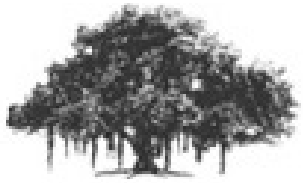


IIIT-H

Web Mining
Lecture 4: Link Analysis Algorithms

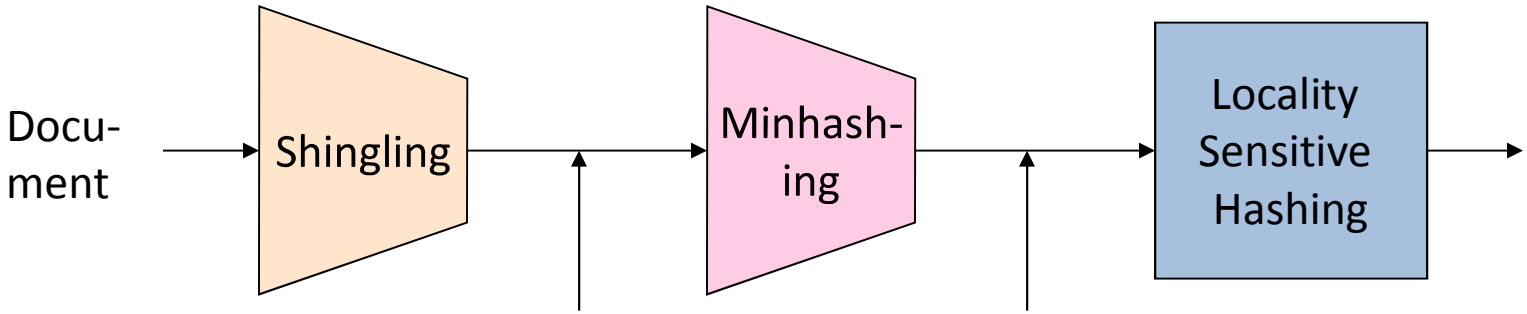
Manish Gupta

10th Aug 2013

Slides borrowed (and modified) from
http://infolab.stanford.edu/~ullman/mining/2009/index.html
http://www.mpi-inf.mpg.de/departments/d5/teaching/ws11_12/irdm/slides/irdm-4-2-4.pptx

Recap of Lecture 3: Similarity Search

- Shingling
- Min-hash
- Locally Sensitive Hashing (LSH)
- Applications of LSH



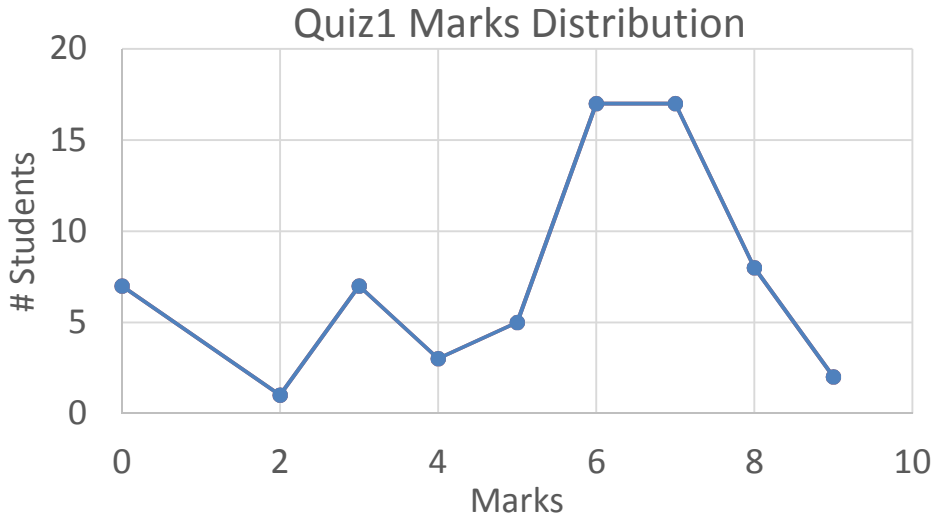
Candidate pairs :
Those pairs of signatures that we need to test for similarity.

