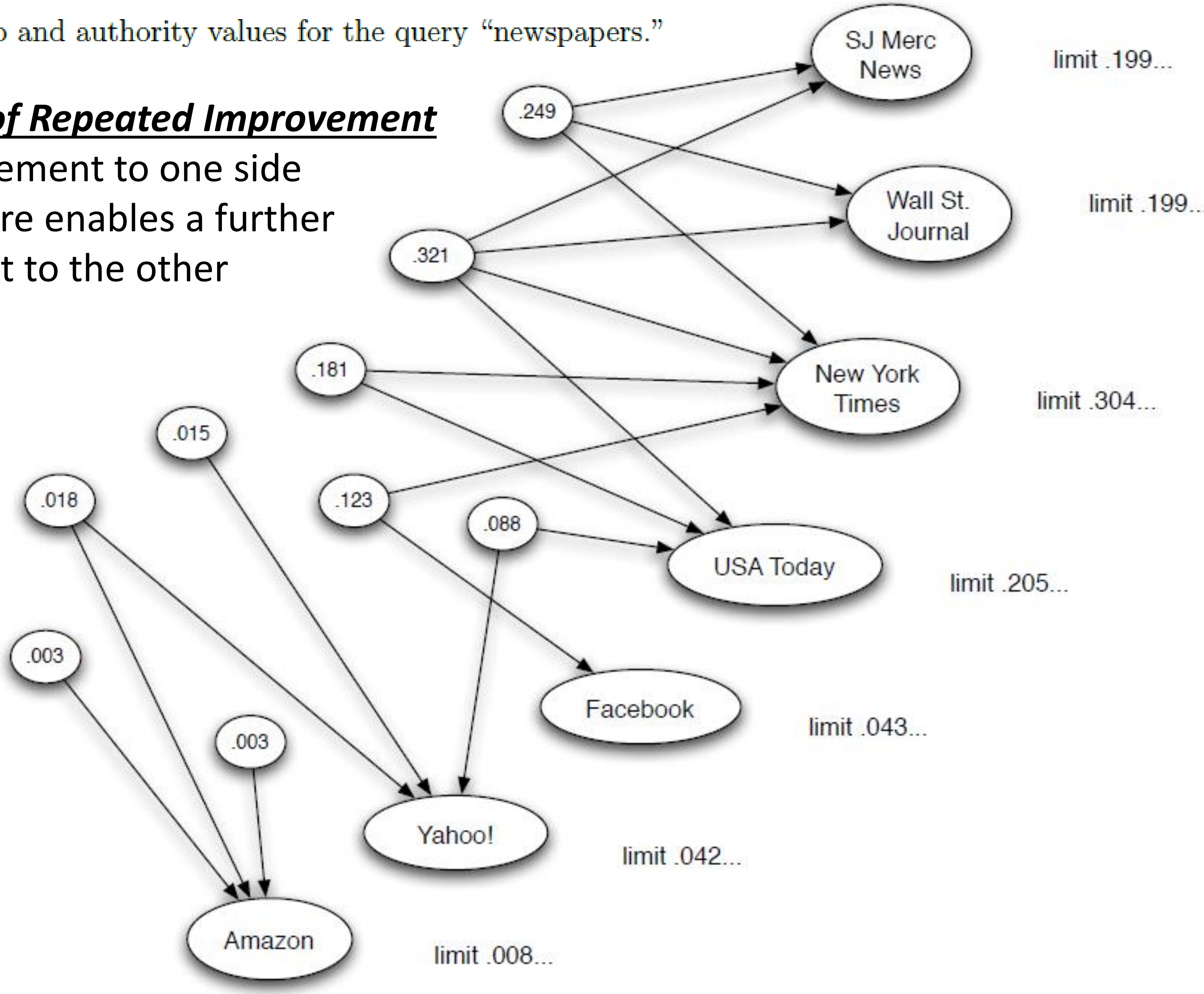


The Principle of Repeated Improvement

Limiting hub and authority values for the query "newspapers."

Principle of Repeated Improvement

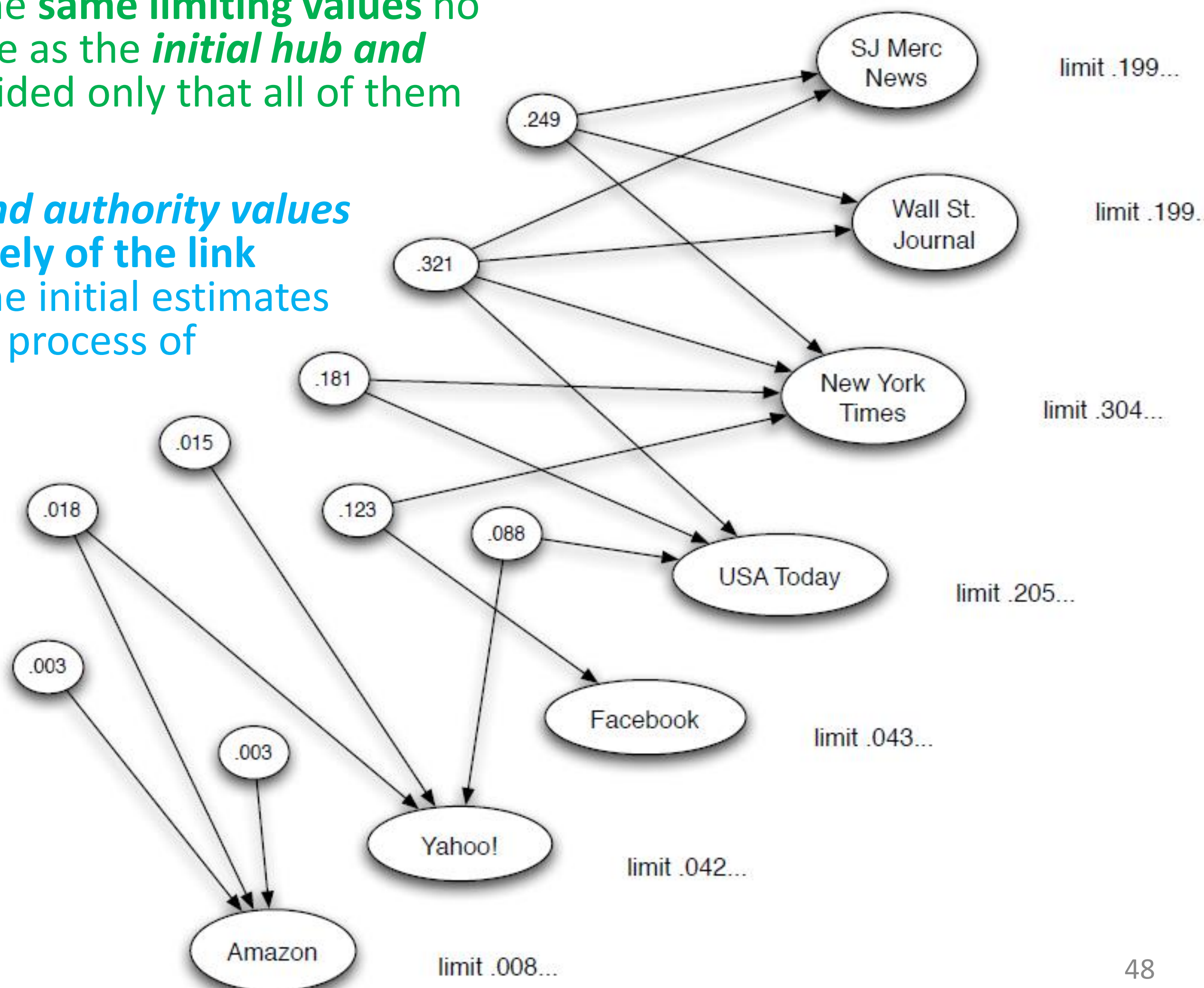
each refinement to one side
of the figure enables a further
refinement to the other



The Principle of Repeated Improvement

Except in a few rare cases (characterized by a certain kind of degenerate property of the link structure), we reach the **same limiting values** no matter what we choose as the ***initial hub and authority values***, provided only that all of them are positive.

⇒ the ***limiting hub and authority values*** are properties, **purely of the link structure**, not of the initial estimates we use to start the process of computing them.



Hubs and Authorities

[Kleinberg '98]

- **Hits in the vector notation:**
  - Vector $\boldsymbol{a} = (\boldsymbol{a}_1 \dots, \boldsymbol{a}_n)$, $\boldsymbol{h} = (\boldsymbol{h}_1 \dots, \boldsymbol{h}_n)$
  - Adjacency matrix $\boldsymbol{A}$ (n x n): $\boldsymbol{A}_{ij} = \mathbf{1}$ if $\boldsymbol{i} \to \boldsymbol{j}$
- **Can rewrite** $\boldsymbol{h}_i = \sum_{i \to j} \boldsymbol{a}_j$ **as** $\boldsymbol{h}_i = \sum_j \boldsymbol{A}_{ij} \cdot \boldsymbol{a}_j$
- **So:** $\boldsymbol{h} = \boldsymbol{A} \cdot \boldsymbol{a}$ **And similarly:** $\boldsymbol{a} = \boldsymbol{A}^T \cdot \boldsymbol{h}$
- **Repeat until convergence:**
  - $h^{(t+1)} = A \cdot a^{(t)}$
  - $a^{(t+1)} = A^T \cdot h^{(t)}$
  - Normalize $a^{(t+1)}$ and $h^{(t+1)}$

Hubs and Authorities

[Kleinberg '98]

- What is $a = A^T \cdot h$?
- Then: $\boldsymbol{a} = \boldsymbol{A}^T \cdot \underbrace{(\underbrace{\boldsymbol{A} \cdot \boldsymbol{a}}_{\text{new } h})}_{\text{new } a}$
- a is updated (in 2 steps):

  $$a = A^T(A\,a) = (A^T A)\,a$$

- h is updated (in 2 steps)

  $$h = A\,(A^T h) = (A\,A^T)\,h$$

- Thus, in $2k$ steps:

  $$a = (A^T \cdot A)^k \cdot a$$

  $$h = (A \cdot A^T)^k \cdot h$$

Repeated matrix powering

Hubs and Authorities

[Kleinberg '98]

