

CS6322: Information Retrieval

Sanda Harabagiu

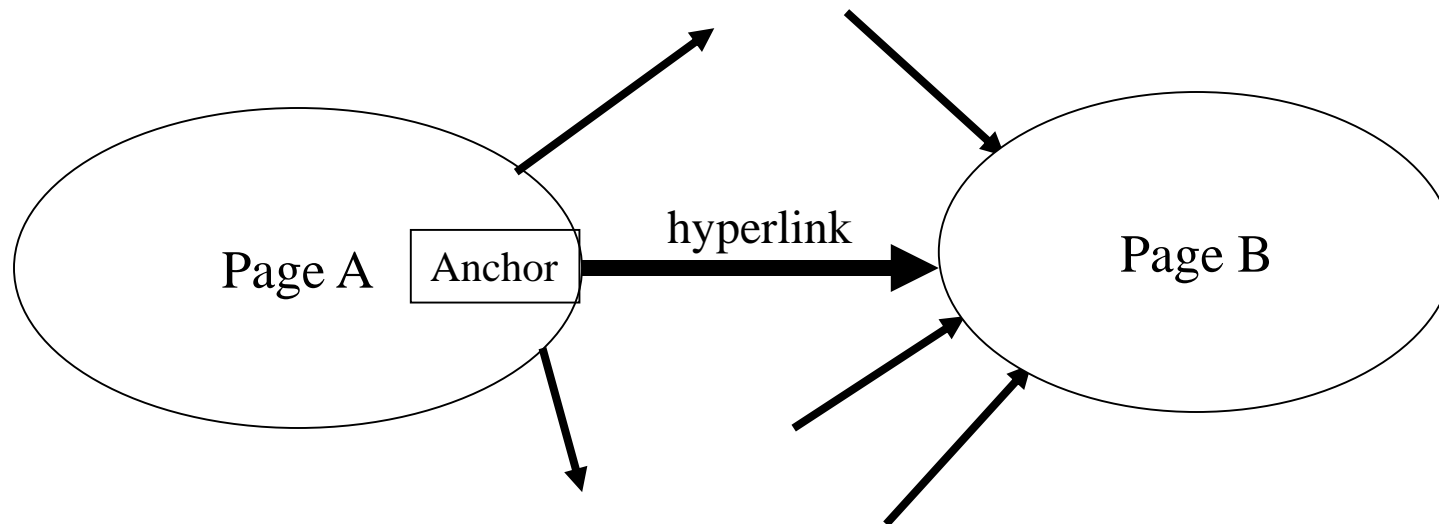
Lecture 10: Link analysis



Today's lecture

- Link analysis for ranking
 - Pagerank and variants
 - HITS

The Web as a Directed Graph



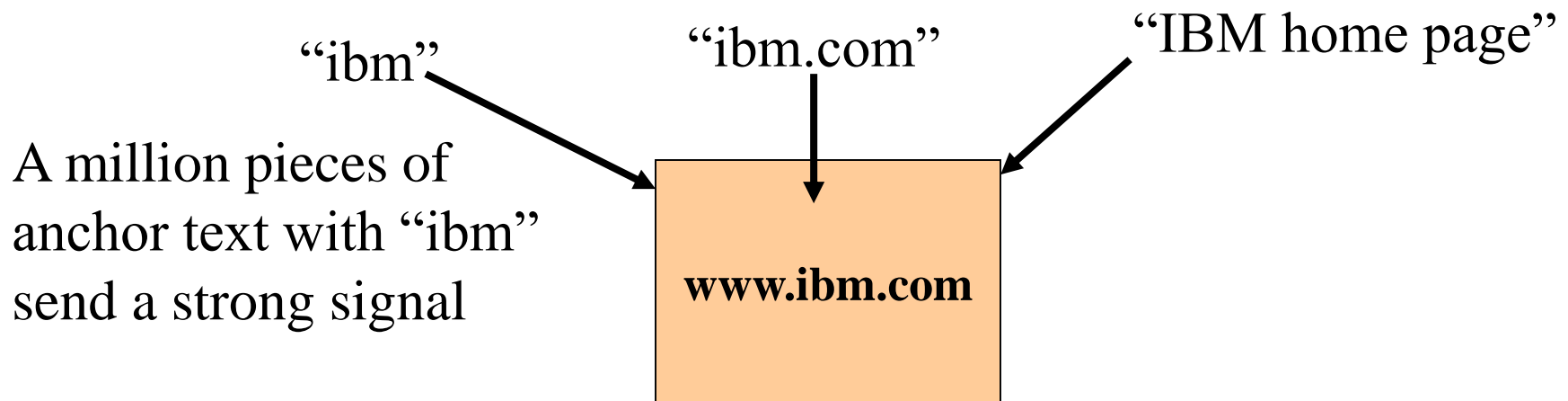
Assumption 1: A hyperlink between pages denotes author perceived relevance (quality signal)

Assumption 2: The text in the anchor of the hyperlink describes the target page (textual context)

Anchor Text

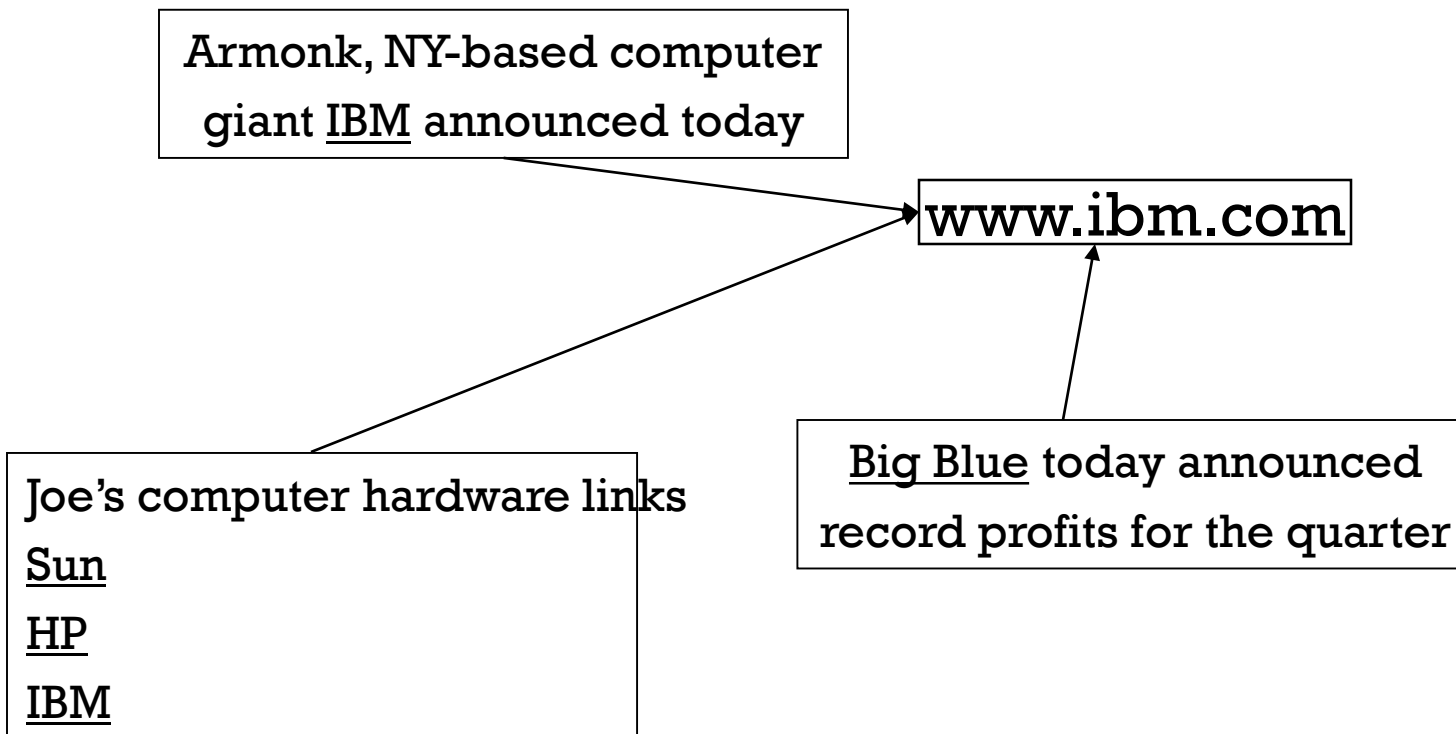
WWW Worm - McBryan [Mcbr94]

- For **ibm** how to distinguish between:
 - IBM's home page (mostly graphical)
 - IBM's copyright page (high term freq. for 'ibm')
 - Rival's spam page (arbitrarily high term freq.)



Indexing anchor text

- When indexing a document D , include anchor text from links pointing to D .



Indexing anchor text

- Can sometimes have unexpected side effects - *e.g., evil empire.*
- Can score anchor text with weight depending on the authority of the anchor page's website
 - E.g., if we were to assume that content from cnn.com or yahoo.com is authoritative, then trust the anchor text from them

Anchor Text

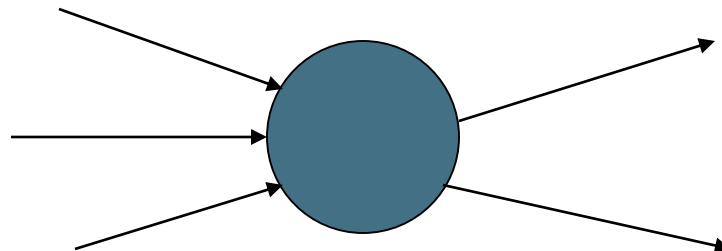
- Other applications
 - Weighting/filtering links in the graph
 - Generating page descriptions from anchor text

Citation Analysis

- Citation frequency
- Co-citation coupling frequency
 - Cocitations with a given author measures “impact”
 - Cocitation analysis
- Bibliographic coupling frequency
 - Articles that co-cite the same articles are related
- Citation indexing
 - Who is this author cited by? (Garfield 1972)
- Pagerank preview: Pinski and Narin '60s

Query-independent ordering

- First generation: using link counts as simple measures of popularity.
- Two basic suggestions:
 - Undirected popularity:
 - Each page gets a score = the number of in-links plus the number of out-links ($3+2=5$).
 - Directed popularity:
 - Score of a page = number of its in-links (3).



Query processing

- First retrieve all pages meeting the text query (say ***venture capital***).
- Order these by their link popularity (either variant on the previous slide).
- More nuanced – use link counts as a measure of static goodness, combined with text match score

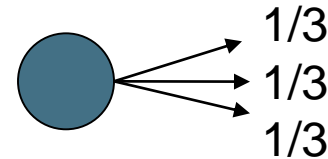
Spamming simple popularity

- *Exercise:* How do you spam each of the following heuristics so your page gets a high score?
- Each page gets a static score = the number of in-links plus the number of out-links.
- Static score of a page = number of its in-links.

Pagerank scoring

- Imagine a browser doing a random walk on web pages:

- Start at a random page

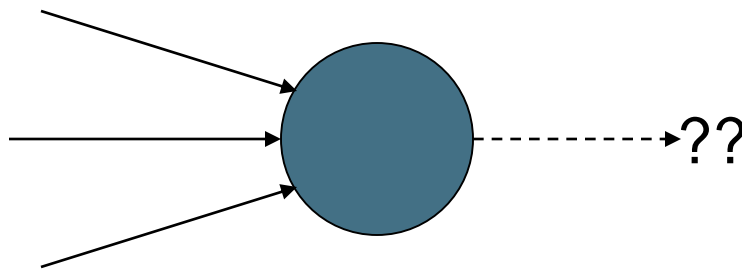


- At each step, go out of the current page along one of the links on that page, equiprobably

- “In the steady state” each page has a long-term visit rate - use this as the page’s score.

Not quite enough

- The web is full of dead-ends.
 - Random walk can get stuck in dead-ends.
 - Makes no sense to talk about long-term visit rates.



Teleporting

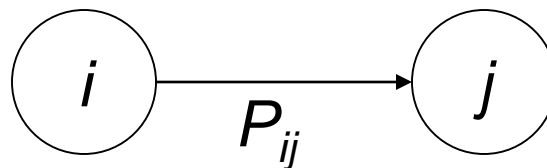
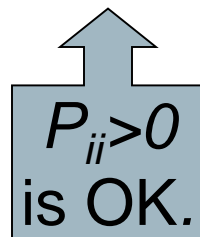
- At a dead end, jump to a random web page.
- At any non-dead end, with probability 10%, jump to a random web page.
 - With remaining probability (90%), go out on a random link.
 - 10% - a parameter.

Result of teleporting

- Now cannot get stuck locally.
- There is a long-term rate at which any page is visited (not obvious, will show this).
- How do we compute this visit rate?

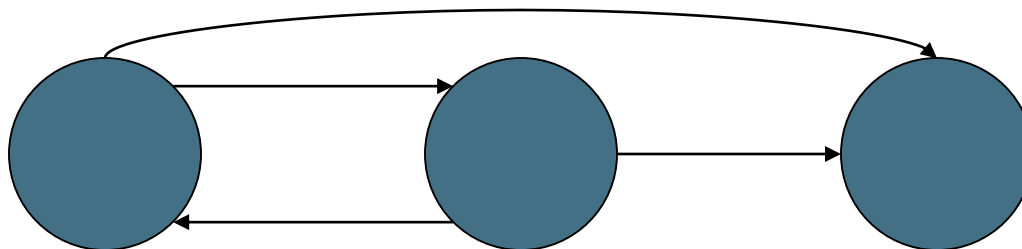
Markov chains

- A Markov chain consists of n states, plus an $n \times n$ transition probability matrix \mathbf{P} .
- At each step, we are in exactly one of the states.
- For $1 \leq i, j \leq n$, the matrix entry P_{ij} tells us the probability of j being the next state, given we are currently in state i .

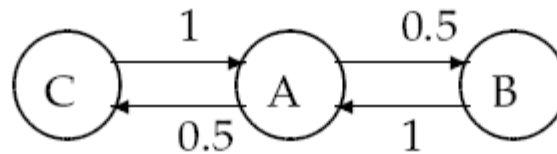


Markov chains

- Clearly, for all i , $\sum_{j=1}^n P_{ij} = 1$.
- **Markov chains are abstractions of random walks.**
- *Exercise:* represent the teleporting random walk from 3 slides ago as a Markov chain, for this case:



Example of Transition Probabilities



► **Figure 21.2** A simple Markov chain with three states; the numbers on the links indicate the transition probabilities.