The set of strings of length *k* that appear in the doc-ument

Signatures :
Short integer vectors that represent the sets, and reflect their similarity

Surprise Quiz 1 Analysis

- Class Strength
  - 27 PG, 4 PGSSP, 2 PhD, 33 UG = 66
- Positional Index: The dictionary stores the number of documents containing the term and not the number of occurrences of the word.
- 6 surprise quizzes (of which best 4 will count towards final grade)



Averages

- PG – 6.15
- PGSSP – 0.75
- UG – 5.47
- PhD – 5
- Overall – 5.45

Today's Agenda

- PageRank
- Topic-Specific PageRank
- HITS (Hypertext-Induced Topic Selection)
- Spam Detection Algorithms: TrustRank

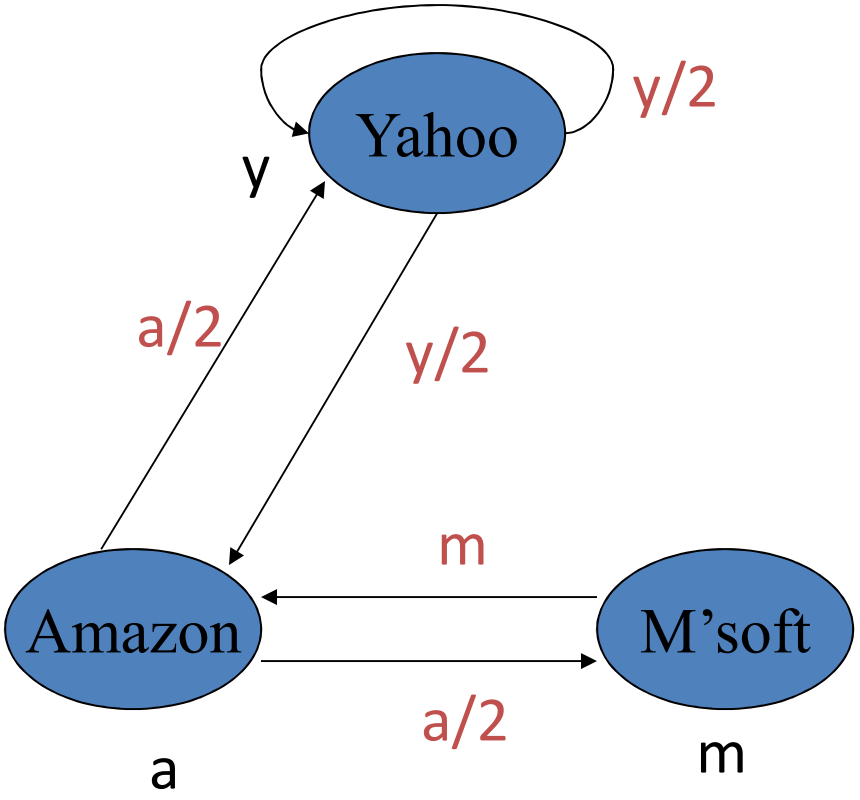
Today's Agenda

- **PageRank**
- Topic-Specific PageRank
- HITS (Hypertext-Induced Topic Selection)
- Spam Detection Algorithms: TrustRank

Ranking Web Pages

- Web pages are not equally "important"
  - www.joe-schmoe.com vs www.stanford.edu
- Inlinks as votes
  - www.stanford.edu has 23,400 inlinks
  - www.joe-schmoe.com has 1 inlink
- Are all inlinks equal?
  - Recursive question
  - Each link's vote is proportional to the importance of its source page
  - If page P with importance x has n outlinks, each link gets x/n votes
  - Page P's own importance is the sum of the votes on its inlinks