- **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:**
  - $\boldsymbol{A}^{\boldsymbol{T}} \cdot \boldsymbol{A} \cdot \boldsymbol{a} = \boldsymbol{c}' \cdot \boldsymbol{a}$
  - $\boldsymbol{A} \cdot \boldsymbol{A}^{\boldsymbol{T}} \cdot \boldsymbol{h} = \boldsymbol{c}'' \cdot \boldsymbol{h}$

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

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

In order to get a set, rich in both hubs and authorities for a query Q;

- First collect the top 200 documents that contain the highest number of occurrences of the search phrase *Q*.
  - These, may not be of tremendous practical relevance, but one has to start somewhere
    - a page that repeats the phrase a lot of times
- Pages from this set called ***root*** **(R_Q)** are essentially very heterogeneous and in general contain only a few (if any) links to each other
  - Web subgraph determined by these nodes is almost totally disconnected
    - in particular, we can not enforce Page Rank techniques on R_Q.
- ***Authorities*** for the query *Q* are not extremely likely to be in the root set R_Q
  - However, they are likely to be pointed out by at least one page in R_Q
  - It makes sense to extend the subgraph R_Q by including all edges coming from or pointing to nodes from R_Q
- Resulting subgraph is denoted by S_Q **(*seed*)** and call it the *seed* of the search
  - Notice that S_Q we have constructed is a reasonably small graph
    - certainly much smaller then the billions of nodes in the web graph!
    - likely to contain a lot of authoritative sources for *Q*

HITS: Hyperlink-Induced Topic Search

It was designed to ***score*** ***focused subgraph***

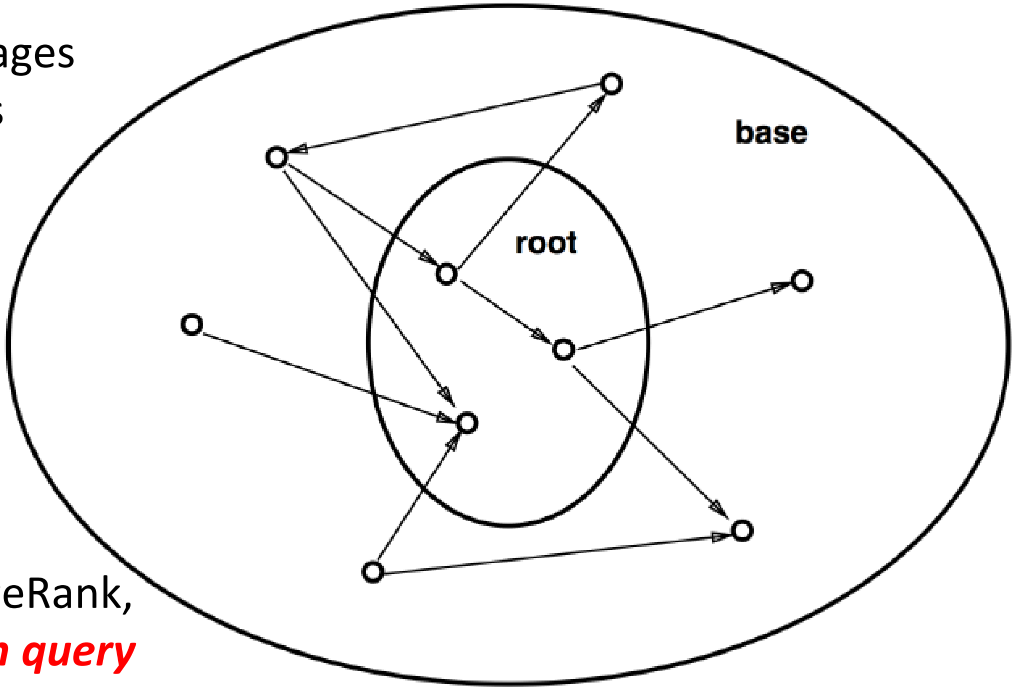
- take the search query and send it to search engine and get ***search results***
- get the ***connections*** between those pages
- ***expand*** set a little bit by adding nodes that are pointed by/pointed to the nodes in this set
- ***query dependent***: scores resulting from the link analysis are influenced by the search terms

=> So we get ***local rankings***

In practice

- used not on the whole web like PageRank, but only on the ***result set of a given query***
- developed for IBM Clever project
- variant was said to be used by Teoma which was acquired by Ask.com

Focused subgraph of WWW



Expanding the root set into a base set

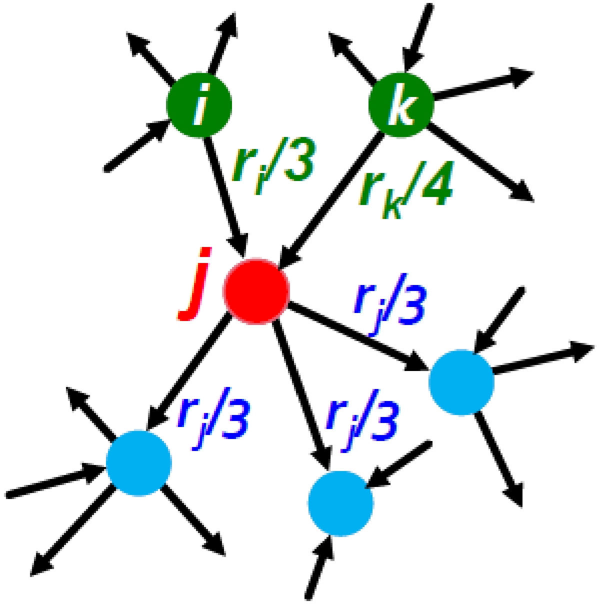
PageRank Algorithm

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 has 23,400 in-links
  - www.joe-schmoe.com has 1 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



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

If a page has no out-going links, it passes all its current PageRank to itself

PageRank: The "Flow" Model

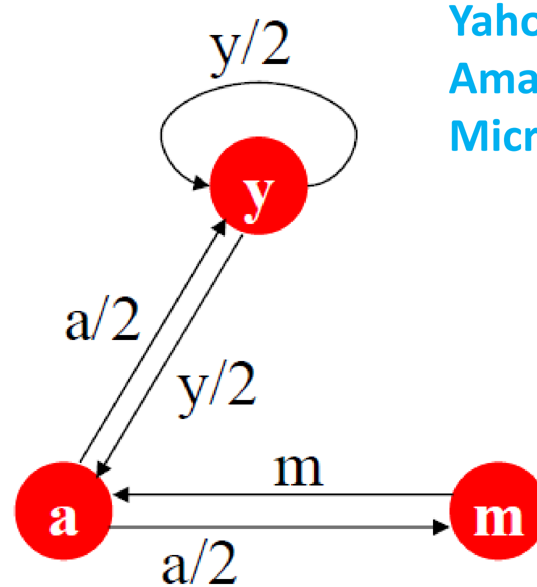
- **A page is important if it is pointed to by other important pages**
- **Define a "rank" r_j for node j**

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

d_i … out-degree of node i

The web in 1839

Yahoo
Amazon
Microsoft

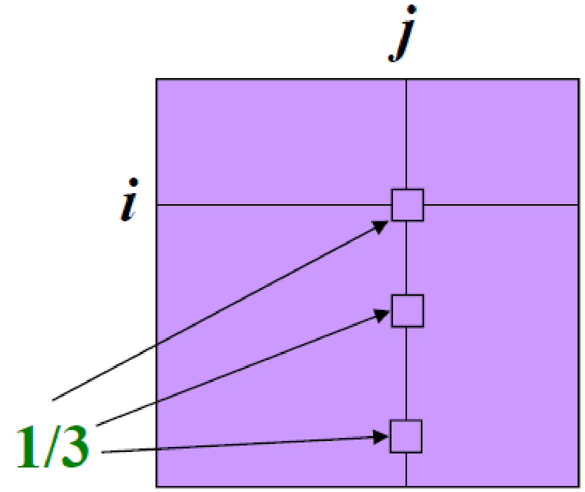


"Flow" equations:

$$\mathbf{r}_y = \mathbf{r}_y/2 + \mathbf{r}_a/2$$
$$\mathbf{r}_a = \mathbf{r}_y/2 + \mathbf{r}_m$$
$$\mathbf{r}_m = \mathbf{r}_a/2$$

PageRank: Matrix Formulation

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



M

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

$$r = M \cdot r$$

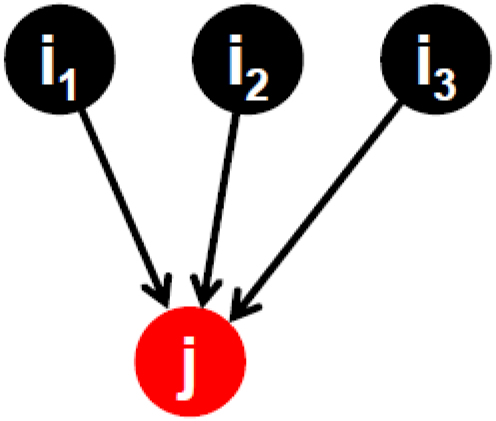
$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_i}$$

Solving the Equations

- Because there are ***no constant terms***, the equations ***$v = Mv$*** do ***not have a unique solution***
  - i.e., doubling each component of solution v yields another solution
- In ***Web-sized examples***, we cannot solve by Gaussian elimination (direct method)
  - ***too slower*** and require ***huge storage spaces***
  - we need to use ***relaxation*** (= iterative solution)
    - indirect method

Random Walk Interpretation

- **Imagine a random web surfer:**
  - At any time t, surfer is on some page i
  - At time $t+1$, the surfer follows an out-link from i uniformly at random
  - Ends up on some page j linked from i
  - Process repeats indefinitely

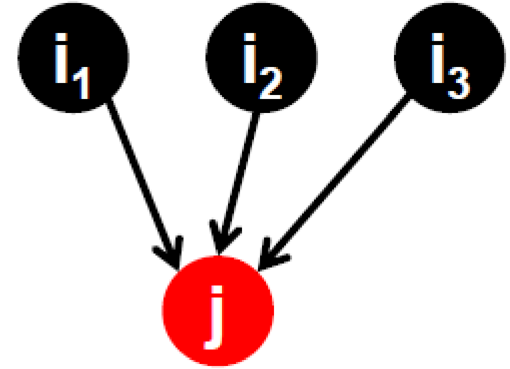


$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_{\mathrm{out}}(\mathrm{i})}$$

Such an exploration of nodes performed by randomly following links is called a ***random walk*** on the network

- **Let:**
  - $\boldsymbol{p(t)}$ … vector whose i^{th} coordinate is the prob. that the surfer is at page i at time $\boldsymbol{t}$
  - So, $\boldsymbol{p(t)}$ is a probability distribution over pages

The Stationary Distribution



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

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

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

- Suppose the random walk reaches a state

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

  => then $\boldsymbol{p}(\boldsymbol{t})$ is ***stationary distribution*** of a random walk

- **Our original rank vector** $\boldsymbol{r}$ satisfies $\boldsymbol{r} = \mathbf{M} \cdot \boldsymbol{r}$
  - So, $\boldsymbol{r}$ is a stationary distribution for the random walk

- ***PageRank of a page j*** is the ***limiting probability*** that a ***random walk across hyperlinks*** will end up at j, as we run the walk for larger and larger numbers of steps

PageRank: How to solve?