The Transition
Probability Matrix

$$\begin{pmatrix} 0 & 0.5 & 0.5 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

A Random Walk

- Assumptions:

- We can view a random surfer on the web graph as a Markov chain, with:

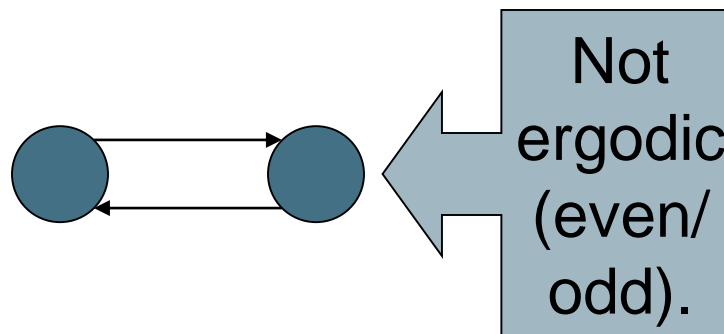
- one state for each web page, and each transition probability representing the probability of moving from one web page to another.
- The teleport operation contributes to these transition probabilities.
- The adjacency matrix A of the web graph is defined as follows:
 - if there is a hyperlink from page i to page j , then $A_{ij} = 1$, otherwise $A_{ij} = 0$.
- We can readily derive the transition probability matrix P for our Markov chain from the $N \times N$ matrix A ! How???

From Adjacency Matrix to Transition Probability Matrix:

- Derive the transition probability matrix P for our Markov chain from the $N \times N$ matrix A :
 1. If a row of A has no 1's, then replace each element by $1/N$. For all other rows proceed as follows.
 2. Divide each 1 in A by the number of 1's in its row. Thus, if there is a row with three 1's, then each of them is replaced by $1/3$.
 3. Multiply the resulting matrix by $1 - \alpha$.
 4. Add α/N to every entry of the resulting matrix, to obtain P .

Ergodic Markov chains

- A Markov chain is ergodic if
 - you have a path from any state to any other
 - For any start state, after a finite transient time T_0 , the probability of being in any state at a fixed time $T > T_0$ is nonzero.



Ergodic Markov chains

- For any ergodic Markov chain, there is a unique long-term visit rate for each state.
 - *Steady-state probability distribution.*
- Over a long time-period, we visit each state in proportion to this rate.
- It doesn't matter where we start.

Probability vectors

- A probability (row) vector $\mathbf{x} = (x_1, \dots, x_n)$ tells us where the walk is at any point.
- E.g., $(\underset{1}{000}\dots\underset{i}{1}\dots\underset{n}{000})$ means we're in state i .

More generally, the vector $\mathbf{x} = (x_1, \dots, x_n)$ means the walk is in state i with probability x_i .

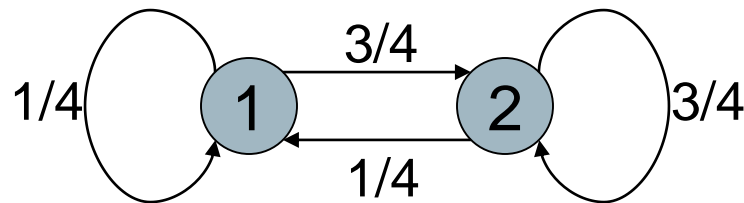
$$\sum_{i=1}^n x_i = 1.$$

Change in probability vector

- If the probability vector is $\mathbf{x} = (x_1, \dots, x_n)$ at this step, what is it at the next step?
- Recall that row i of the transition prob. Matrix \mathbf{P} tells us where we go next from state i .
- So from \mathbf{x} , our next state is distributed as \mathbf{xP} .

Steady state example

- The steady state looks like a vector of probabilities $\mathbf{a} = (a_1, \dots, a_n)$:
 - a_i is the probability that we are in state i .



For this example, $a_1=1/4$ and $a_2=3/4$.

State-State Theorem

STEADY-STATE

Theorem 21.1. *For any ergodic Markov chain, there is a unique steady-state probability vector $\vec{\pi}$ that is the principal left eigenvector of P , such that if $\eta(i, t)$ is the number of visits to state i in t steps, then*

$$\lim_{t \rightarrow \infty} \frac{\eta(i, t)}{t} = \pi(i),$$

where $\pi(i) > 0$ is the steady-state probability for state i .

How do we compute this vector?

- Let $\mathbf{a} = (a_1, \dots, a_n)$ denote the row vector of steady-state probabilities.
- If our current position is described by \mathbf{a} , then the next step is distributed as \mathbf{aP} .
- But \mathbf{a} is the steady state, so $\mathbf{a} = \mathbf{aP}$.
- Solving this matrix equation gives us \mathbf{a} .
 - So \mathbf{a} is the (left) eigenvector for \mathbf{P} .
 - (Corresponds to the “principal” eigenvector of \mathbf{P} with the largest eigenvalue.)
 - Transition probability matrices always have largest eigenvalue 1.

One way of computing \mathbf{a}

- Recall, regardless of where we start, we eventually reach the steady state \mathbf{a} .
- Start with any distribution (say $\mathbf{x}=(10\dots0)$).
- After one step, we're at \mathbf{xP} ;
- after two steps at \mathbf{xP}^2 , then \mathbf{xP}^3 and so on.
- “Eventually” means for “large” k , $\mathbf{xP}^k = \mathbf{a}$.
- Algorithm: multiply \mathbf{x} by increasing powers of \mathbf{P} until the product looks stable.

Connectivity-Based Ranking

Query-Independent Connectivity-Based Ranking

- A ranking score is assigned to a Web page without a specific user query → goal: **measure intrinsic quality of a page**
 - Why ?
 - At query time this score is used with or without some query-dependent criteria to rank all documents matching the query
- It is used by most search engines
- Introduced by Google

Assumptions

1. The larger the number of hyperlinks pointing to a page, the better the page !
2. Main drawback: each link is equally important.
 - It cannot distinguish between the quality of a page pointed to by a number of low-quality pages and the quality of a page pointed to by the same number of high-quality pages
 - ➔ it is easy to make a page appear to be high- quality by just creating many other pages that point to it.

Solution: The Page Rank Measure

- Weight each hyperlink *proportionally* to the quality of the page containing the hyperlink

Sergey Brin and Lawrence Page wrote a paper in 1998, entitled
“The Anatomy of a Large-Scale Hypertextual Web Search Engine”
<http://www-db.stanford.edu/~backrub/google.html>

The Google search engine has two important features that help it produce high precision results:

1. it makes use of the link structure of the Web to calculate a quality ranking for each web page. This is called the **PageRank** measure.
2. Google utilizes the link structure to improve search results.

The citation (link) graph of the web is an important resource that has largely gone unused before the Google web search engine. **PageRank was viewed as Bringing Order to the Web.**

Computing the Page Rank

- Academic citation literature has been applied to the web, largely by counting **citations** or **backlinks** to a given page. This gives some approximation of a page's importance or quality.
- PageRank extends this idea by not counting links from all pages equally, and by normalizing by the number of links on a page. PageRank is defined as follows:

*We assume page **A** has pages **T1...Tn** which point to it (i.e., are citations). The parameter **d** is a **damping factor** which can be set between 0 and 1. We usually set **d** to 0.85. Also **C(A)** is defined as the number of links going out of page **A**. The PageRank of a page **A** is given as follows:*

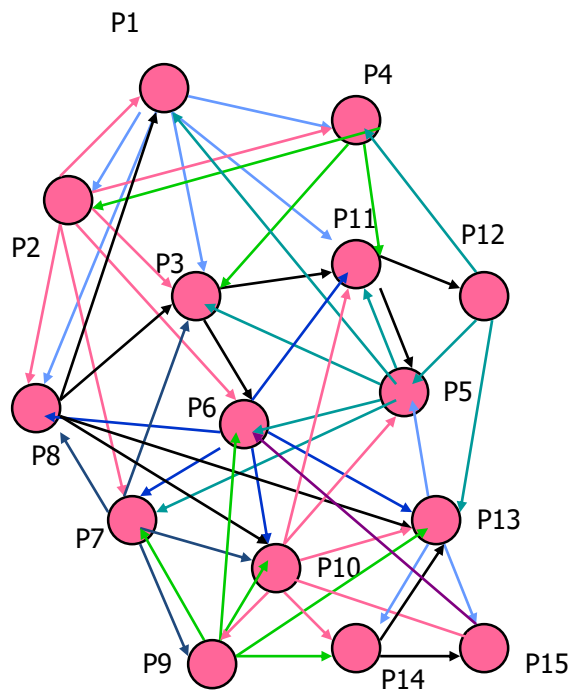
$$PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

*Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be **one**.*

Example of PageRanks

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

Web Graph with n=15 pages



Node	Source nodes	Outdegree	Target nodes
P1	P2,P5,P8	5	P2,P3,P4,P8, P11
P2	P1,P4	6	P1,P3,P4,P6,P7,P8
P3	P1,P2,P4,P5,P7,P8	2	P6,P11
P4	P1,P2,P12,P13	3	P2,P3,P11
P5	P10,P11,P12,P13	5	P1,P3,P6,P7,P11
P6	P2,P3,P5,P9,P15	4	P7,P8,P10,P13
P7	P2,P5,P6,P9	4	P3,P8,P9,P10
P8	P1,P2,P6,P7	4	P1,P3,P10,P13
P9	P7,P10	5	P6,P7,P10,P13,P14
P10	P6,P7,P8,P9	6	P5,P9,P11,P13,P14,P15
P11	P3,P4,P5,P10	2	P5,P12
P12	P11	3	P4,P5,P13
P13	P6,P8,P9,P10,P12,P14	3	P4,P5,P15
P14	P9,P10	2	P13,P15
P15	P10,P13,P14	1	P6

Computing the PageRanks

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

Node	Source nodes	C	PageRank (PR)
P1	P2,P5,P8	5	$PR(P1)=0.15+0.85[PR(P2)/6+PR(P5)/5+PR(P8)/4]$
P2	P1,P4	6	$PR(P2)=0.15+0.85[PR(P1)/5+PR(P4)/3]$
P3	P1,P2,P4,P5,P7,P8	2	$PR(P3)=0.15+0.85[PR(P1)/5+PR(P2)/6+PR(P4)/3+PR(P5)/5+PR(P7)/4+PR(P8)/4]$
P4	P1,P2,P12,P13	3	$PR(P4)=0.15+0.85[PR(P1)/5+PR(P2)/6+PR(P12)/3+PR(P13)/3]$
P5	P10,P11,P12,P13	5	$PR(P5)=0.15+0.85[PR(P10)/6+Pr(P11)/2+PR(P12)/3+PR(P13)/3]$
P6	P2,P3,P5,P9,P15	4	$PR(P6)=0.15+0.85[PR(P2)/6+Pr(P3)/2+PR(P5)/5+PR(P9)/5+PR(P15)]$
P7	P2,P5,P6,P9	4	$PR(P7)=0.15+0.85[PR(P2)/6+PR(P5)/5+PR(P6)/4+PR(P9)/5]$
P8	P1,P2,P6,P7	4	$PR(P8)=0.15+0.85[PR(P1)/5+PR(P2)/6+PR(P6)/4+PR(P7)/4]$
P9	P7,P10	5	$PR(P9)=0.15+0.85[PR(P7)/4+PR(P10)/6]$
P10	P6,P7,P8,P9	6	$PR(P10)=0.15+0.85[PR(P6)/4+PR(P7)/4+PR(P8)/4+PR(P9)/5]$
P11	P3,P4,P5,P10	2	$PR(P11)=0.15+0.85[PR(P3)/2+PR(P4)/3+PR(P5)/5+PR(P10)/6]$
P12	P11	3	$PR(P12)=0.15+0.85[PR(P11)/2]$
P13	P6,P8,P9,P10,P12,P14	3	$PR(P13)=0.15+0.85[PR(P6)/4+PR(P8)/4+PR(P9)/5+PR(P10)/6+PR(P12)/3+PR(P14)/2]$
P14	P9,P10	2	$PR(P14)=0.15+0.85[PR(P9)/5+PR(P10)/6]$
P15	P10,P13,P14	1	$PR(P15)=0.15+0.85[PR(P10)/6+PR(P13)/3+PR(P14)/2]$

To compute the PageRank of a web page, you need to know the page ranks of all the web pages pointing to it!

Explaining the Page Rank

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

PageRank or PR(A) can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.

but that's not too helpful so let's break it down into sections.

- **PR(T_n)** - Each page has a notion of its own self-importance. That's "PR(T₁)" for the first page in the web all the way up to "PR(T_n)" for the last page
- **C(T_n)** - Each page spreads its vote out evenly amongst all of it's outgoing links. The count, or number, of outgoing links for page 1 is "C(T₁)", "C(T_n)" for page n, and so on for all pages.
- **PR(T_n)/C(T_n)** - so if our page (page A) has a backlink from page "n" the share of the vote page A will get is "PR(T_n)/C(T_n)"
- **d(...** - All these fractions of votes are added together but, to stop the other pages having too much influence, this total vote is "damped down" by multiplying it by 0.85 (the factor "d")
- **(1 - d)** - The (1 - d) bit at the beginning is a bit of probability math magic so the "*sum of all web pages' PageRanks will be one*": it adds in the bit lost by the **d(...**. It also means that if a page has no links to it (no backlinks) even then it will still get a small PR of 0.15 (i.e. 1 - 0.85). (Aside: the Google paper says "the sum of all pages" but they mean the "the normalised sum" – otherwise known as "the average" to you and me.

Page Ranking

- **Basic idea:** if page u has a link to a page v , then the author of u is implicitly conferring some importance to page v .


Example: yahoo.com is an important page, since many other pages point to it. Likewise, pages prominently pointed to from yahoo.com are also important.

- How much importance does a page u confer to its outlinks?
 - Let: C_u -be the out degree of page u
 - Rank (u)-the importance (PageRank) of page u
 - The *link* (u,v) confers $\text{Rank}(u)/C_u$ units to page v .

Fixpoint Computation

- If N is the number of pages, assign all pages the initial value $1/N$.
- Let B_u the set of pages pointing to u .

For each iteration, the PageRanks are propagated as follows:


$$\forall u \quad Rank_{i+1}(u) = \sum_{v \in B_u} \frac{Rank_i(v)}{C_v}$$



Teleporting???

- How many iterations?
 - until Rank stabilizes within some threshold.
 - The final Rank* contains the PageRank vector over the Web. This vector is computed once after each crawl of the Web.