Simple "Flow" Model



$$y = y/2 + a/2$$
$$a = y/2 + m$$
$$m = a/2$$

Solving the Flow Equations

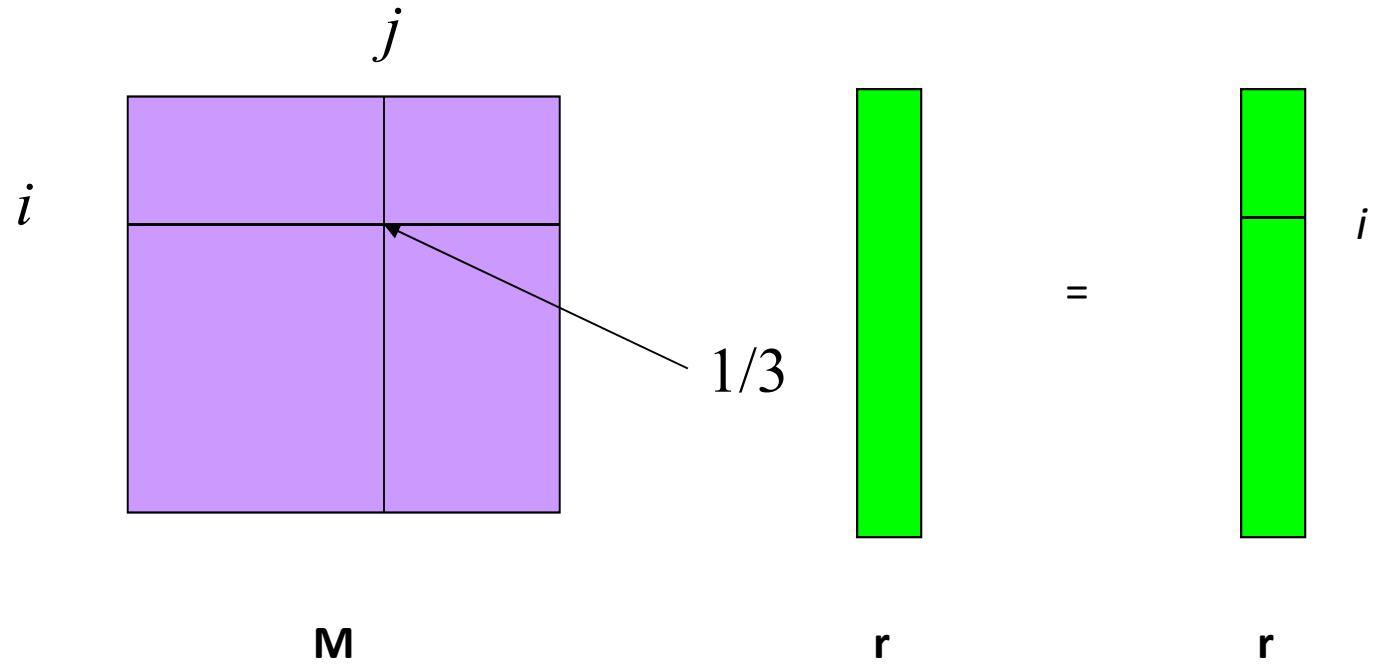
- 3 equations, 3 unknowns, no constants
  - No unique solution
  - All solutions equivalent modulo scale factor
- Additional constraint forces uniqueness
  - y+a+m = 1
  - y = 2/5, a = 2/5, m = 1/5
- Gaussian elimination method works for small examples, but we need a better method for large graphs

Matrix Formulation

- Matrix **M** has one row and one column for each web page
- Suppose page j has n outlinks
  - If j != i, then M_{ij}=1/n
  - Else M_{ij}=0
- **M** is a column stochastic matrix
  - Columns sum to 1
- Suppose **r** is a vector with one entry per web page
  - r_i is the importance score of page i
  - Call it the rank vector
  - $|\mathbf{r}| = 1$

Example

Suppose page j links to 3 pages, including i



$$y = y/2 + a/2$$

$$a = y/2 + m$$

$$m = a/2$$

Eigenvector Formulation

- The flow equations can be written as

$$\mathbf{r} = \mathbf{M}\mathbf{r}$$

- So the rank vector is an eigenvector of the stochastic web matrix
  - In fact, its first or principal eigenvector, with corresponding eigenvalue 1

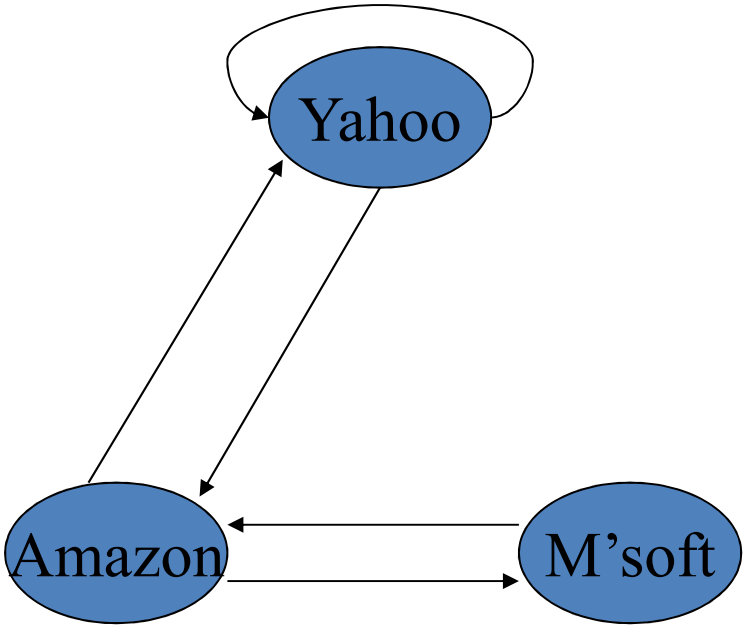
$$y = y/2 + a/2$$
$$a = y/2 + m$$
$$m = a/2$$