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

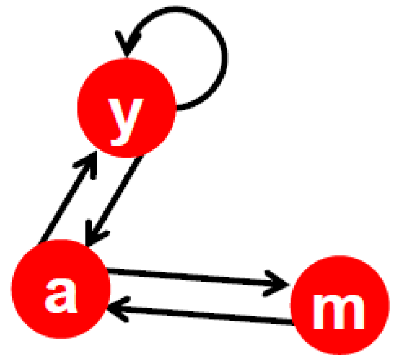
- Assign each node an initial page rank
- Repeat until convergence ($\Sigma_i \, |r_i^{(t+1)} - r_i^{(t)}| < \varepsilon$)
  - Calculate the page rank of each node

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

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**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

- **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, …

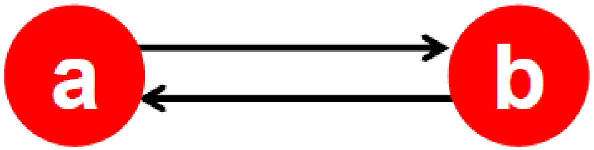
PageRank: Three Questions

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_\mathrm{i}} \quad \text{or equivalently} \quad r = Mr$$

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

Does this converge?

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



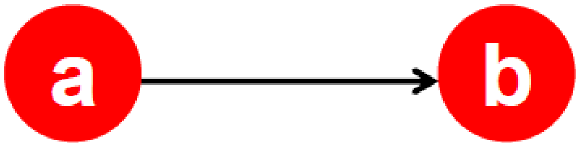
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

- **Example:**

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

Does it converge to what we want?

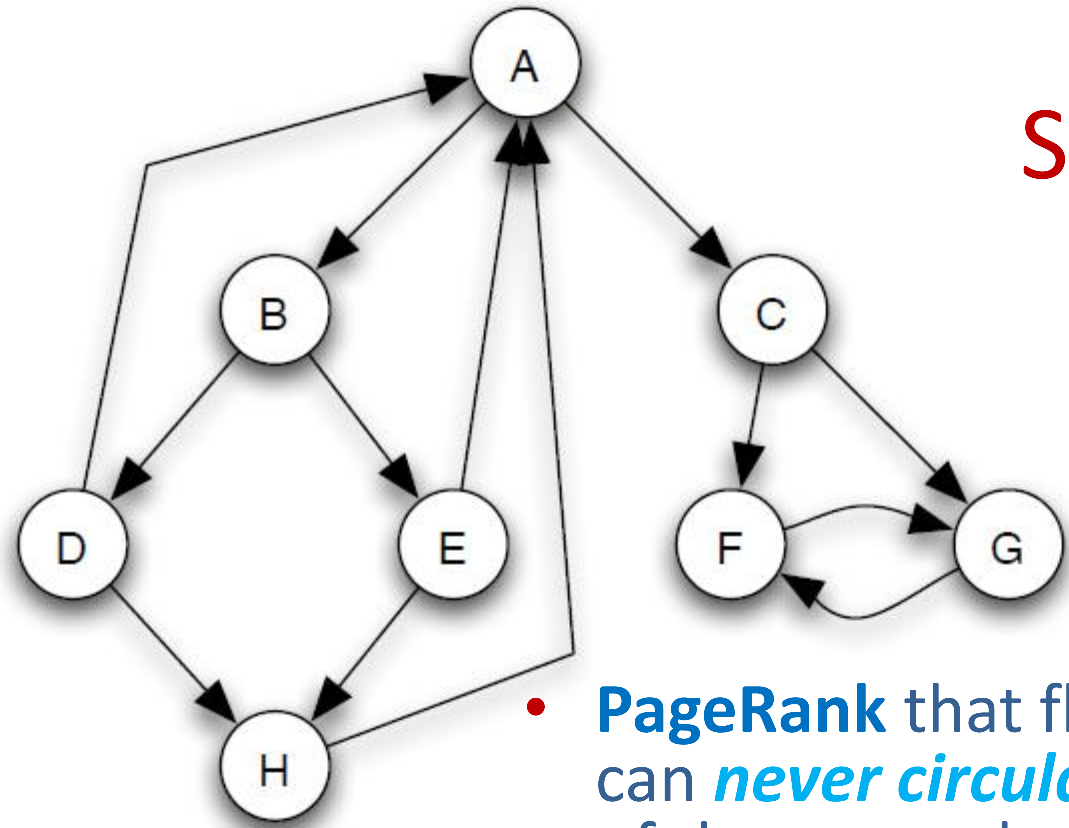
- **The "Dead end" problem:**



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

- **Example:**

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



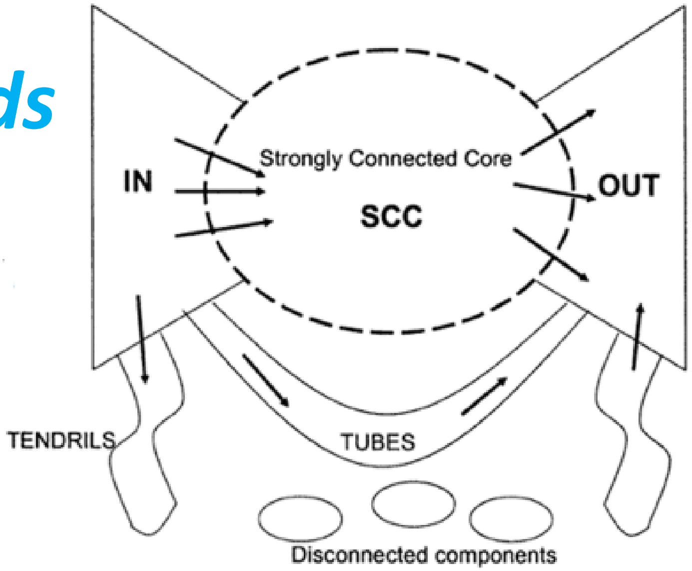
Spider Trap & Dead End

- **PageRank** that flows from C to F & G can ***never circulate back*** into the rest of the network
  - So the links out of C, function as a kind of ***"slow leak"*** that eventually causes all the PageRank to end up at F and G
- Repeatedly running the algorithm, we converge to PageRank values of **1/2 for each of F and G**, and ***0 for all other nodes***

PageRank: Problems

2 Problems:

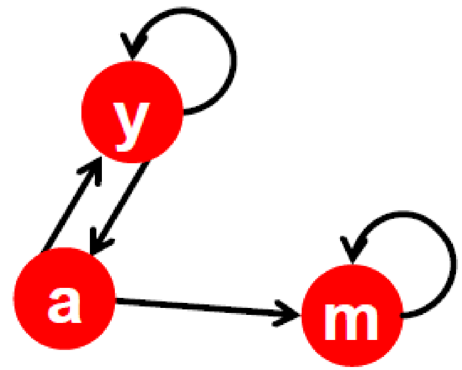
- **(1)** Some pages are ***dead ends*** (have no out-links)
  - Such pages cause importance to ***"leak out"***



- **(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

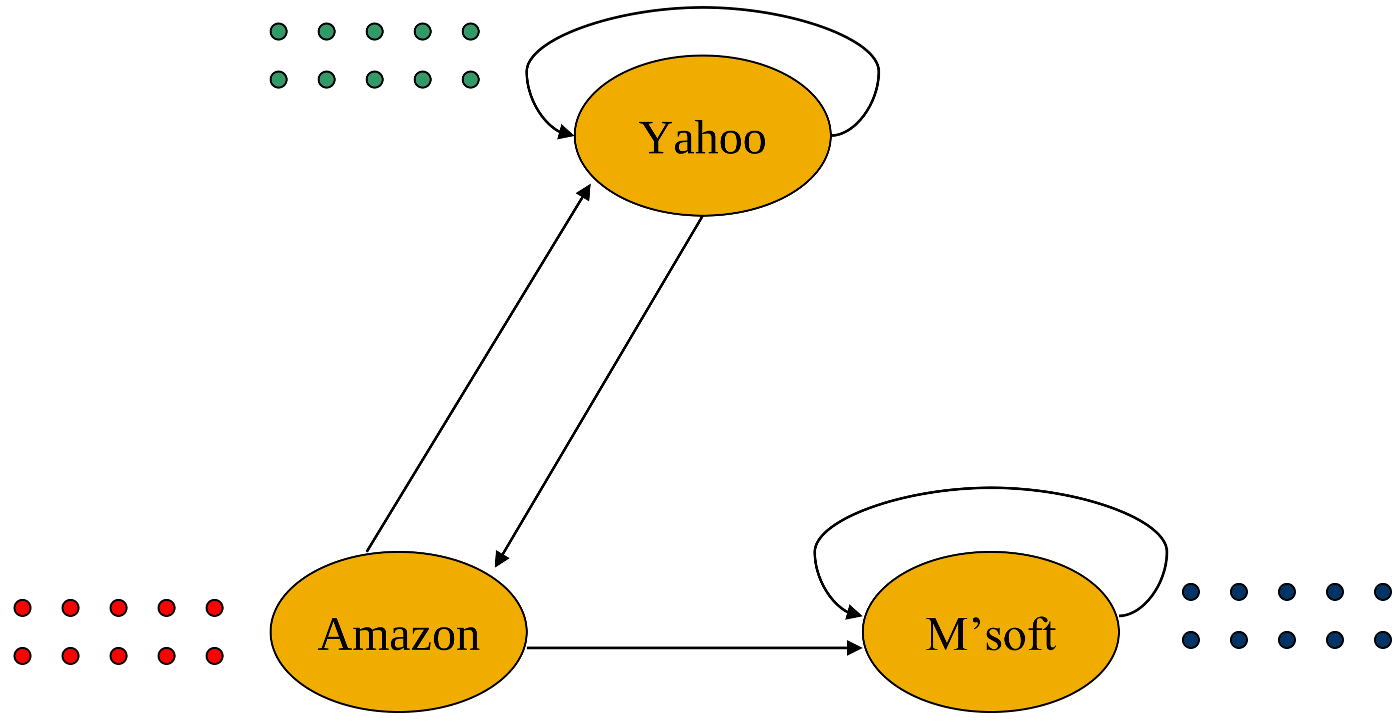
$$\mathbf{r_y} = \mathbf{r_y}/2 + \mathbf{r_a}/2$$
$$\mathbf{r_a} = \mathbf{r_y}/2$$
$$\mathbf{r_m} = \mathbf{r_a}/2 + \mathbf{r_m}$$