PageRanking at iteration 1

$$\text{Rank}_0(u) = 1/15$$

$$\forall u \text{ Rank}_{i+1}(u) = \sum_{v \in B_u} \frac{\text{Rank}_i(v)}{C_v}$$

Node	PR(i=1)	PageRank (PR)
P1	0.184595	$\text{PR}(P1) = 0.15 + 0.85 [\text{PR}(P2)/6 + \text{PR}(P5)/5 + \text{PR}(P8)/4 + \text{PR}(P11)/2]$
P2	0.17992	$\text{PR}(P2) = 0.15 + 0.85 [\text{PR}(P1)/5 + \text{PR}(P4)/3]$
P3	0.22854	$\text{PR}(P3) = 0.15 + 0.85 [\text{PR}(P1)/5 + \text{PR}(P2)/6 + \text{PR}(P4)/3 + \text{PR}(P5)/5 + \text{PR}(P7)/4 + \text{PR}(P8)/4]$
P4	0.20797	$\text{PR}(P4) = 0.15 + 0.85 [\text{PR}(P1)/5 + \text{PR}(P2)/6 + \text{PR}(P12)/3 + \text{PR}(P13)/3]$
P5	0.2248	$\text{PR}(P5) = 0.15 + 0.85 [\text{PR}(P10)/6 + \text{PR}(P11)/2 + \text{PR}(P12)/3 + \text{PR}(P13)/3]$
P6	0.26594	$\text{PR}(P6) = 0.15 + 0.85 [\text{PR}(P2)/6 + \text{PR}(P3)/2 + \text{PR}(P5)/5 + \text{PR}(P9)/5 + \text{PR}(P15)]$
P7	0.195815	$\text{PR}(P7) = 0.15 + 0.85 [\text{PR}(P2)/6 + \text{PR}(P5)/5 + \text{PR}(P6)/4 + \text{PR}(P9)/5]$
P8	0.19862	$\text{PR}(P8) = 0.15 + 0.85 [\text{PR}(P1)/5 + \text{PR}(P2)/6 + \text{PR}(P6)/4 + \text{PR}(P7)/4]$
P9	0.173375	$\text{PR}(P9) = 0.15 + 0.85 [\text{PR}(P7)/4 + \text{PR}(P10)/6]$
P10	0.203295	$\text{PR}(P10) = 0.15 + 0.85 [\text{PR}(P6)/4 + \text{PR}(P7)/4 + \text{PR}(P8)/4 + \text{PR}(P9)/5]$
P11	0.21732	$\text{PR}(P11) = 0.15 + 0.85 [\text{PR}(P3)/2 + \text{PR}(P4)/3 + \text{PR}(P5)/5 + \text{PR}(P10)/6]$
P12	0.17805	$\text{PR}(P12) = 0.15 + 0.85 [\text{PR}(P11)/2]$
P13	0.24537	$\text{PR}(P13) = 0.15 + 0.85 [\text{PR}(P6)/4 + \text{PR}(P8)/4 + \text{PR}(P9)/5 + \text{PR}(P10)/6 + \text{PR}(P12)/3 + \text{PR}(P14)/2]$
P14	0.17057	$\text{PR}(P14) = 0.15 + 0.85 [\text{PR}(P9)/5 + \text{PR}(P10)/6]$
P15	0.2061	$\text{PR}(P15) = 0.15 + 0.85 [\text{PR}(P10)/6 + \text{PR}(P13)/3 + \text{PR}(P14)/2]$

PageRanking at iteration n

$$\forall u \text{ Rank}_{i+1}(u) = \sum_{v \in B_u} \frac{\text{Rank}_i(v)}{C_u}$$

Node	PR(i=1)	PR(i=2)	PR(i=3)	PR(i=4)	PR(i=5)	PR(i=6)	PR(i=7)	PR(i=8)	PR(i=9)
P1	0.184595	0.124602	0.12003	0.126352	0.127836	0.126169	0.12463	0.123834	0.123193
P2	0.17992	0.106242	0.094269	0.086881	0.083472	0.086139	0.085455	0.084379	0.083845
P3	0.22854	0.279798	0.258325	0.26286	0.261619	0.261184	0.257781	0.256023	0.254606
P4	0.20797	0.208046	0.188625	0.174605	0.181714	0.180663	0.178359	0.177236	0.176059
P5	0.2248	0.283683	0.31051	0.32014	0.323773	0.322586	0.319639	0.31741	0.314779
P6	0.26594	0.429992	0.431867	0.399791	0.378678	0.38148	0.38245	0.375413	0.371516
P7	0.195815	0.176107	0.198509	0.201245	0.195504	0.190785	0.19124	0.190272	0.187821
P8	0.19862	0.182344	0.194152	0.197311	0.190009	0.183025	0.182657	0.182591	0.180251
P9	0.173375	0.082836	0.077321	0.08524	0.087244	0.084982	0.082446	0.082113	0.081831
P10	0.203295	0.199769	0.213678	0.221596	0.216635	0.208497	0.205819	0.205576	0.203491
P11	0.21732	0.262436	0.299279	0.289753	0.290592	0.292241	0.290079	0.286574	0.284835
P12	0.17805	0.10866	0.131218	0.149639	0.144876	0.145296	0.146121	0.14504	0.143287
P13	0.24537	0.329333	0.273445	0.276252	0.278675	0.271009	0.268081	0.266649	0.263929
P14	0.17057	0.068558	0.049862	0.051077	0.053981	0.053555	0.051746	0.050792	0.050685
P15	0.2061	0.200958	0.177351	0.151692	0.154555	0.155988	0.151863	0.149537	0.148542

Page Ranking as Eigenvector Calculation

- Let M represent the square, stochastic matrix corresponding to the directed graph G of the Web. If there is a link from page j to page i , then let the matrix entry M_{ij} have the value $1/C_j$ and let all the other entries have the value 0.
- One iteration of the previous fixpoint computation corresponds to the matrix multiplication $M \times \text{Rank}_i$.



$$\forall u \text{ Rank}_{i+1}(u) = \sum_{v \in B_u} \frac{\text{Rank}_i(v)}{C_v}$$

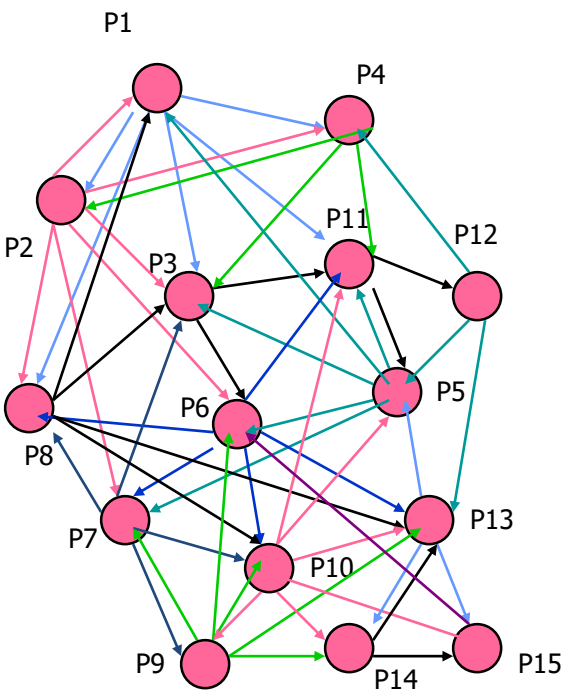
Repeatedly multiplying with M yields the dominant eigenvector of the matrix M .

$$\vec{R}_{i+1} = M \times \vec{R}_i$$

- Because M is a stochastic transition matrix over graph G , PageRank can be viewed as a stationary probability distribution over pages induced by a *random walk* on the Web.

Eigenvector Calculation

Web Graph with n=15 pages



Node	Source nodes	C
P1	P2,P5,P8	5
P2	P1,P4	6
P3	P1,P2,P4,P5,P7,P8	2
P4	P1,P2,P12,P13	3
P5	P10,P11,P12,P13	5
P6	P2,P3,P5,P9,P15	4
P7	P2,P5,P6,P9	4
P8	P1,P2,P6,P7	4
P9	P7,P10	5
P10	P6,P7,P8,P9	6
P11	P3,P4,P5,P10	2
P12	P11	3
P13	P6,P8,P9,P10,P12,P14	3
P14	P9,P10	2
P15	P10,P13,P14	1

→

$m_{12}=1/6; m_{15}=1/5; m_{18}=1/4$

→

$m_{31}=1/5; m_{32}=1/6; m_{34}=1/3;$
 $m_{35}=1/5; m_{37}=1/4; m_{38}=1/4$

→

$m_{62}=1/6; m_{63}=1/2; m_{65}=1/5;$
 $m_{69}=1/5; m_{6,15}=1$

Computing the transition matrix

Node	Source nodes	C	M
P1	P2,P5,P8	5	$M(1,2)=1/6; M(1,5)=1/5; M(1,8)=1/4$
P2	P1,P4	6	$M(2,1)=1/5; M(2,4)=1/3$
P3	P1,P2,P4,P5,P7,P8	2	$M(3,1)=1/5; M(3,2)=1/6; M(3,4)=1/3; M(3,5)=1/5; M(3,7)=1/4; M(3,8)=1/4$
P4	P1,P2,P12,P13	3	$M(4,1)=1/5; M(4,2)=1/6; M(4,12)=1/3; M(4,13)=1/3$
P5	P10,P11,P12,P13	5	$M(5,10)=1/6; M(5,11)=1/2; M(5,12)=1/3; M(5,13)=1/3$
P6	P2,P3,P5,P9,P15	4	$M(6,2)=1/6; M(6,3)=1/2; M(6,5)=1/5; M(6,9)=1/5; M(6,15)=1$
P7	P2,P5,P6,P9	4	$M(7,2)=1/6; M(7,5)=1/5; M(7,6)=1/4; M(7,9)=1/5$
P8	P1,P2,P6,P7	4	$M(8,1)=1/5; M(8,2)=1/6; M(8,6)=1/4; M(8,7)=1/4$
P9	P7,P10	5	$M(9,7)=1/4; M(9,10)=1/6$
P10	P6,P7,P8,P9	6	$M(10,6)=1/4; M(10,7)=1/4; M(10,8)=1/4; M(10,9)=1/5$
P11	P3,P4,P5,P10	2	$M(11,3)=1/2; M(11,4)=1/3; M(11,5)=1/5; M(11,10)=1/6$
P12	P11	3	$M(12,11)=1/2$
P13	P6,P8,P9,P10,P12,P14	3	$M(13,6)=1/4; M(13,8)=1/4; M(13,9)=1/5; M(13,P10)=1/6; M(13,12)=1/3; M(13,14)=1/2$
P14	P9,P10	2	$M(14,9)=1/5; M(14,10)=1/6$
P15	P10,P13,P14	1	$M(15,10)=1/6; M(15,13)=1/3; M(15,14)=1/2$

The transition matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1/6	0	0	1/5	0	0	1/4	0	0	0	0	0	0	0
2	1/5	0	0	1/3	0	0	0	0	0	0	0	0	0	0	0
3	1/5	1/6	0	1/3	1/5	0	1/4	1/4	0	0	0	0	0	0	0
4	1/5	1/6	0	0	0	0	0	0	0	0	0	1/3	1/3	0	0
5	0	0	0	0	0	0	0	0	0	1/6	1/2	1/3	1/3	0	0
6	0	1/6	1/2	0	1/5	0	0	0	1/5	0	0	0	0	0	1
7	0	1/6	0	0	1/5	1/4	0	0	1/5	0	0	0	0	0	0
8	1/5	1/6	0	0	0	1/4	1/4	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1/4	0	0	1/6	0	0	0	0	0
10	0	0	0	0	0	1/4	1/4	1/4	1/5	0	0	0	0	0	0
11	1/5	0	1/2	1/3	1/5	0	0	0	0	1/6	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1/2	0	0	0	0
13	0	0	0	0	0	1/4	0	1/4	1/5	1/6	0	1/3	0	1/2	0
14	0	0	0	0	0	0	0	0	1/5	1/6	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	1/6	0	0	1/3	1/2	0

A stochastic matrix (entries in each column sum to 1)

The transition matrix and the computation of PageRanks

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1/6	0	0	1/5	0	0	1/4	0	0	0	0	0	0	0
2	1/5	0	0	1/3	0	0	0	0	0	0	0	0	0	0	0



Node	PageRank (PR)
P1	$PR(P1) = 0.15 + 0.85 [PR(P2)/6 + PR(P5)/5 + PR(P8)/4]$
P2	$PR(P2) = 0.15 + 0.85 [PR(P1)/5 + PR(P4)/3]$

PageRank Convergence

How do we know that there is an *nr* = number of iterations for performing such that the eigenvector will stabilize?



$$\vec{R}_{i+1} = M \times \vec{R}_i$$


It is guaranteed only if graph G (the Web Structure) is strongly connected, thus **M is irreducible and aperiodic**.

1. Practice shows that the Web is aperiodic
2. To guarantee that M is irreducible, a damping factor (1- α) needs to be added to the rank propagation.

PageRank Convergence

- Practice from the Web shows that M is aperiodic
- M is irreducible if we add a damping factor $(1-\alpha)$ to the rank propagation.

We can define a new matrix M' in which we add transition edges of probability α/N between every pair of nodes in G

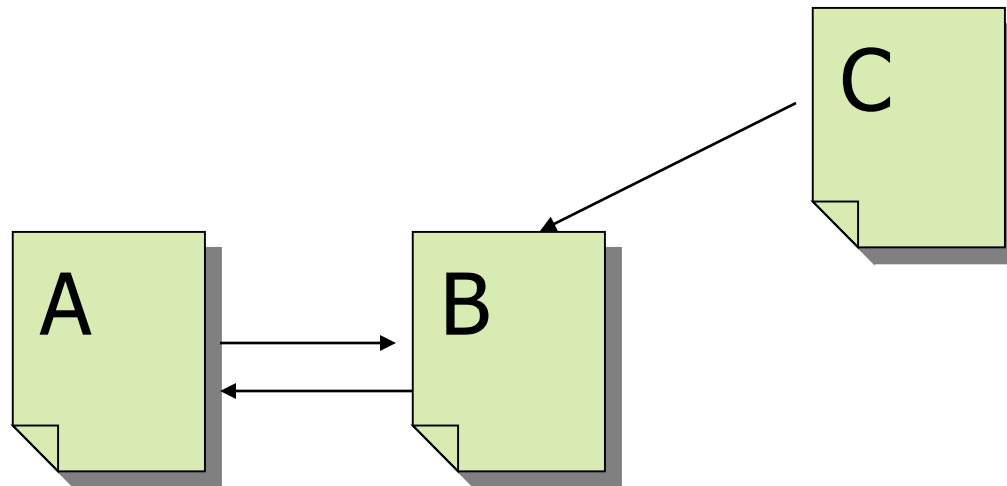


$$M' = (1 - \alpha)M + \alpha \underbrace{\left[\frac{1}{N} \right]_{N \times N}}$$

A matrix of size N , in which every element has the value $1/N$

Problems

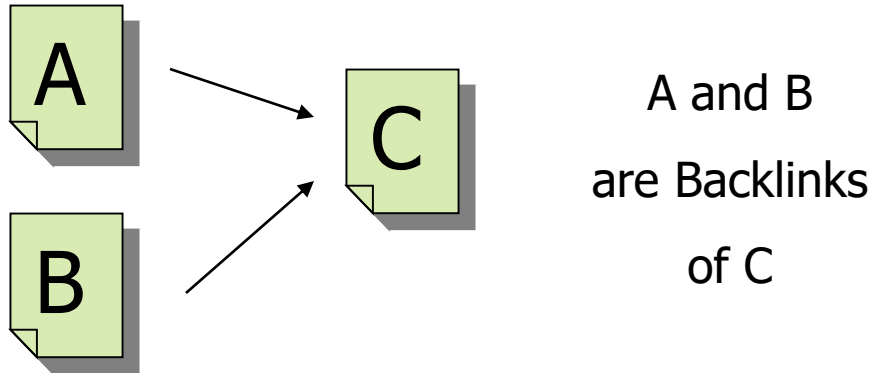
Consider 2 pages that point to each other but to no other page. Also there is some web page that points to one of them:



During iterations the **A-B loop** will accumulate rank, but never distribute any rank! (there are no out-edges)

This loop forms a kind of trap => rank sink

Backlinks



- Is C an important page?
- Netscape home has 62,804 backlinks.
- Citation computing is used for many things (including speculating on the future winner of the Nobel Prize)
- PageRank provides a more sophisticated method for doing citation counting.