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

Power Iteration Method

- Simple iterative scheme (aka relaxation)
- Suppose there are N web pages
- Initialize: $\mathbf{r}^0 = [1/N,\ldots.,1/N]^T$
- Iterate: $\mathbf{r}^{k+1} = \mathbf{M}\mathbf{r}^k$
- Stop when $|\mathbf{r}^{k+1} - \mathbf{r}^k|_1 < \varepsilon$
  - $|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1 norm
  - Can use any other vector norm e.g., Euclidean

Power Iteration Example



	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

y		1/3	1/3	5/12	3/8		2/5
a	=	1/3	1/2	1/3	11/24	. . .	2/5
m		1/3	1/6	1/4	1/6		1/5

Random Walk Interpretation

- Imagine a random web surfer
  - At any time t, surfer is on some page P
  - At time t+1, the surfer follows an outlink from P uniformly at random
  - Ends up on some page Q linked from P
  - Process repeats indefinitely
- Let **p**(t) be a vector whose i^{th} component is the probability that the surfer is at page i at time t
  - **p**(t) is a probability distribution on pages

The Stationary Distribution

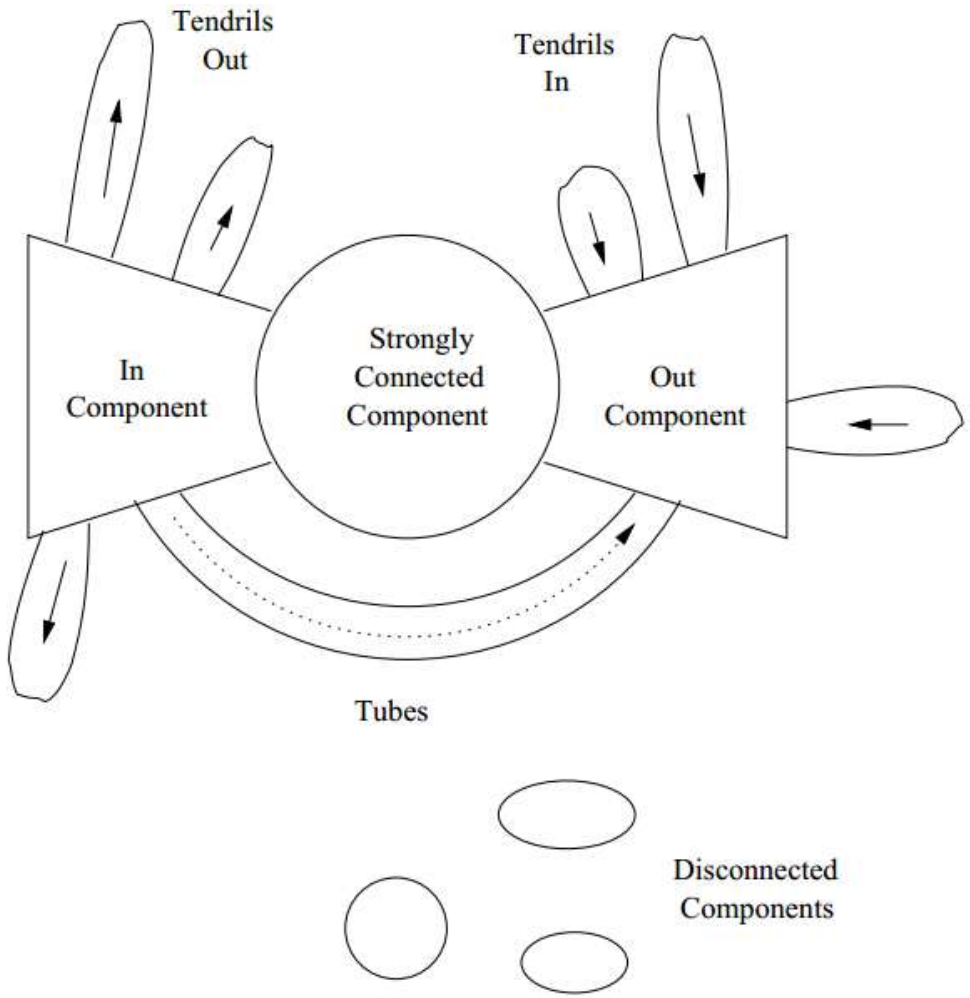
- Where is the surfer at time t+1?
  - Follows a link uniformly at random
  - $\mathbf{p}(t+1) = \mathbf{M}\mathbf{p}(t)$
- Suppose the random walk reaches a state such that $\mathbf{p}(t+1) = \mathbf{M}\mathbf{p}(t) = \mathbf{p}(t)$
  - Then $\mathbf{p}(t)$ is called a stationary distribution for the random walk
- Our rank vector $\mathbf{r}$ satisfies $\mathbf{r} = \mathbf{M}\mathbf{r}$
  - So it is a stationary distribution for the random surfer

Existence and Uniqueness

A central result from the theory of random walks (aka Markov processes):

For graphs that satisfy certain conditions (ergodicity, i.e., aperiodicity and irreducibility), the stationary distribution is unique and eventually will be reached no matter what the initial probability distribution at time t = 0.