- **Example:**

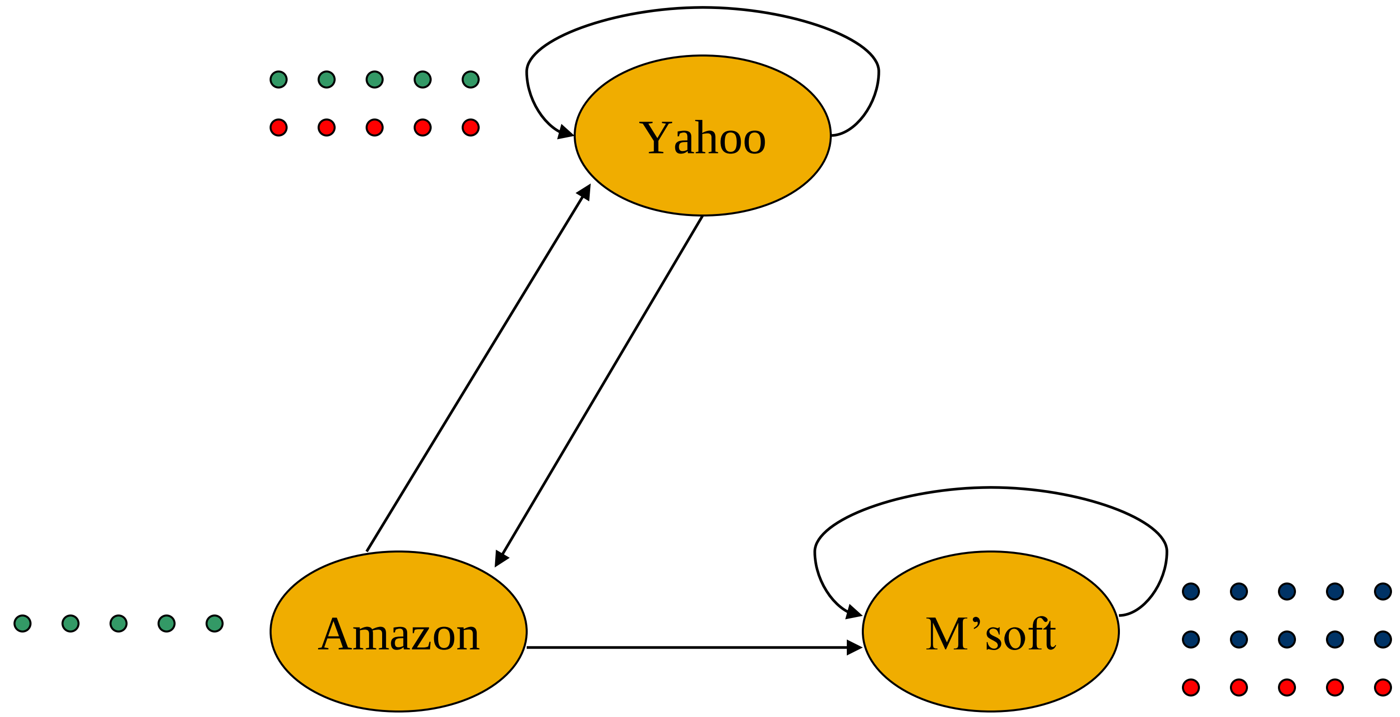
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \quad \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix} \; \Bigg| \; \begin{matrix} 2/6 \\ 1/6 \\ 3/6 \end{matrix} \; \Bigg| \; \begin{matrix} 3/12 \\ 2/12 \\ 7/12 \end{matrix} \; \Bigg| \; \begin{matrix} 5/24 \\ 3/24 \\ 16/24 \end{matrix} \quad \ldots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