PageRank is interesting

- There are many cases where simple citation counting does not correspond to our common-sense notion of importance.
- PageRank attempts to see how good an approximation of “importance” can be obtained from the **link structure**.

One possibility:

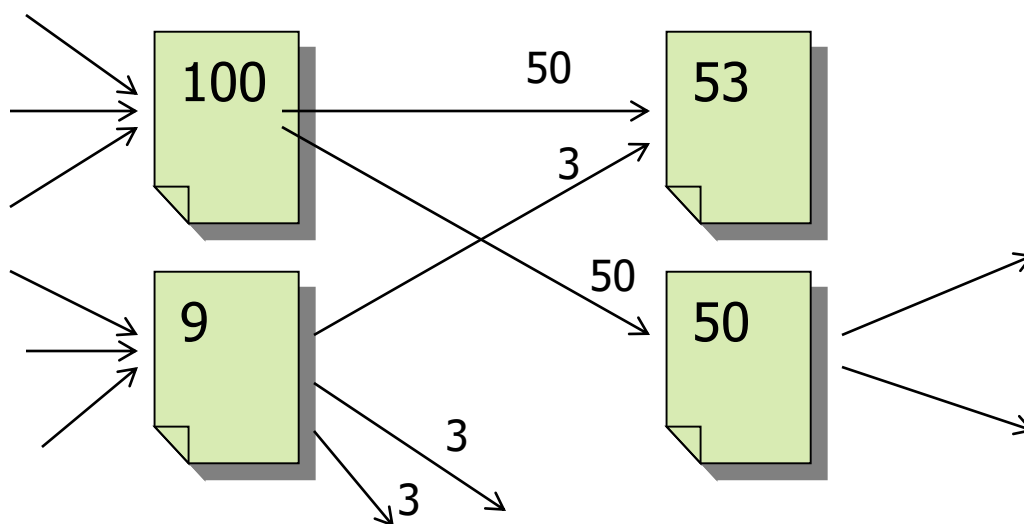
Simple Ranking:

-- **For $c < 1$**

$$R(u) = \frac{1}{c} \sum_{v \in \text{IngoingLinks}(u)} \frac{R(v)}{\# \text{OutgoingLinks}(v)}$$

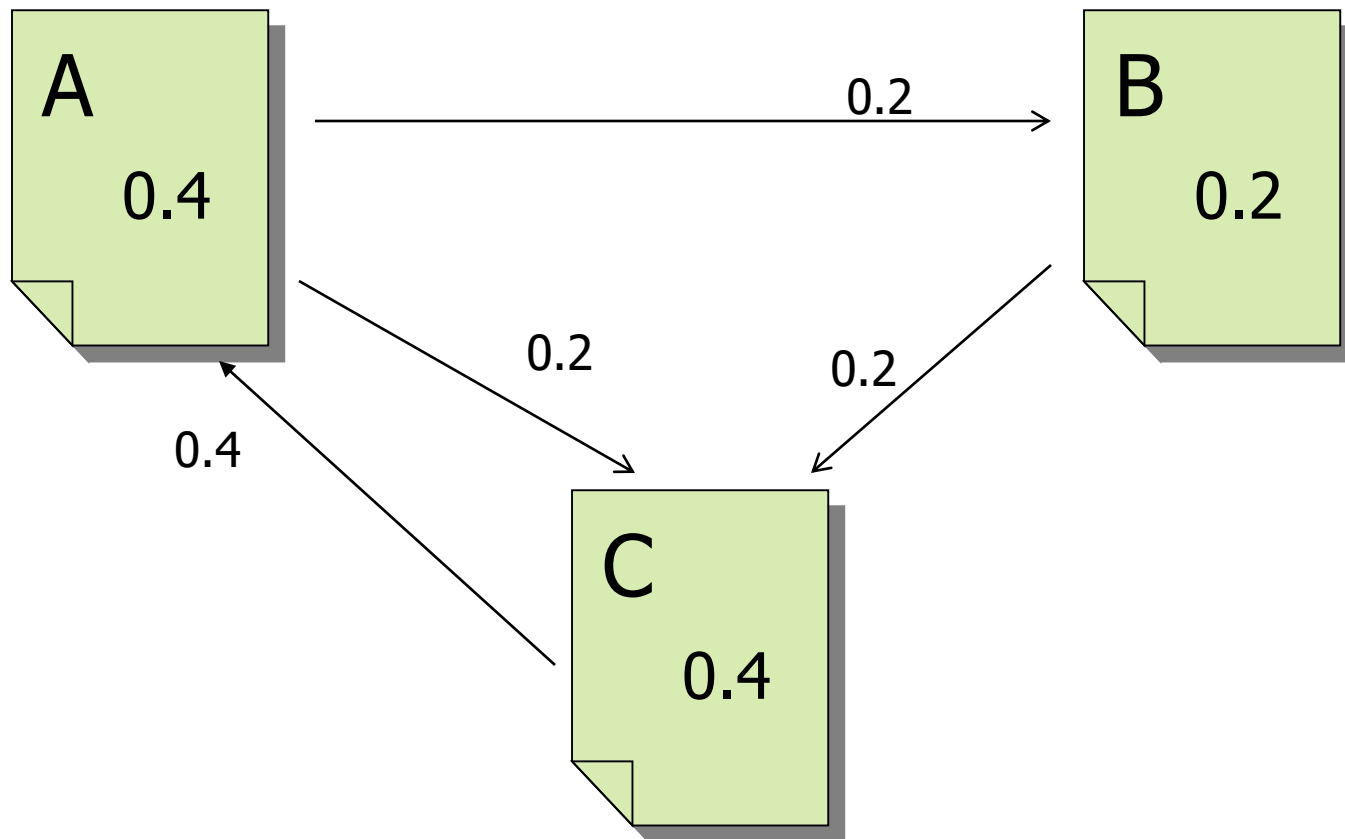
Example

Simplified Page Ranking Calculation



- Note: the Rank of a page is divided evenly among its forwarding links to contribute to the ranks of the pages they point to.
- $c < 1$ because there are pages with no forwarding links, their weight is lost!

A Consistent Steady State for PageRank Calculation



Formalization

- Let M be a square matrix with rows and columns corresponding to web pages.
- Let $M_{u,v} = 1/C_u$ if there is an edge between u and v
and $C_u = \#$ of outgoing links from u

- Let R be a vector over web pages

$$\Rightarrow C \times V = M \times R$$



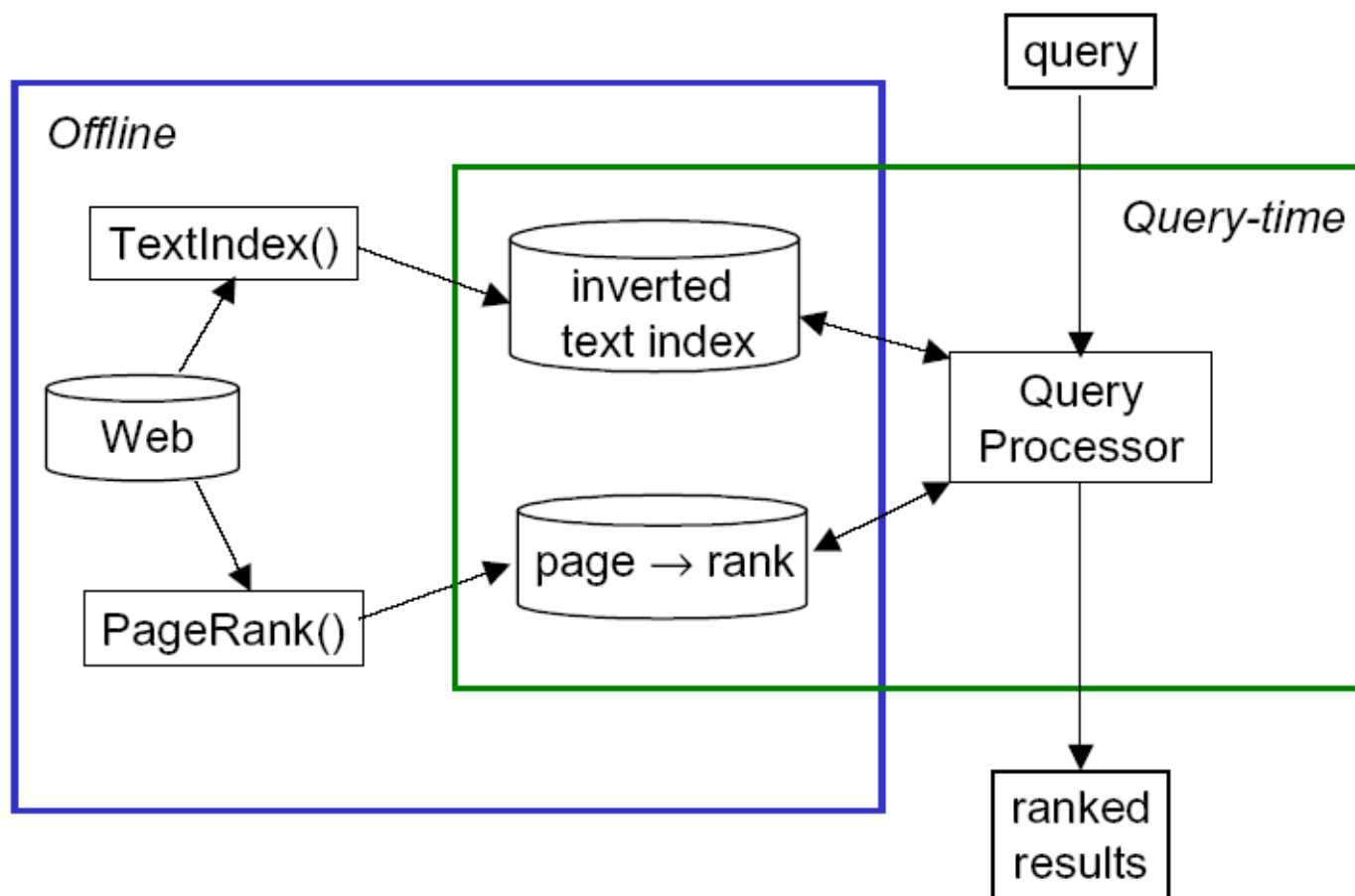
V is an eigenvector of M with eigenvalue C

- What do we want?
- The dominant eigenvector of M *Can be computed by repeatedly applying M to any nondegenerate initial rank vector.*
- Since M is stochastic, the dominant eigenvalue is 1, leading to the equality $M \times \text{Rank}^* = \text{Rank}^*$.

Interpretation

- 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.
 - And, the d damping factor is the probability at each page the "random surfer" will get bored and request another random page.
 - One important variation is to only add the damping factor d to a single page, or a group of pages. This allows for personalization and can make it nearly impossible to deliberately mislead the system in order to get a higher ranking.

Simple Search Engine Utilizing the Standard PageRank Scheme



Pagerank summary

- **Preprocessing:**
 - Given graph of links, build stochastic matrix \mathbf{P} .
 - From it compute \mathbf{a} .
 - The entry a_i is a number between 0 and 1: the pagerank of page i .
- **Query processing:**
 - Retrieve pages meeting query.
 - Rank them by their pagerank.
 - Order is *query-independent*.

The reality

- Pagerank is used in google, but is hardly the full story of ranking
 - Many sophisticated features are used
 - Some address specific query classes
 - Machine learned ranking heavily used
- Pagerank still very useful for things like crawl policy

Pagerank: Issues and Variants

- *How realistic is the random surfer model?*
 - (Does it matter?)
 - What if we modeled the back button?
 - Surfer behavior sharply skewed towards short paths
 - Search engines, bookmarks & directories make jumps non-random.
- *Biased Surfer Models*
 - Weight edge traversal probabilities based on match with topic/query (non-uniform edge selection)
 - Bias jumps to pages on topic (e.g., based on personal bookmarks & categories of interest)

Topic Specific Pagerank

- Goal – pagerank values that depend on query *topic*
- Conceptually, we use a random surfer who teleports, with say 10% probability, using the following rule:
 - Selects a topic (say, one of the 16 top level ODP categories) based on a query & user -specific distribution over the categories
 - Teleport to a page uniformly at random within the chosen topic
- Sounds hard to implement: can't compute PageRank at query time!

Topic Specific Pagerank

- **Offline:** Compute pagerank for *individual* topics
 - Query independent as before
 - Each page has multiple pagerank scores – one for each ODP category, with teleportation only to that category
- **Online:** Query context classified into (distribution of weights over) topics
 - Generate a dynamic pagerank score for each page – weighted sum of topic-specific pageranks

Topic-Sensitive Page Ranking

- Idea first published in 2002 in the WWW Conference-by Taher Haveliwala (Stanford University)
- The Idea:
 - In the original Page Ranking algorithm, the link structure of the web is used to capture the “importance” of Web pages by computing a simple vector $\text{Rank}(\text{page})$
 - Topic-sensitive Page Ranking computes a set of PageRank vectors , biased using a set of representative topics
 - As in ordinary PageRanking, the importance scores are computed offline
 - But instead of computing a single score (PageRank), we compute multiple scores, each corresponding to a given topic
 - This score can be used in conjunction with other IR-based scoring schemes to produce a final rank for the result pages with respect to a query.

The Key to Topic-Sensitive PageRank

Bias the computation to increase the effect of certain categories of pages.

- Generate a set of biased PageRank vectors using a set of “basic” topics

How? First use a non-uniform $N \times 1$ personalization vector:

$$\vec{p} = \left[\frac{1}{N} \right]_{N \times 1}$$

- Why? We have considered a decay factor for computing PageRanks.

$$M' = (1 - \alpha)M + \alpha \left[\frac{1}{N} \right]_{N \times N}$$

$$\vec{R}_{i+1} = M \times \vec{R}_i$$

The Substitution

Use a **non**-uniform $N \times 1$ personalization vector:

$$\vec{p} = \left[\frac{1}{N} \right]_{N \times 1}$$

- Why? We have considered a decay factor for computing PageRanks.

$$M' = (1 - \alpha)M + \alpha \left[\frac{1}{N} \right]_{N \times N}$$

$$\vec{R}_{i+1} = M \times \vec{R}_i$$

Substitute M' for M

$$Rank = (1 - \alpha)M + Rank + \alpha \vec{p}$$

By using a non-uniform personalization vector, we can bias the calculation to increase the effect of certain categories of pages!