Iteration 0, 1, 2, …

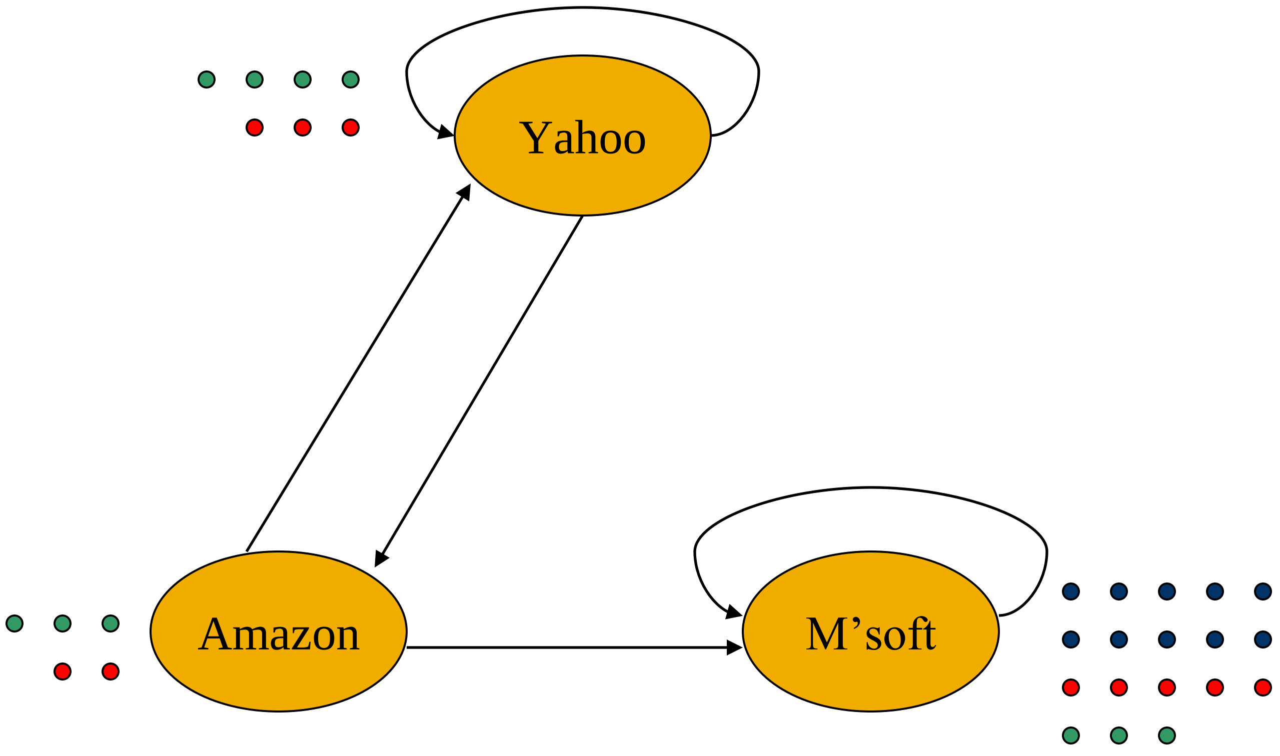
Microsoft Becomes a Spider Trap



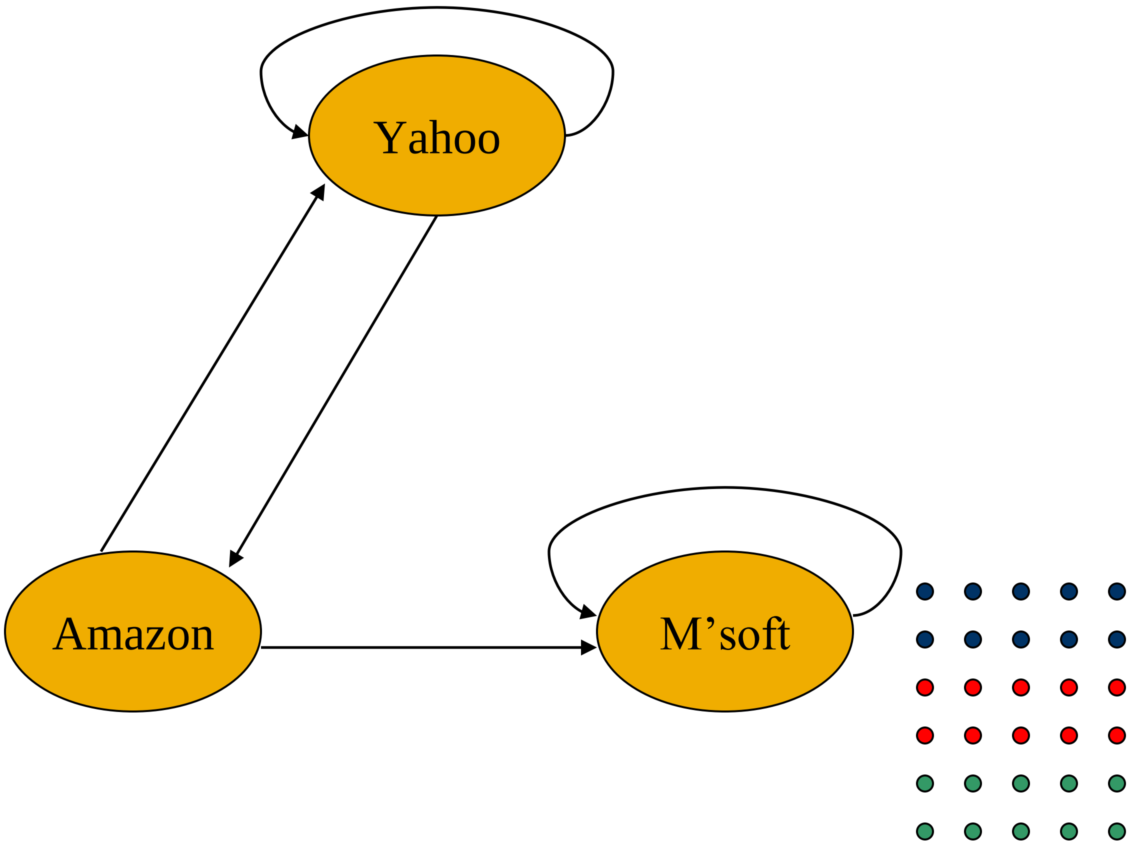
Microsoft Becomes a Spider Trap



Microsoft Becomes a Spider Trap



Microsoft a Spider Trap – In the Limit

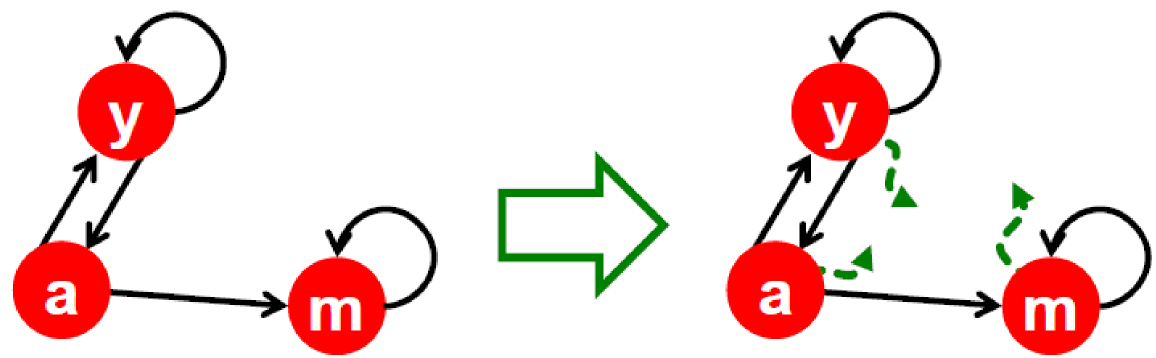


Solution: Random Teleports

- **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**

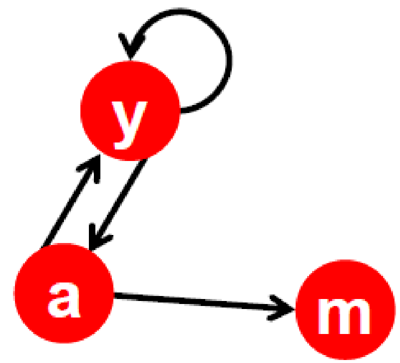
Scaled PageRank Update Rule

divide the residual $1-\beta$ units of PageRank equally over all nodes, giving $(1-\beta)/n$ to each



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

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

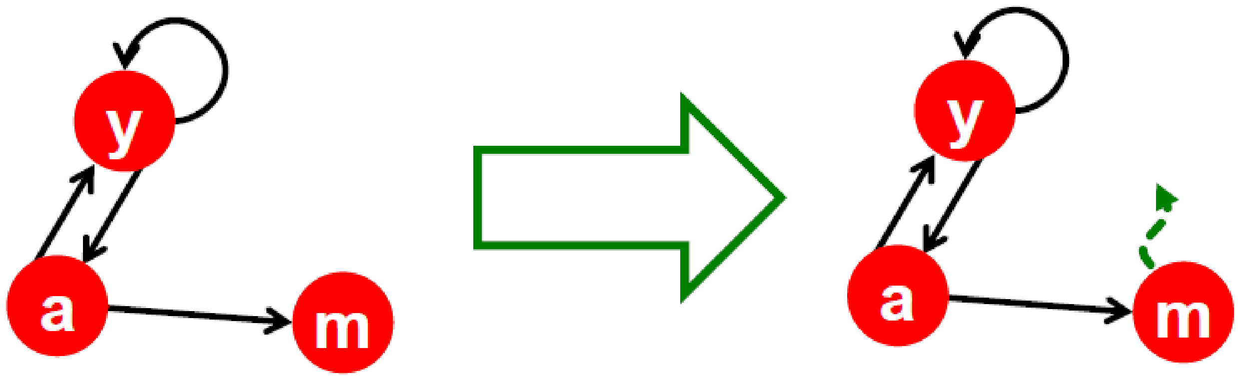
- **Example:**

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

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]

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

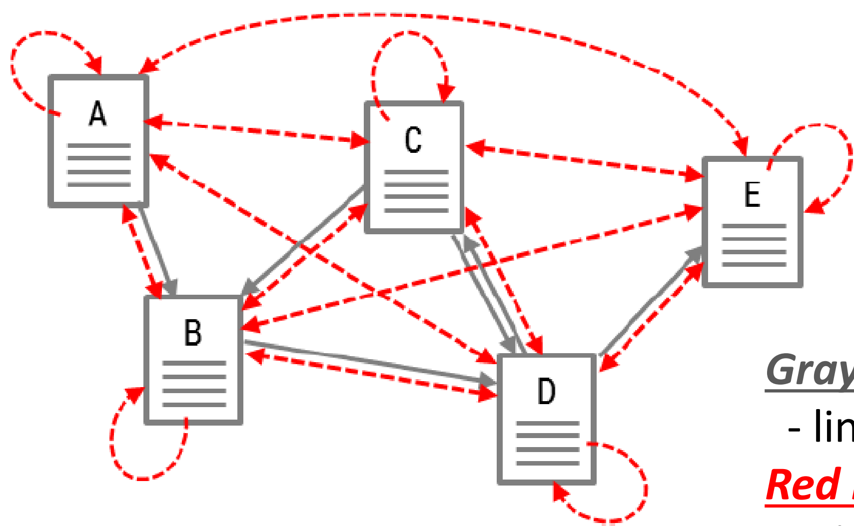
d_i ... out-degree of node i

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.

Page Rank

"PageRank can be thought of as a model of user behavior. We assume there is a "random surfer" who is given a web page at random and keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page. The **probability** that the random surfer visits a page is its **PageRank**."

$$PR(A) = (1 - d) + d(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$



Gray links

- links that exist on the pages

Red links

- virtual links that you add to the graph

Sergey Brin and Larry Page, 1998

PageRank & Eigenvectors

- **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 (**):**

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

  d_i … out-degree of node i

- **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 $1 - \beta$?**
  - In practice 0.15 (Jump approx. every 5-6 links)

Note: M is a sparse matrix but 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:**

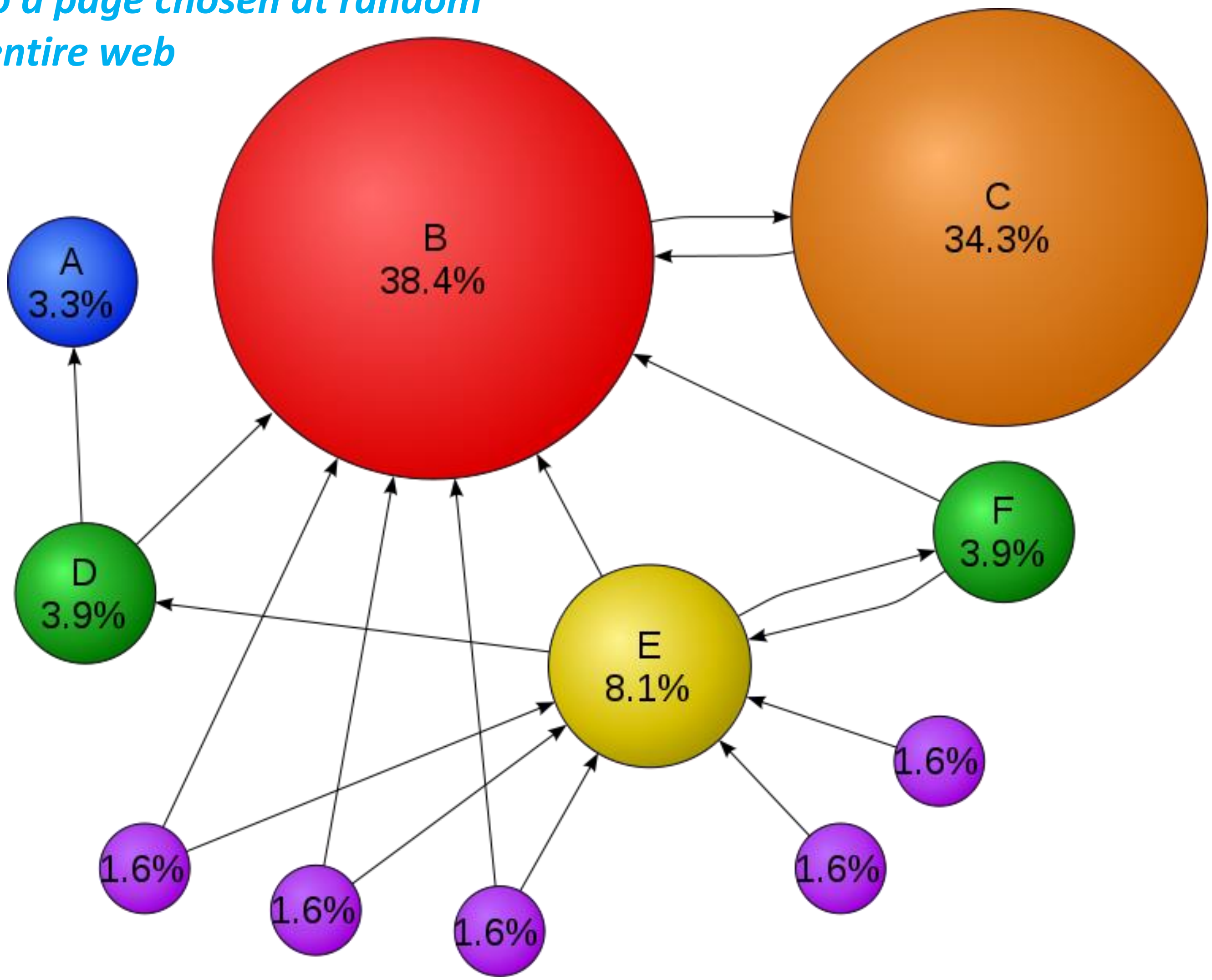
      $\forall \boldsymbol{j}$: $\boldsymbol{r}_j^{(t)} = \boldsymbol{r'}_j^{(t)} + \boxed{\frac{\boxed{1-S}}{N}}$ where: $S = \sum_j r'^{(t)}_j$
    - $\boldsymbol{t = t + 1}$
  - **while** $\sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| > \varepsilon$

Leaked Amount

Re-insert back the leaked amount to all nodes uniformly

Example: PageRank

85% likelihood of choosing a random link from the page they are currently visiting,* and a *15% likelihood of jumping to a page chosen at random from the entire web



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**

PageRank beyond the Web

1. GeneRank
2. ProteinRank
3. FoodRank
4. SportsRank
5. HostRank
6. TrustRank
7. BadRank
8. ObjectRank
9. ItemRank
10. ArticleRank
11. BookRank
12. FutureRank
13. TimedPageRank
14. SocialPageRank
15. DiffusionRank
16. ImpressionRank
17. TweetRank
18. TwitterRank
19. ReversePageRank
20. PageTrust
21. PopRank
22. CiteRank
23. FactRank
24. InvestorRank
25. ImageRank
26. VisualRank
27. QueryRank
28. BookmarkRank
29. StoryRank
30. PerturbationRank
31. ChemicalRank
32. RoadRank
33. PaperRank
34. Etc…

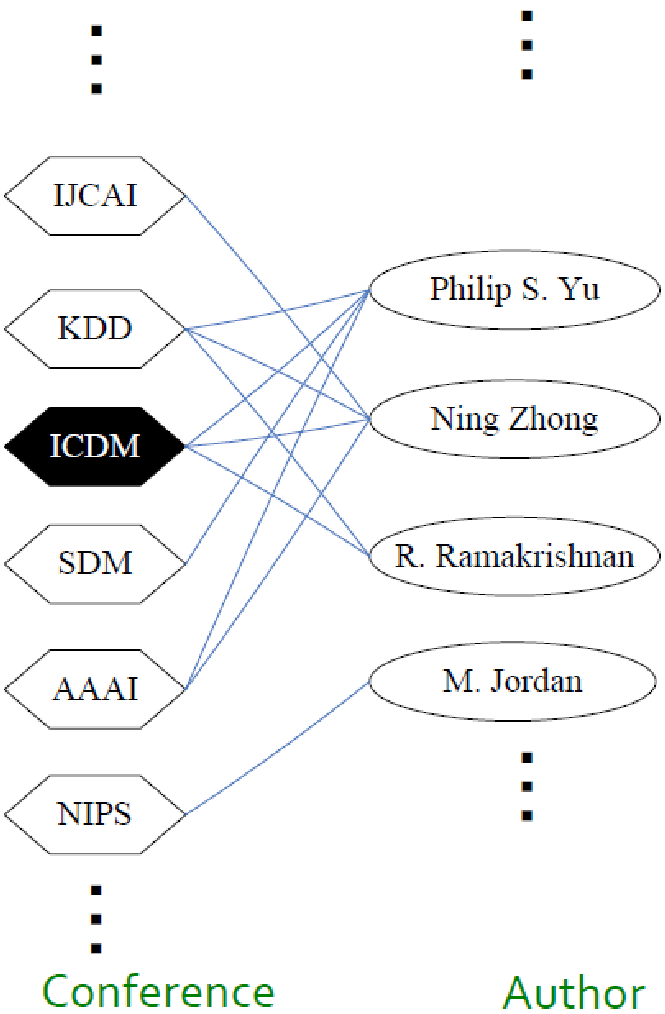
from David Gleich

Personalized PageRank

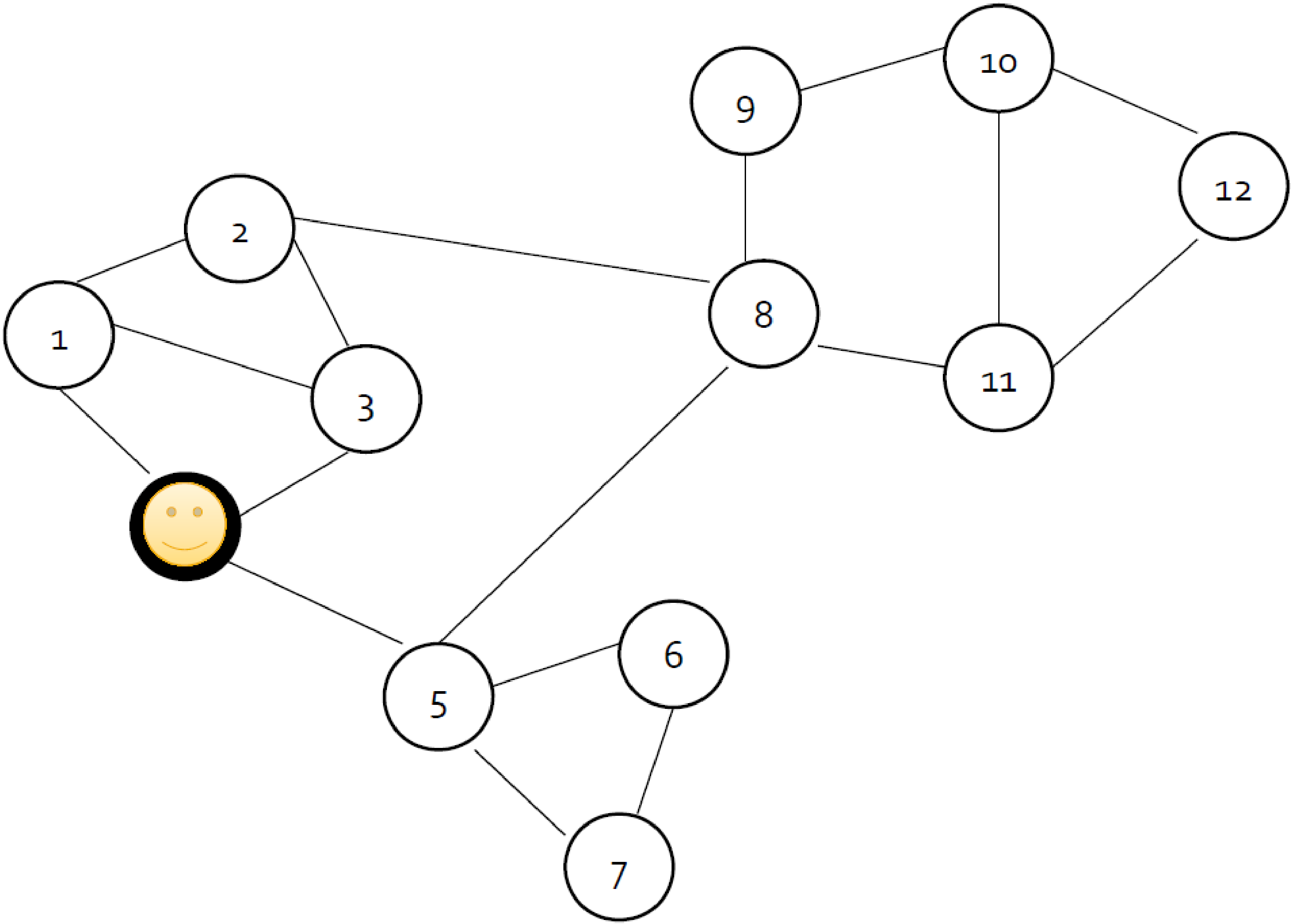
Random Walk with Restarts

Example Application: Graph Search

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



Random Walk with Restarts



Personalized PageRank

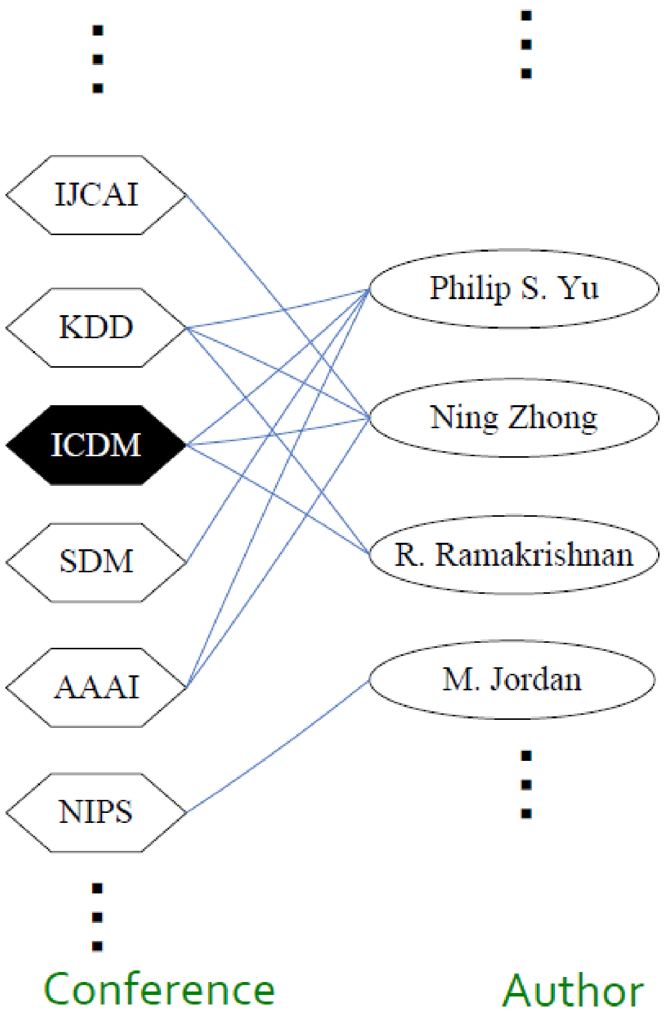
- ***Goal:*** Evaluate pages not just by popularity but by how close they are to the topic
- ***Teleporting can go to:***
  - **Any page with equal probability**
    - (we used this so far)
  - **A topic-specific set of "relevant" pages**
    - Topic-specific (personalized) PageRank **(S ...teleport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