Biasing

What is biasing?

Introduce an additional rank to the appropriate nodes in each iteration of computation

Initially uniform personalization

$$\vec{Rank} = (1 - \alpha) M \times \vec{Rank} + \alpha \left[\frac{1}{N} \right] \times \underset{\substack{\downarrow \\ \text{appropriate} \\ \text{vector}}}{A}$$

$$A = [\underbrace{0.1 \quad 0.5 \quad 1 \quad 1 \quad 0.2}_{\text{important}} \quad \dots \quad 1 \quad 0.1 \quad 0.4 \quad 0.6 \quad 0.7 \quad 0.1 \quad \dots \quad 0.3]$$

Some web pages are more important !!!

Personalization in the Random-Walk Model

- The personalization vector entails the addition of a complete set of transition edges with the probability of an edge (u, v) given by $\alpha \times p_v$ with $0 \leq p_v \leq 1$.
 - By selecting a certain value in vector p we state the preference that we want to give to a page regardless of the value α .

Open Directory Project (ODP) <http://dmoz.org> <http://dmoztools.net/>

- 3.8 million sites
 - Not as large as Google
- 91,929 editors
- 1,031,722 categories
- Helps crawling.
- **The Open Directory Project** is the largest, most comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors.



The Republic of the Web

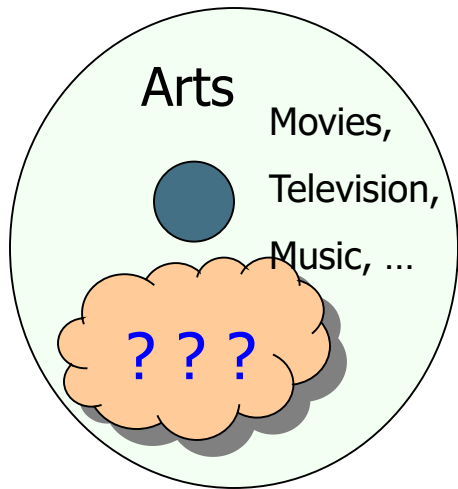
- The web continues to grow at staggering rates.
 - Automated search engines are increasingly unable to turn up useful results to search queries.
 - The small paid editorial staffs at commercial directory sites can't keep up with submissions, and the quality and comprehensiveness of their directories has suffered.

- Instead of fighting the explosive growth of the Internet, the Open Directory provides the means for the Internet to organize itself.
 - As the Internet grows, so do the number of net-citizens.
 - These citizens can each organize a small portion of the web and present it back to the rest of the population, culling out the bad and useless and keeping only the best content.

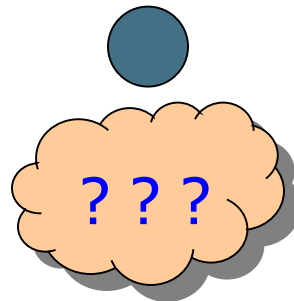
DMOZ- The name???

- The ODP is also known as DMOZ, an acronym for Directory Mozilla. This name reflects its loose association with Netscape's [Mozilla](#) project, an Open Source browser initiative. The ODP was developed in the spirit of Open Source, where development and maintenance are done by net-citizens, and results are made freely available for all net-citizens.

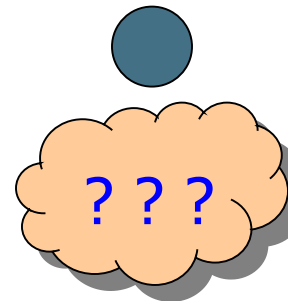
Crawler Starting Points



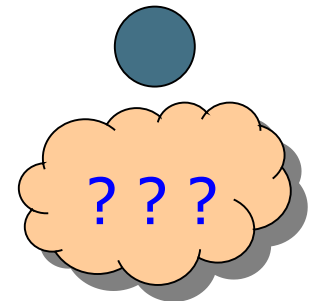
Business



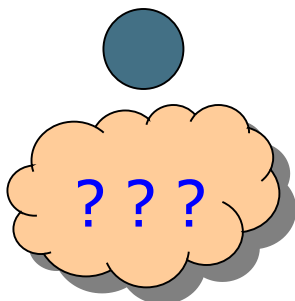
Computers



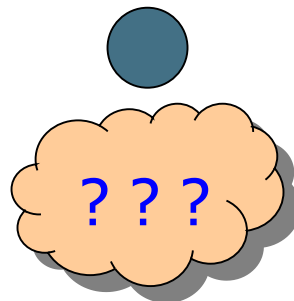
Games



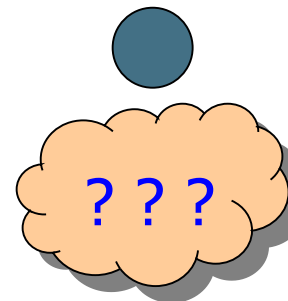
Health



Home

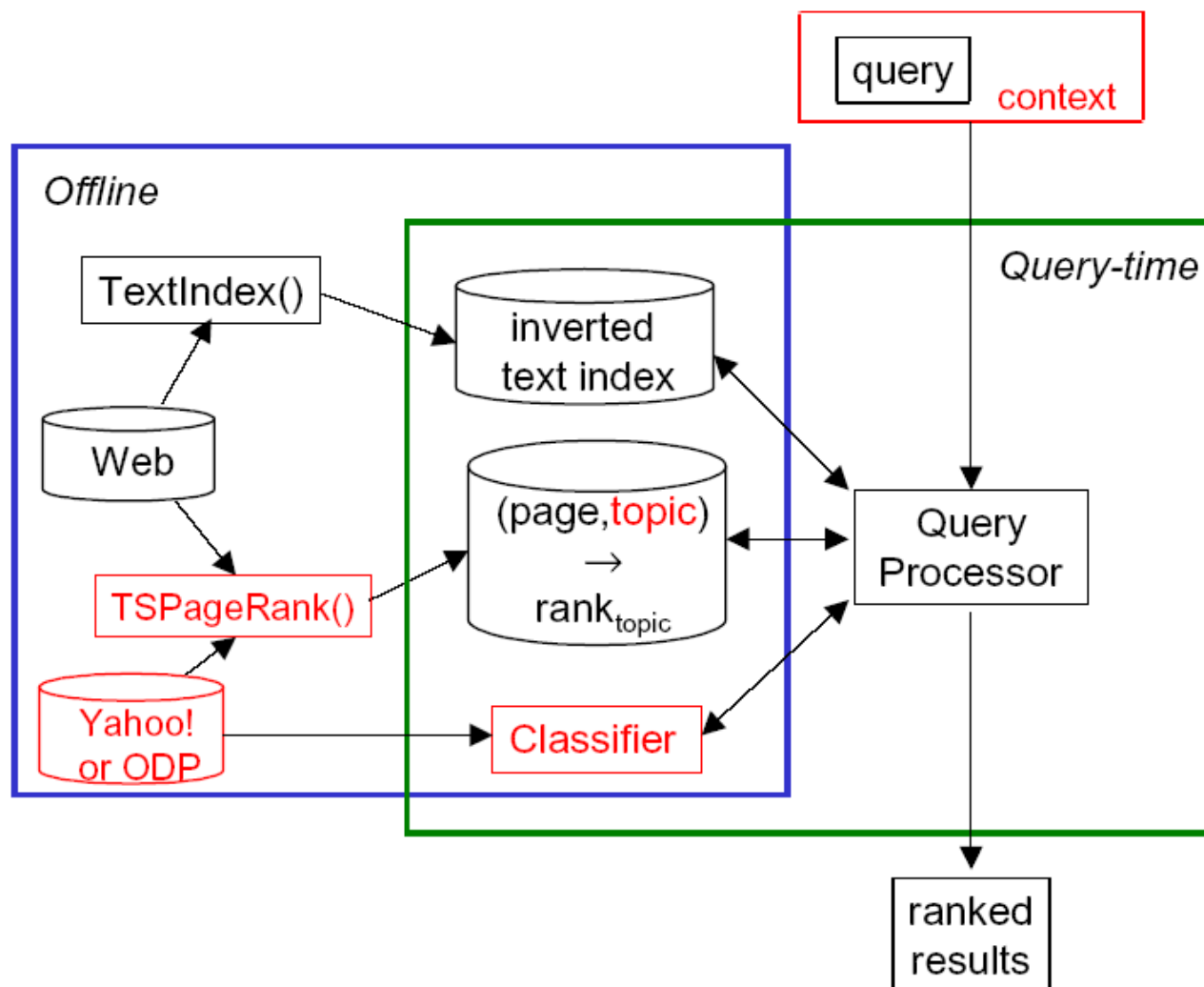


Kids & Teens



...

System Utilizing Topic-Sensitive PageRank



More on Topic-Biasing

To generate a set of biased PageRank vectors, we used a set of “basic” topics

- Where should we take these basic topics from?
- Answer: The Open Directory Project
 - A web directory for over 3.8 million URL's
 - Where <http://www.dmoztools.net>
 - The Open Directory Project is the largest, most comprehensive human-edited directory on the web.
 - Constructed and maintained by a vast, global community of volunteer editors.
 - Hosted and administered by Netscape Communications Corporation

ODP-based Biasing

- Step 1: Create 16 different biased Page Rank vectors by using the URLs below each of the top level categories of the ODPs as personalization vectors
 - Example: ODP categories C_1, C_2, \dots, C_{16}
 Let T_j be the set of URLs of the ODP category j
 When computing the PageRank vector for topic c_j , in place of the uniform damping vector $p=v_j$

where

$$\vec{p} = \left[\frac{1}{N} \right]_{N \times 1}$$

use

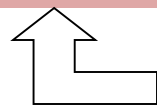
$$v_{ij} = \begin{cases} \frac{1}{|T_j|} & i \in T_j \\ 0 & i \notin T_j \end{cases}$$

Example

$C_2 = \text{Games}$

$T_2 =$ all hyperlinks from <http://dmoz.org/Games>
(52,481 of them)

$$v_2 = \begin{bmatrix} 0 & 0 & \dots & \frac{1}{52481} & 0 & 0 & \dots & \frac{1}{52481} & \dots \end{bmatrix}$$



This is one of the pages categorized under C_2

Similarly we generate all the other vectors v_1, v_2, \dots, v_{16}
Additionally, generate the simple, unbiased PageRank vector

Topic-Sensitive Page Ranging

-the Approach

- At query time, these importance scores are combined based on the **topics of the query** and generate a composite PageRank score for the pages

Watching the query  IR watchings.

- The composite PageRank score can be used in conjunction with other IR-based scoring schemes to produce the final rank for the result pages with respect to the query.

Where is the IR part?

- STEP 1: Compute 16 class *term vectors* D_j , $j=1,\dots,16$
 - Each term vector consists of the terms in the documents from each corresponding category
 - D_{jt} – occurrences of term t in documents listed under class c_j in ODP

Step 2: Query-Time Importance Score

- Let q be the query. What is the context of q ?
 - If a document d contains the terms of q , then all the other terms from web page d are part of the context of q
- How do we generate the context of a query?
 - Step 1: Retrieve web pages matching the query
 - Step 2: all terms from all retrieved pages are the context $\text{Con}(q)$
- Why do we want to use the context?
 - To measure the personalization vector \vec{p}

Computing Probabilities

- Unigram language model \rightarrow class probabilities for each of the 16 top level ODP classes
- Let $\text{Con}(q_i)$ be the context of the i -th term in a query
- Given query q , compute for each ODP category c_j

$$P(c_j | \text{Con}(q_i)) = \frac{P(c_j) \cdot P(\text{Con}(q) | c_j)}{P(\text{Con}(q))} \propto P(c_j) \prod_i P(\text{Con}(q_i) | c_j)$$

How do we compute $P(\text{Con}(q_i) | C_j)$?

$$\text{from } \vec{D}_j \Rightarrow \frac{|\text{Con}(q_i)|}{|\vec{D}_j|} = P(\text{Con}(q_i) | c_j)$$

Multinomial Naïve
Bayes Classifier

The term vector for Category j

Topic Probabilities

- Computing $P(c_j)$ is not easy!!!
 - For a user K we can use a prior distribution $P_k(c_j)$ that reflects the interests of user K for category c_j (how many web pages from category c_j (s)he has visited in the past month/total #visited pages)
- Alternative framework for user-based personalization rather than directly varying the damping personalization vector \vec{p}

Topic Specific Pagerank [Have02]

- Conceptually, we use a random surfer who teleports, with say 10% probability, using the following rule:
 - Selects a category (say, one of the 16 top level ODP categories) based on a query & user -specific distribution over the categories
 - Teleport to a page uniformly at random within the chosen category
- Sounds hard to implement: can't compute PageRank at query time!