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

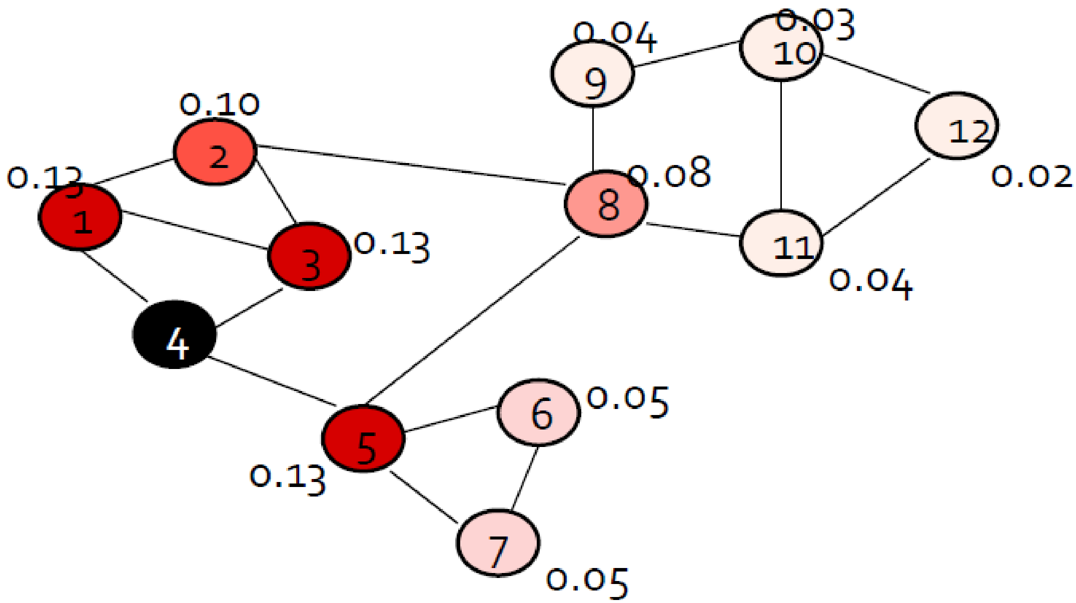
- **Random Walk with Restart:** ***S is a single element***

PageRank: Applications

- **Graphs and web search:**
  – Ranks nodes by "importance"
- **Personalized PageRank:**
  – Ranks proximity of nodes to the teleport nodes
- **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

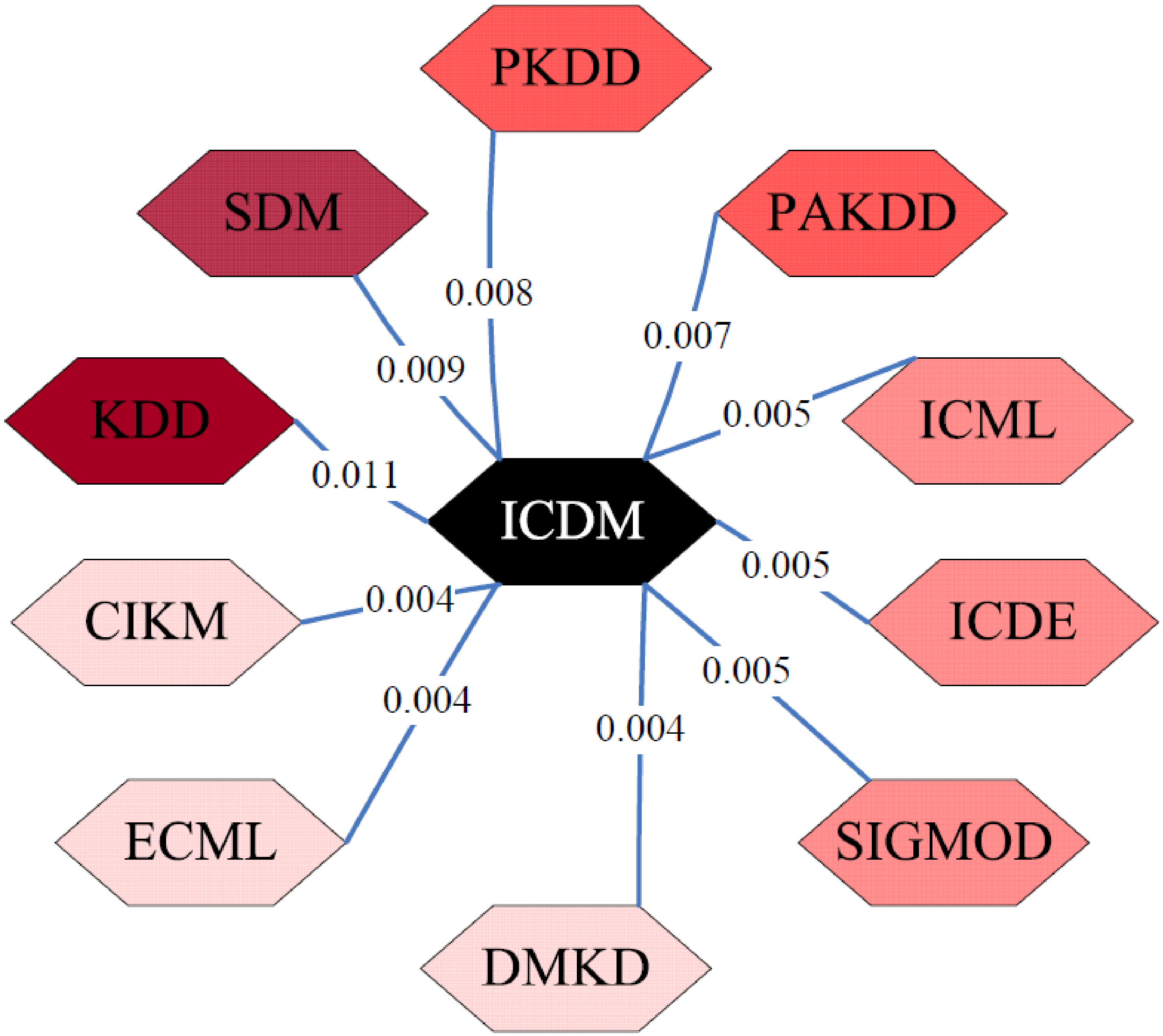
Nearby nodes, higher scores

More red, more relevant

Ranking vector

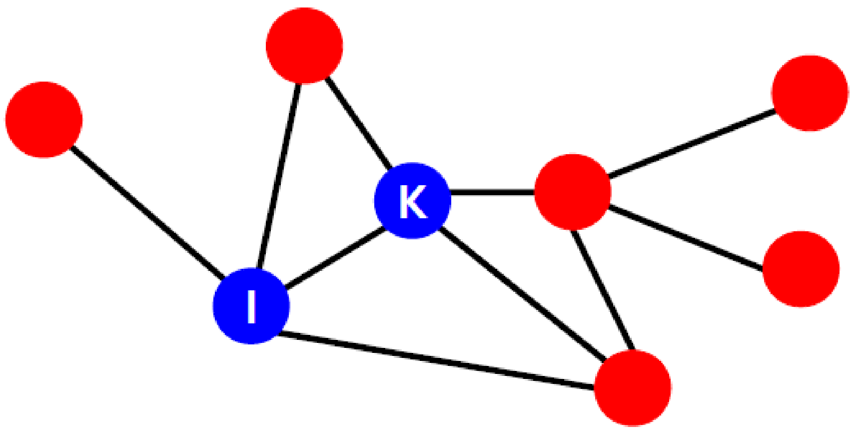
$\vec{r}_4$

Most Related Conferences to ICDM



Personalized PageRank

- ***Q: Which conferences are closest to KDD & ICDM?***
- **A: Personalized PageRank with teleport set**
  S={KDD, ICDM}



Graph of CS conferences

Application: Automatic Image Captioning

PROBLEM 1 (AUTO-CAPTIONING). *Given a set S of color images, each with caption words; and given one more, uncaptioned image I, find the best t (say, t=5) caption words to assign to it.*

I_1 ("sea", "sun", "sky", "waves")

Image regions are extracted by a standard image segmentation algorithm

I_2 ("cat", "forest", "grass", "tiger")

I_3 - no caption

I_1 ("sea", "sun", "sky", "waves")

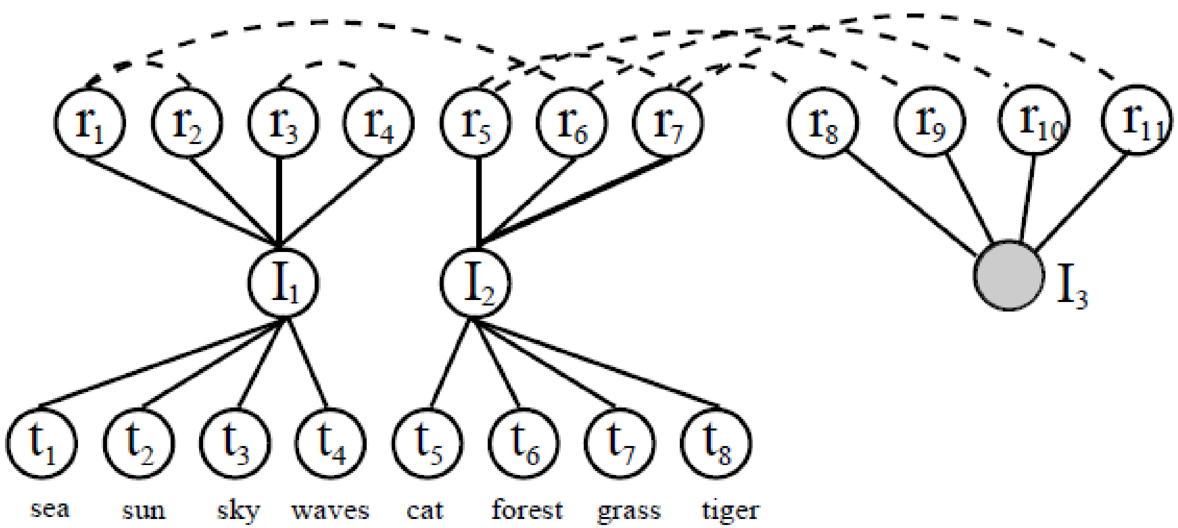
Features

- Each region is mapped into a ***30-dim feature vector***
- The p=30 ***features extracted from each region***
  - the mean and standard deviation of RGB values
  - average responses to various texture filters
  - its position in the entire image layout
  - shape descriptors
    - e.g., major orientation, or bounding region to real region area ratio

Mixed Media Graph

[Pan et. al.v2]

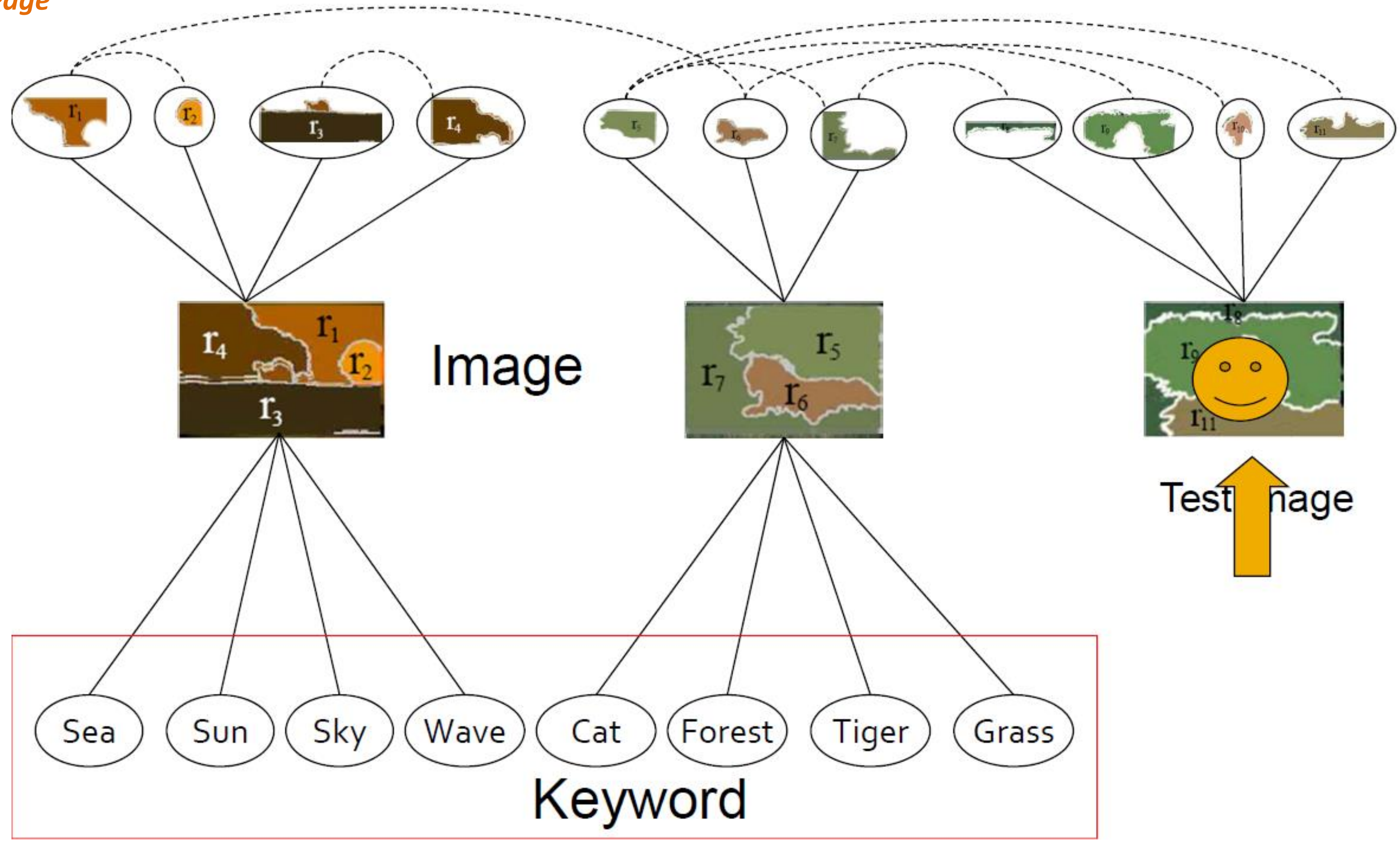
- Consider the image set S = {I1; I2; I3}
  - The graph corresponds to this data set has ***three types of nodes***:
    - one for the ***image objects i_j's*** (j = 1, 2, 3)
    - one for the ***regions*** r_j's (j = 1, …, 11)
    - one for the ***terms*** {t_1, … , t_8} = {sea, sun, sky, waves, cat, forest, grass, tiger}
  - ***Solid arcs*** indicate the **object-attribute-value relationships**
    - ***OAV-links:*** between an object node and an attribute value node
  - ***Dashed arcs*** indicate ***nearest-neighbor relationships***
    - ***NN-links:*** between the nodes of two similar domain tokens
    - ***k=1 in this example***



Automatic Image Captioning

[Pan et. al.v2]

orange ***tiger region*** r_6 and orange ***sky region*** r_1 have feature vectors that are close in Euclidean distance => **V(r_1) and V (r_6)** are ***connected by an edge***



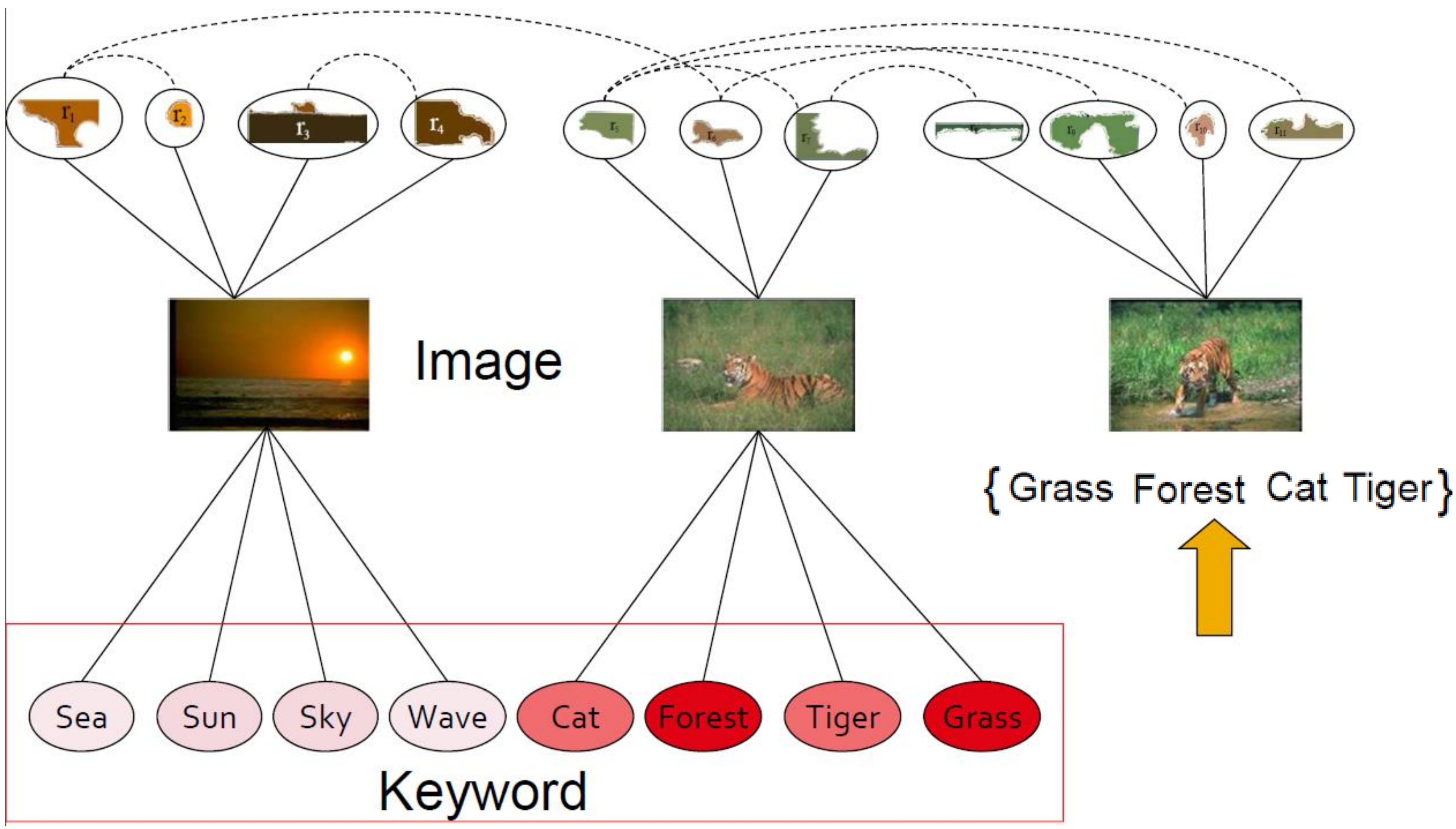
Random Walk with Restarts

- ***Affinity:*** The importance of node B with respect to node A is the ***steady-state probability $u_A(B)$*** of random walk with restarts
- The ***random walk with restarts*** operates as follows: to ***compute the affinity of node B for node A***
  - Consider a random walker that starts from node A
  - The random walker ***chooses randomly among the available edges every time***, except that, before he makes a choice, ***with probability c, he goes back to node A (restart)***
  - Let $u_A(B)$ denote the ***steady-state probability*** that our random walker will find himself at node B
  - Then, ***$u_A(B)$ is the affinity of B with respect to A***

Pan et. al.v2

Automatic Image Captioning

estimate the steady-state probabilities $u_{i_3}(*)$ for all nodes of the graph
keep only the nodes that correspond to terms & report top few (say 5), as caption words for I_3



Applications

- An evaluation of SimRank and Personalized PageRank to build a recommender system for the Web of Data [link]
- SoK: The Evolution of Sybil Defense via Social Networks [link]
- Article recommendation system on a citation network using Personalized Pagerank and Neo4j [link]

Datasets

- Citation Dataset
  - https://www.aminer.org/citation
- Yelp Dataset
  - https://www.yelp.com/dataset/download