Topic Specific Pagerank [Have02]

- Implementation

- **offline**: Compute pagerank distributions wrt to *individual* categories

Query independent model as before

Each page has multiple pagerank scores – one for each ODP category, with teleportation only to that category

- **online**: Distribution of weights over categories computed by query context classification

Generate a dynamic pagerank score for each page – as a weighted sum of category-specific pageranks

Influencing PageRank (“Personalization”)

- Input:
 - Web graph W
 - influence vector \mathbf{v}
 $\mathbf{v} : (\text{page} \rightarrow \text{degree of influence})$
- Output:
 - Rank vector \mathbf{r} : (page \rightarrow page importance wrt \mathbf{v})
- $\mathbf{r} = \text{PR}(W, \mathbf{v})$

Influencing PageRank (“Personalization”)

- Input:

- Web graph W
- Influence vector \mathbf{v} over topics
 $\mathbf{v} : (\text{page} \rightarrow \text{degree of influence})$

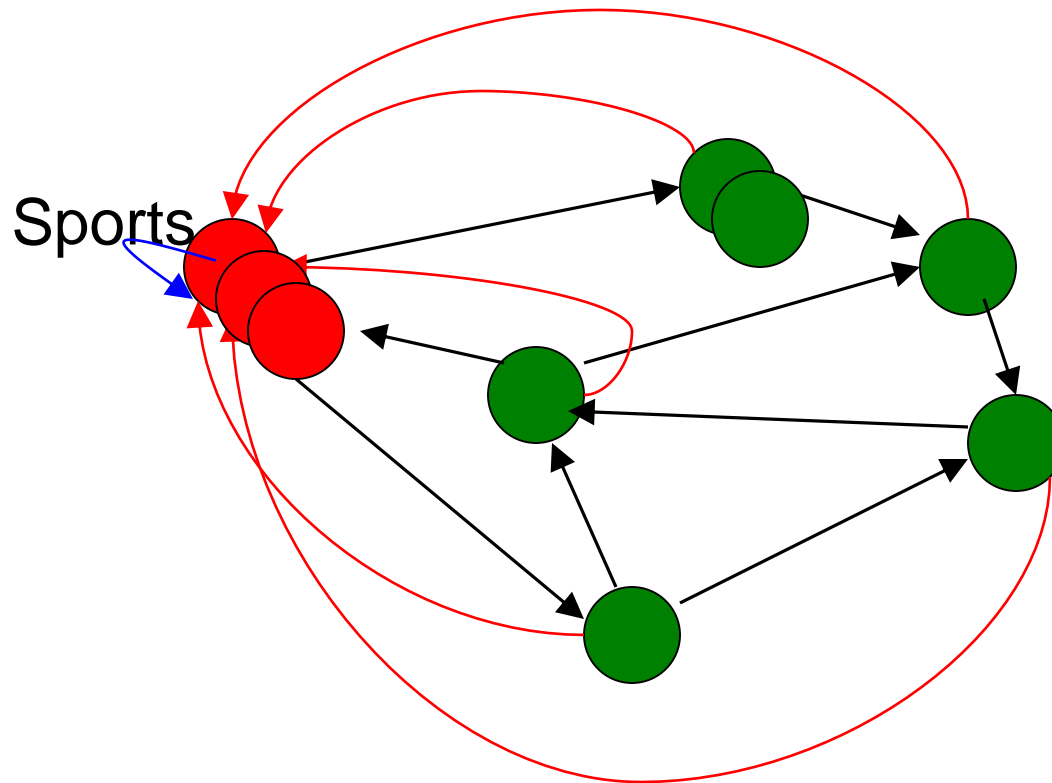
Vector has one
component for
each topic

- Output:

- Rank vector \mathbf{r} : (page \rightarrow page importance wrt \mathbf{v})

- $\mathbf{r} = \text{PR}(W, \mathbf{v})$

Non-uniform Teleportation



Teleport with 10% probability to a Sports page

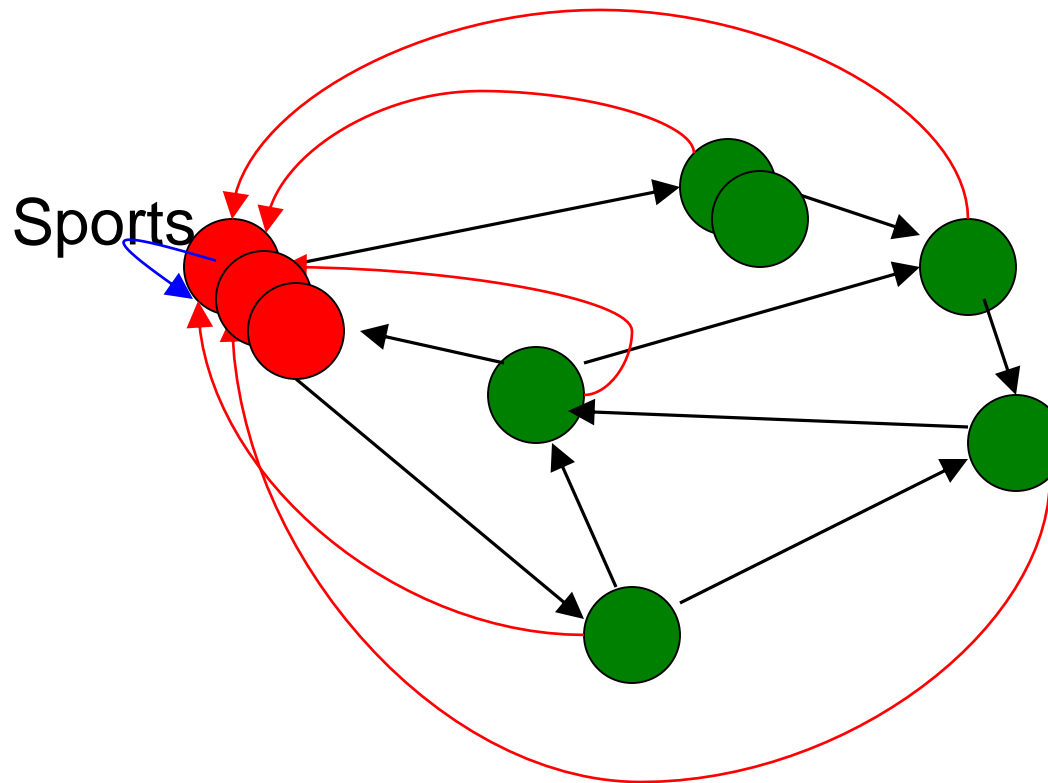
Interpretation of Composite Score

- Given a set of personalization vectors $\{\mathbf{v}_j\}$

$$\sum_j [w_j \cdot \text{PR}(W, \mathbf{v}_j)] = \text{PR}(W, \sum_j [w_j \cdot \mathbf{v}_j])$$

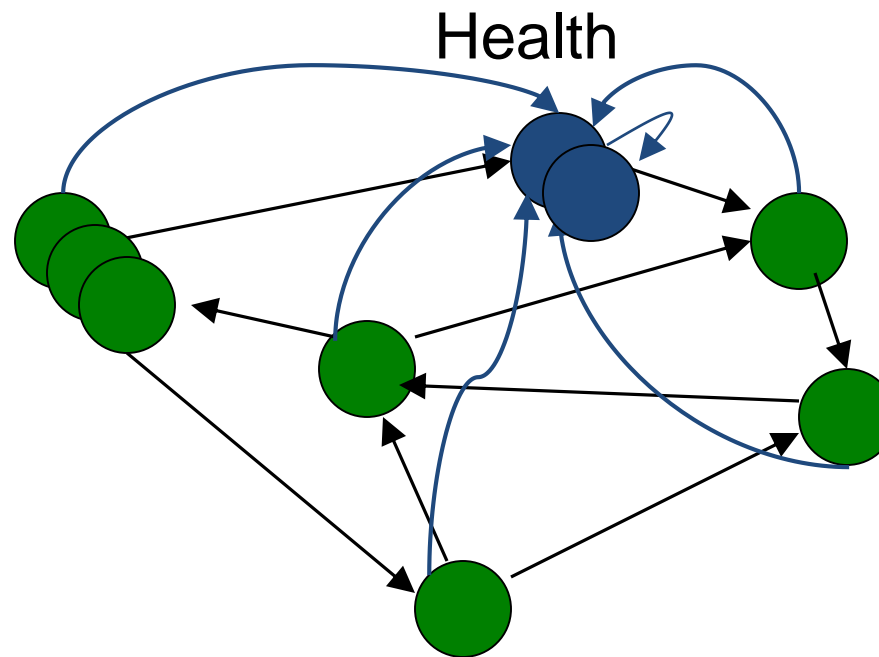
Given a user's preferences over topics, express as a combination of the “basis” vectors \mathbf{v}_j

Interpretation



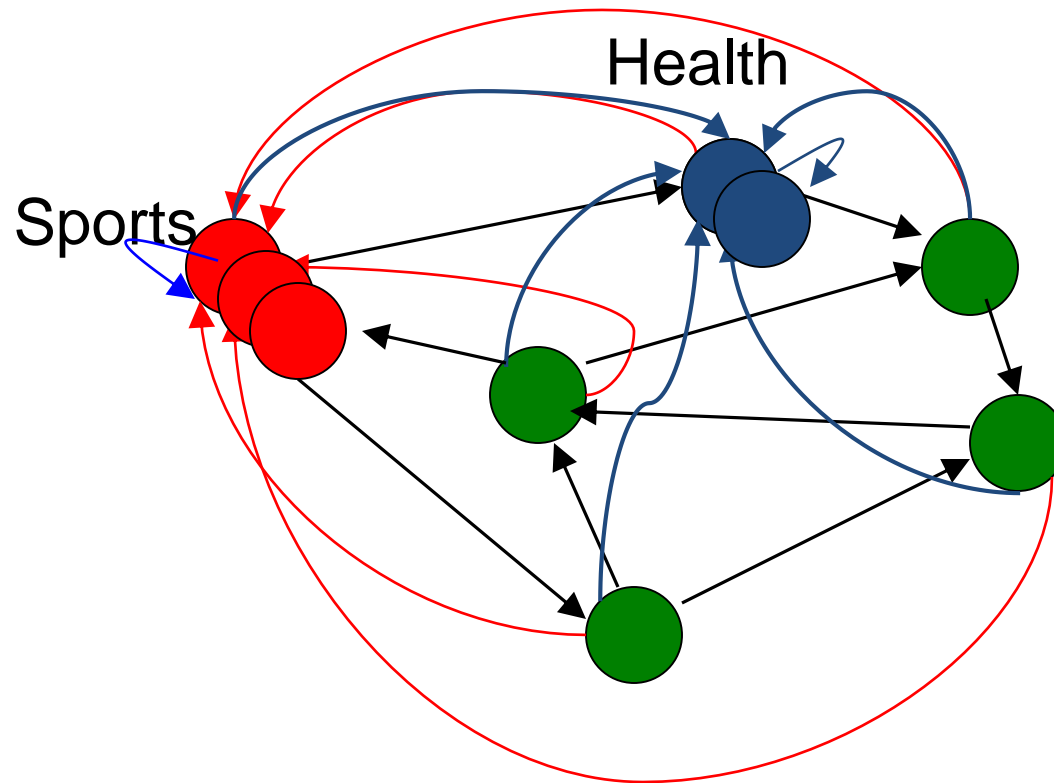
10% Sports teleportation

Interpretation



90% Health teleportation

Interpretation



$pr = (0.1 PR_{\text{sports}} + 0.9 PR_{\text{health}})$ gives you:
10% sports teleportation, 90% health teleportation

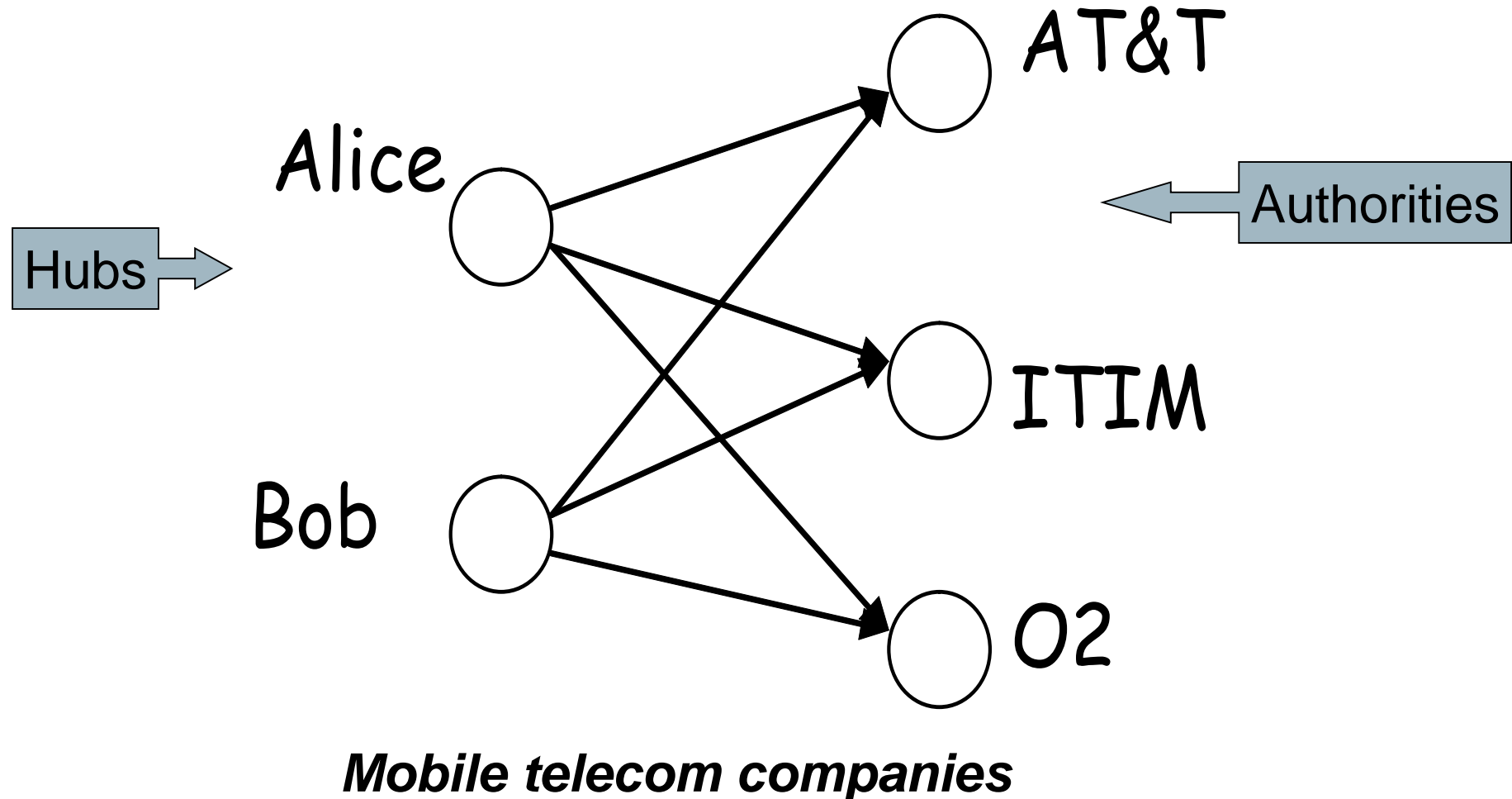
Hyperlink-Induced Topic Search (HITS)

- In response to a query, instead of an ordered list of pages each meeting the query, find two sets of inter-related pages:
 - *Hub pages* are good lists of links on a subject.
 - e.g., “Bob’s list of cancer-related links.”
 - *Authority pages* occur recurrently on good hubs for the subject.
- Best suited for “broad topic” queries rather than for page-finding queries.
- Gets at a broader slice of common *opinion*.

Hubs and Authorities

- Thus, a good hub page for a topic *points* to many authoritative pages for that topic.
- A good authority page for a topic is *pointed to* by many good hubs for that topic.
- Circular definition - will turn this into an iterative computation.

The hope



High-level scheme

- Extract from the web a base set of pages that *could* be good hubs or authorities.
- From these, identify a small set of top hub and authority pages;
 - iterative algorithm.

The HITS Algorithm

Authored by Kleinberg (Hypertext Induced Topic Search)

- Given a **user query**, the algorithm iteratively computes a **hub score** and an **authority score** for each node in the neighborhood graph
- Documents ranked by hub and authority scores

Notes:

- Nodes with high authority scores are expected to have relevant content
- Nodes with high hub scores are expected to contain hyperlinks to relevant content

Intuitions behind the HITS Algorithm

- A document that points to many others **might be a good hub**
- A document that many documents point to **might be a good authority**
- Recursively – a document that points to many good authorities might be an even better hub
- Similarly – a document pointed to by many good hubs might be an even better authority

Hyperlink-Induced Topic Search (HITS) – Kleinberg 98

- In response to a query, instead of an ordered list of pages each meeting the query, find two sets of inter-related pages:
 - *Hub pages* are good lists of links on a subject.
 - e.g., “Bob’s list of cancer-related links.”
 - *Authority pages* occur recurrently on good hubs for the subject.
- Best suited for “broad topic” queries rather than for page-finding queries.
- Gets at a broader slice of common *opinion*.

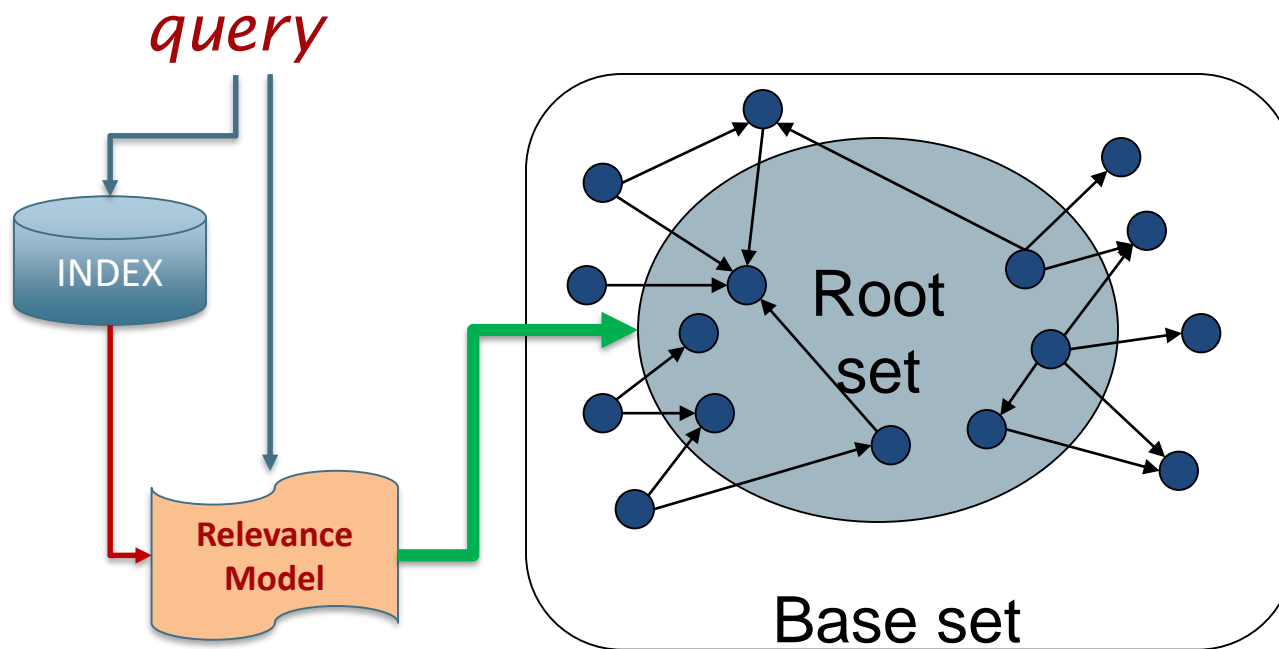
High-level scheme

- Extract from the web a **base set** of pages that *could* be good hubs or authorities.
 - From these, identify a small set of top hub and authority pages;

Base set

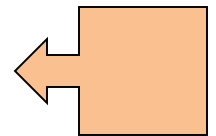
- Given text query (say ***browser***), use a text collection index to get all pages containing the term ***browser***.
 - Call this the root set of pages.
- Add in any page that either:
 - points to a page in the root set, or
 - is pointed to by a page in the root set.
- ✓ Call this the base set.

Visualization



Assembling the base set [Klei98]

- Root set typically 200-1000 nodes.
- Base set may have up to 5000 nodes.
- How do you find the base set nodes?
 - Follow out-links by parsing root set pages.
 - Get in-links (and out-links) from a *connectivity server*.
- (Actually, suffices to text-index strings of the form *href*=“URL” to get in-links to URL.)



Distilling hubs and authorities

➤ From the Base Set

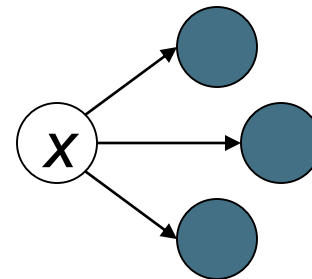
- Compute, for each page x in the base set, a hub score $h(x)$ and an authority score $a(x)$.
 - A. Initialize: for all x , $h(x) \leftarrow 1$; $a(x) \leftarrow 1$;
 - B. Iteratively update all $h(x)$, $a(x)$;
 - C. After each iteration
 - find page with highest $h()$ score (it is the top hub) – use it to normalize all hub scores!!!!
 - find page with highest $a()$ score (it is the as top authority) – use it to normalize all authority scores!!!



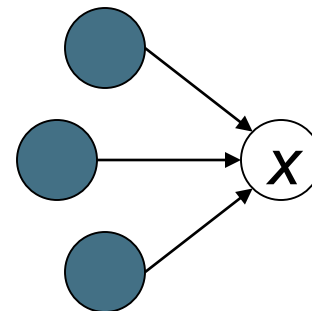
Iterative update

- Use the following updates, for all x :

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$



$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$



Scaling

- To prevent the $h()$ and $a()$ values from getting too big, we need to scale them down after each iteration.
- **Scaling factor** doesn't really matter:
 - we only care about the *relative* values of the scores.
 - SOLUTION: normalize all hub and authority values!

How many iterations?

- Claim: relative values of scores will converge after a few iterations:
 - in fact, suitably scaled, $h()$ and $a()$ scores settle into a steady state!
 - proof of this comes later.
- We only require the relative orders of the $h()$ and $a()$ scores - not their absolute values.
- In practice, ~ 5 iterations get you close to stability.

Constructing a Focused Subgraph of the WWW

- We can view any collection V of hyperlinked pages as a directed graph $G=(V,E)$: the nodes correspond to the pages, and a directed edge $(p,q) \in E$ indicates the presence of a link from p to q .
- We say that the *out-degree* of a node p is the number of nodes it has links to, and the *in-degree* of p is the number of nodes that have links to it. From a graph G , we can isolate **small regions**, or *subgraphs*, in the following way:
 - If $W \subseteq V$ is a subset of the pages, we use $G[W]$ to denote the graph *induced* on W : its nodes are the pages in W , and its edges correspond to all the links between pages in W .

Constructing a Focused Subgraph of the WWW

- Suppose we are given a broad-topic query, specified by a query string **s**.
- We wish to determine authoritative pages by an analysis of the link structure; but first we must determine the subgraph of the WWW on which our algorithm will operate. Our goal here is to focus the computational effort on relevant pages. Thus, for example, we could restrict the analysis to the set Q_s of all pages containing the query string; but this has two significant drawbacks:
 1. First this set may contain well over a million pages, and hence entail a considerable computational cost; and
 2. second, we have already noted that some or most of the best authorities may not belong to this set.

Properties

Ideally, we would like to focus on a collection S_s of pages with the following properties.

- (i) S_s is relatively small
- (ii) S_s is rich in relevant pages.
- (iii) S_s contains most (or many) of the strongest authorities.

By keeping S_s small, we are able to afford the computational cost of applying non-trivial algorithms; by ensuring it is rich in relevant pages we make it easier to find good authorities, as these are likely to be heavily referenced with S_s .

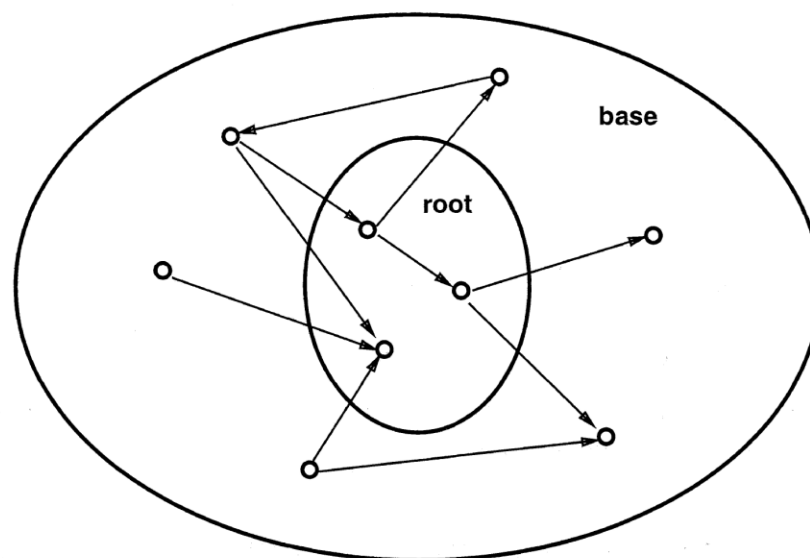
How can we find such a collection of pages? For a parameter t (typically set to about 200), we first collect the t highest-ranked pages for the query s from a text-based search engine such as Google, Bing or our project IR engine. We will refer to these t pages as the *root set* R_s .

The Root Set

This root set satisfies :

- (i) and (ii) of the desiderata ((i) S_s is relatively small; and (ii) S_s is rich in relevant pages), but it generally is far from satisfying (iii) (S_s contains most (or many) of the strongest authorities).
- To see this, note that the top t pages returned by the text-based search engines we use will all contain the query string s , and hence R_s is clearly a subset of the collection Q_s of *all* pages containing s .
- It is also interesting to observe that there are often extremely few links between pages in R_s , rendering it essentially “structureless”.
 - For example, in our experiments, the root set for the query “**java**” contained 15 links between pages in different domains; the root set for the query “**censorship**” contained 28 links between pages in different domains. These numbers are typical for a variety of the queries tried; they should be compared with the $200 \times 199 = 39800$ potential links that could exist between pages in the root set.

The root set and the base set.



Thus, we obtain S_s by growing R_s to include any page pointed to by a page in R_s and any page that points to a page in R_s

—with the restriction that we allow a single page in R_s to bring at most d pages pointing to it into S_s . This is crucial since a number of WWW pages are pointed to by several hundred thousand pages, and we can't include all of them in S_s if we wish to keep it reasonably small.

Graphs – Generating the Base SET!!!

We can use the root set R_s , however, to produce a set of pages S_s that will satisfy the conditions we're seeking. Consider a strong authority for the query topic—although it may well not be in the set R_s , it is quite likely to be *pointed to* by at least one page in R_s . Hence, we can increase the number of strong authorities in our subgraph by expanding R_s along the links that enter and leave it. In concrete terms, we define the following procedure.

HITS-graph(s, e, t, d)

s : a query string; e : a text-based search engine; t, d : natural numbers.

Let R_s denote the top t results of e on s .

Set $S_s := R_s$

For each page $p \in R_s$

Let $G^+(p)$ denote the set of all pages p points to.

Let $G^-(p)$ denote the set of all pages pointing to p .

Add all pages in $G^+(p)$ to S_s .

If $|G^-(p)| \leq d$ then

Add all pages in $G^-(p)$ to S_s .

Else

Add an arbitrary set of d pages from $G^-(p)$ to S_s .

Return S_s

Links in the Base Set

- First, let $G[S_s]$ denote, the subgraph induced on the pages in S_s . We distinguish between two types of links in $G[S_s]$. We say that a link is **transverse** if it is between pages with different domain names, and **intrinsic** if it is between pages with the same domain name.
 - By “domain name” here, we mean here the first level in the URL string associated with a page. Since intrinsic links very often exist purely to allow for navigation of the infrastructure of site, they convey much less information than transverse links about the authority of the pages they point to. Thus, we **delete all intrinsic** links from the graph $G[S_s]$, keeping only the edges corresponding to transverse links; this results in a graph G_s .
 - ✓ This is a very simple heuristic, but we find it effective for avoiding many of the pathologies caused by treating navigational links in the same way as other links.

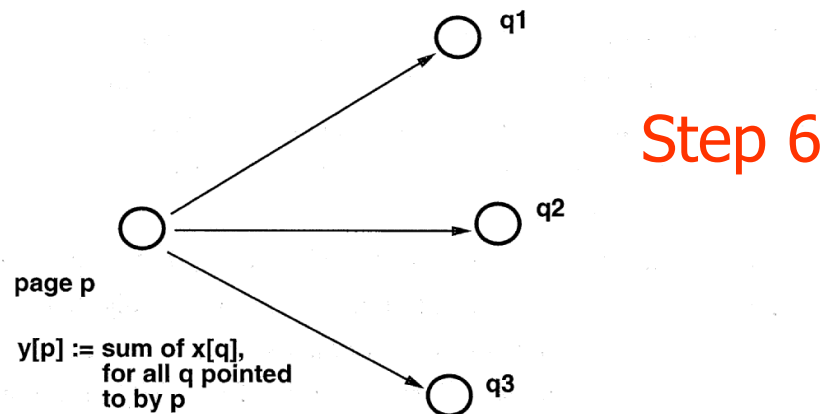
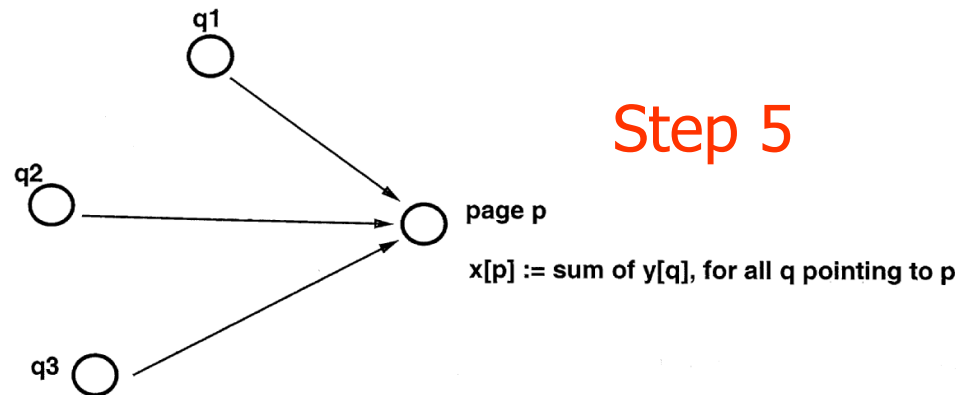
Steps in the HITS Algorithm

- 1) Let N be the set of nodes in the *Base Set graph*.
- 2) For every node n in N , let $H[n]$ be its hub score and $A[n]$ its authority score
- 3) Initialize $H[n]$ to 1 for all n in N . $A[n]$ to 1 as well.
- 4) *While vectors H and A have not converged*
- 5) *for all n in N :*

$$A[n] = \sum_{(n',n) \in E} H[n'] \quad n' \text{ points to } n$$
- 6) *for all n in N :*

$$H[n] = \sum_{(n',n) \in E} A[n'] \quad n \text{ points to } n'$$
- 7) **Normalize the H and A vectors**

The basic operations



Example of HITS

- Given a query q about a topic $t \rightarrow$ a collection C of Web-sites containing communities of hubs and authorities pertaining to t
 - A. Assemble to root set R
 - B. Apply a term based search engine (Google)
 - C. Assemble the base set B containing :
 - a) R ;
 - b) sites pointing to R ;
 - c) sites pointed by R .
 - For (b) we use a search engine again:
 - “Which sites point to [a given URL]”
 - Let $W = |B| \times |B|$ be the adjacency matrix of B .
 For each $s \in B$ we assign a pair of weights:
 hub-weight $h(S)$ and authority-weight $a(S)$

HITS in Action

1. Initialize $a(s) \leftarrow 1$, $h(s) \leftarrow 1$ for all $s \in C$
2. Update the authority weight (**I operation**)

$$a(s) \leftarrow \sum_{\{x|x \text{ points to } s\}} h(x)$$

3. Update the hub weight (**O operation**)

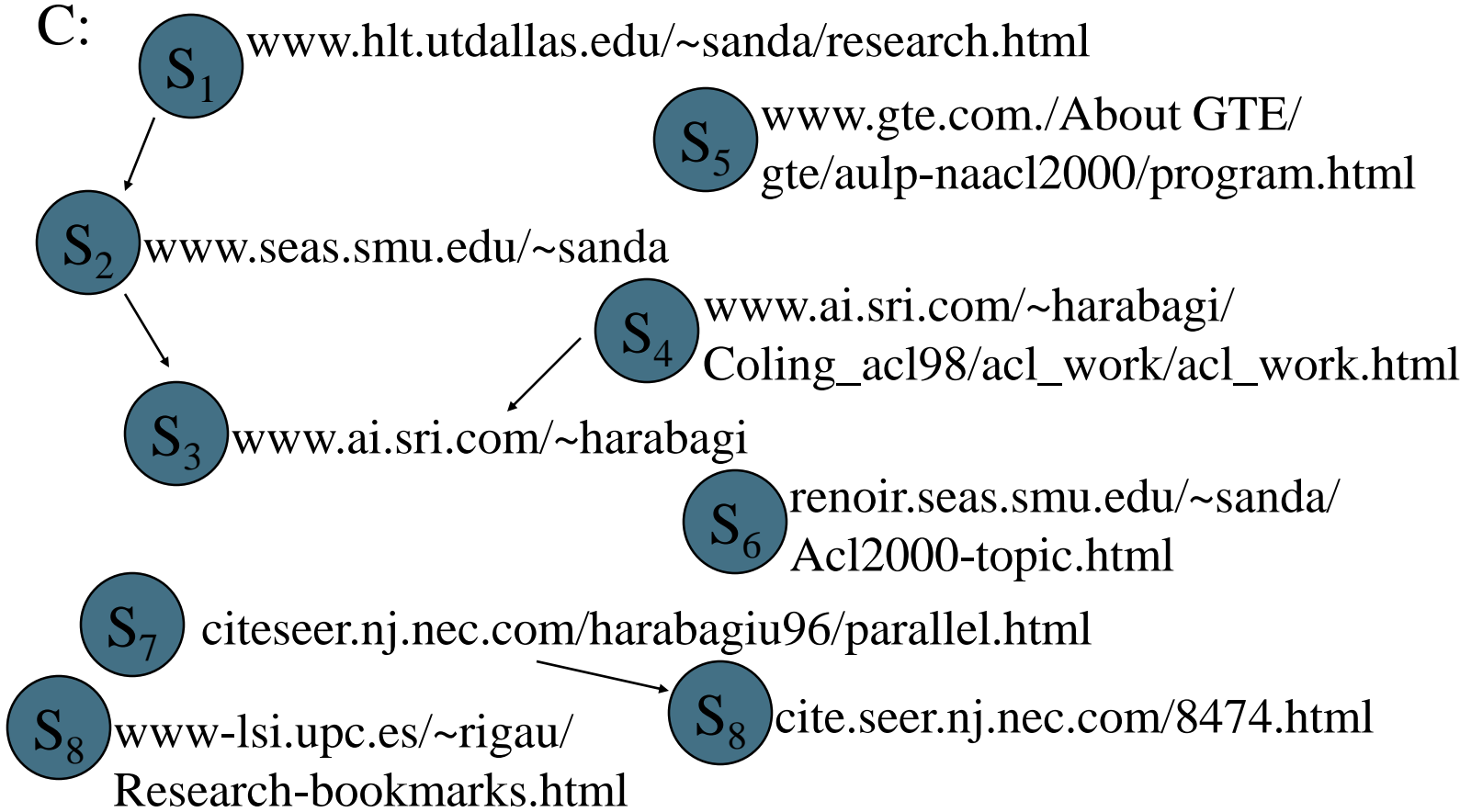
$$h(s) \leftarrow \sum_{\{x|s \text{ points to } x\}} a(x)$$

4. Normalize $a(s)$ and $h(s)$

Repeat steps 2, 3 and 4 until they converge.

Example of C

Query: research + “Sanda Harabagiu”



Simulation of HITS

Interpretation of Results:

$a(s_1)=0.88$ $a(s_2)=0.34$ $a(s_3)=0.2$ $a(s_4)=0.60$ $a(s_5)=0.34$

$a(s_6)=0.89$ $a(s_7)=0.11$ $a(s_8)=0.01$ $a(s_9)=0.05$

$h(s_1)=0.96$ $h(s_2)=0.54$ $h(s_3)=0.23$ $h(s_4)=0.02$ $h(s_5)=0.01$

$h(s_6)=0.03$ $h(s_7)=0.09$ $h(s_8)=0.02$ $h(s_9)=0.01$

.....

Link structure and the Adjacency Matrix

- Let us consider a web crawl C and its link structure that create a directed graph G .
 - G 's nodes are pages from C , and for all $i, j \in C$ the directed edge $i \rightarrow j$ appears in G if and only if a page i contains a hyperlink to a page j .
 - Let W denote the $|C| \times |C|$ adjacency matrix of G
 - **HITS** converges because of two well-know matrices used in the field of *bibliometrics*:
 - The **co-citation matrix**
 - The **bibliographic coupling matrix**

Co-citation Matrix and Bibliographic Coupling Matrix

- $A = W^T \times W$: co-citation matrix
- $[A]_{i,j}$ is the number of pages that jointly point at pages i and j
- HITS converges to authority weights which correspond to the principal eigenvector of A
- $H = W \times W^T$: bibliographic coupling matrix
- $[H]_{i,j}$ is the number of pages jointly referred to by pages i and j
- HITS converges to hub weights which correspond to the principal eigenvector of H

Problems with HITS

- 1) It considers only a relatively small part of the Web graph → adding edges to a few nodes can potentially change the resulting hubs and authority scores considerably
- 2) It is easier for authors of Web pages to manipulate the Page Rank score
- 3) If the neighborhood graph (Root Set) contains more pages on a topic different from the topic of the query, it can happen that the top authority and hub pages are on a different topic. The problem is called **topic drift**.

Notes

When applying the **I** operation

$$a(s) \leftarrow \sum_{\{x | x \text{ points to } s\}} h(x)$$

- Equivalent to assigning authority weights according to $[h] [w^T]$

■ When applying the **O** operation

$$h(s) \leftarrow \sum_{\{x | s \text{ points to } x\}} a(x)$$

- Equivalent to assigning hub score weights according to $[a] [w^T]$

Kleinberg showed that the algorithm converges:

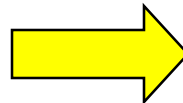
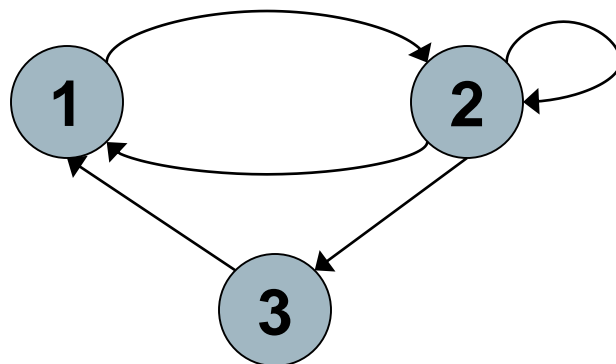
- Authority weights are the coordinates of the principal eigenvector of $w^T w$
- The hub weights are the coordinates of the principal eigenvector of $w w^T$

$A = w^T w \rightarrow$ co-citation matrix

$B = w w^T \rightarrow$ the bibliographic coupling matrix

Proof of convergence

- $n \times n$ adjacency matrix A :
 - each of the n pages in the base set has a row and column in the matrix.
 - Entry $A_{ij} = 1$ if page i links to page j , else $= 0$.



	1	2	3
1	0	1	0
2	1	1	1
3	1	0	0

Hub/authority vectors

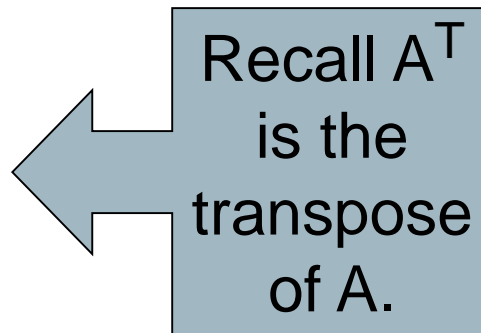
- View the hub scores $h()$ and the authority scores $a()$ as vectors with n components.
- Recall the iterative updates

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$

$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$

Rewrite in matrix form

- $\mathbf{h} = \mathbf{A}\mathbf{a}$.
- $\mathbf{a} = \mathbf{A}^T\mathbf{h}$.



Recall \mathbf{A}^T
is the
transpose
of \mathbf{A} .

Substituting, $\mathbf{h} = \mathbf{A}\mathbf{A}^T\mathbf{h}$ and $\mathbf{a} = \mathbf{A}^T\mathbf{A}\mathbf{a}$.

Thus, \mathbf{h} is an eigenvector of $\mathbf{A}\mathbf{A}^T$ and \mathbf{a} is an eigenvector of $\mathbf{A}^T\mathbf{A}$.

Further, our algorithm is a particular, known algorithm for computing eigenvectors: the *power iteration* method.



Guaranteed to converge.

Ranking Web Pages by Reputation

➤ Mendelzon & Rafrei – 2000

☐ Scenarios:

- Search engine Search-U-Matic just returned 60,000 pages on the query “liver disease”. Where should I start looking?
 - We’re spending \$200K/year maintaining our web pages. What topics are they known for?
 - Prof X, an expert on Icelandic sagas, is up for tenure. I wonder how well known her research is on the web.
- ❖ Idea: analyze links to find pages that are better known/
more authoritative than others *on some topics*.

Reputation measurement

➤ Given a page p , and topic t , compute the rank of p on t , denoted as $RM(p,t)$:

- 1) Let $l(t,p)$ = number of pages on topic t that point to p
- 2) Let N_t = number of pages on topic t

Then the reputation is obtained by computing:

$$RM(p,t) = l(t,p)/N_t$$

Random walks on the Web

- Consider a “random surfer” who wanders the Web, searching for pages on topic t .
 - At each step, the surfer either jumps into a page uniformly chosen at random from a set of pages that are about the topic t , or follows a link uniformly chosen at random from the set of outgoing links of the current page.
 - If the random surfer continues his walk forever, then the number of visits (s)he makes to a page is its reputation on t .
- Intuitively, pages with relatively high reputations on a topic are more likely to be visited by the random surfer searching for that topic.
 - A justification for this is that the reputation of a page on a topic naturally depends both on the number of pages on the same topic that point to it, and on the reputation of these pages on the same topic. The number of visits the surfer makes to a page depends on the same two factors.

Formal Model

- Define the reputation of a page p on topic t as the probability that the random surfer looking for topic t will visit page p .
 - Suppose that at each step, with probability d the surfer jumps into a page uniformly chosen at random from the set of pages that contain the term t , and with probability $(1-d)$ (s)he follows an outgoing link from the current page.
- Let N_t denote the total number of pages on the Web that contain the term t .
- Intuitively, the probability that the surfer at a step visits page p in a random jump is d/N_t if page p contains information about topic t and 0 otherwise.

Formal reputation

- Let $q \rightarrow p$ denote a link from page q to page p ; and $O(q)$ denote the number of outgoing links from page q .
 - The probability that the surfer visits page p at step n after visiting page q and through the link $q \rightarrow p$ is:

$$R^n(p, t) = \frac{1-d}{O(q)} \times R^{n-1}(q, t), \text{ where } R^{n-1}(q, t) \text{ denotes the probability that the surfer visits page } q \text{ for topic } t \text{ at step } n-1.$$
 - The probability of visiting page p for topic t at step n of the walk is:

$$R^n(p, t) = \begin{cases} \frac{d}{N_t} + (1-d) \sum_{q \rightarrow p} \frac{R^{n-1}(q, t)}{O(q)} & \text{if term } t \text{ appears in page } p \\ (1-d) \sum_{q \rightarrow p} \frac{R^{n-1}(q, t)}{O(q)} & \text{otherwise} \end{cases}$$

One-level reputation

- It is the equilibrium probability of visiting page p for topic t :

$$\pi_{p,t} = \lim_{n \rightarrow \infty} R^n(p, t)$$

Unique probability, as demonstrated by Davood Rafiei and Alberto Mendelzon in **"What is a Page Known for? Computing Web Page Reputations"**.

Hubs and Authorities

- We define the *authority reputation* of a page p on a topic t as the probability that the random surfer looking for topic t makes a forward visit to page p ; and
- The hub reputation of a page p on a topic t as the probability that the random surfer looking for topic t makes a backward visit to page p .

Computing the reputation of hubs and authorities

Suppose at each step, with probability d the random surfer picks a direction and jumps into a page uniformly chosen at random from a set of pages on topic t , and with probability $(1-d)$ the surfer follows a link.

- The probability that the surfer makes a forward/backward visit to page p is $d/2N_t$ if the page p contains the term t .
- Let $q \rightarrow p$ denote a link from page q to page p ; and $O(p)$ denote the number of outgoing links from page p . Let $I(p)$ denote the number of incoming links into page p .

Let $A^{n-1}(p, t)$ denote the probability of a forward visit to page p at step $n-1$ and $H^{n-1}(p, t)$ the probability of a backward visit into page p at step $n-1$.

- The probability that the surfer makes a forward/backward visit to page p at step n after visiting page q and through the link $q \rightarrow p$ is:
 - $((1-d)/O(q))H^{n-1}(q, t)$ (forward)
 - $((1-d)/O(q))A^{n-1}(q, t)$ (backward)