Bow-tie Structure of the Web



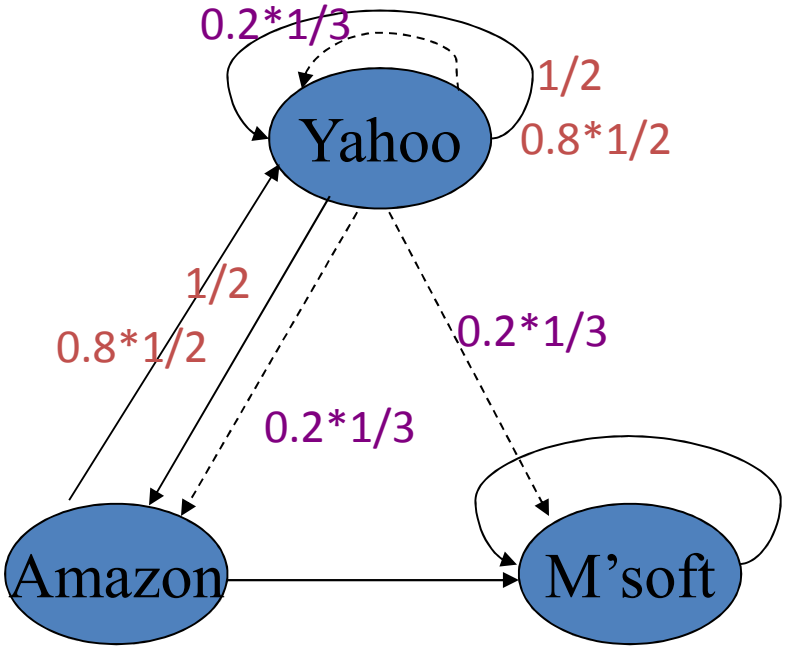
Spider Traps

- A group of pages is a spider trap if there are no links from within the group to outside the group
  - Random surfer gets trapped
- Spider traps violate the conditions needed for the random walk theorem

Random Teleports

- Solution for spider traps
- At each time step, the random surfer has two options:
  - With probability β, follow a link at random
  - With probability 1-β, jump to some page uniformly at random
  - 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

Random Teleports ($\beta = 0.8$)

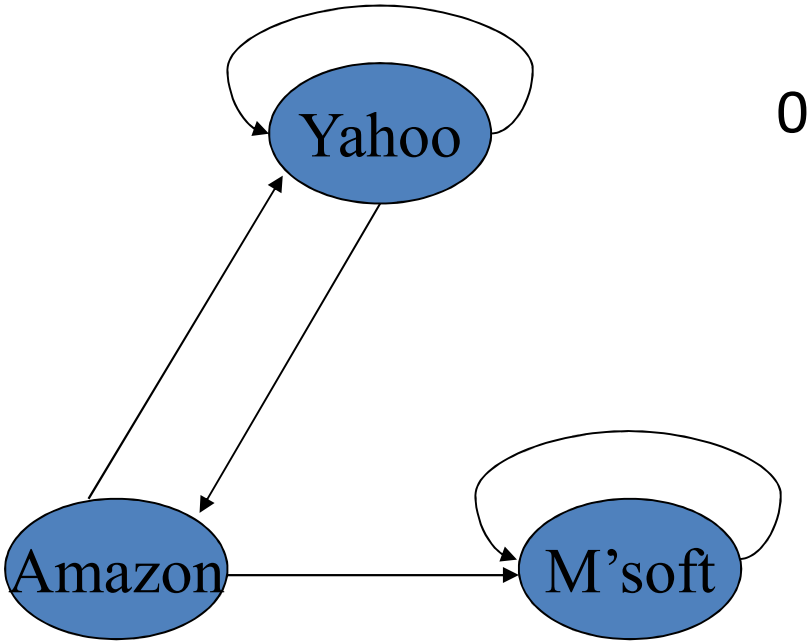


$$\begin{array}{c} \\ y \\ a \\ m \end{array}\begin{array}{c} y \\ \boxed{\begin{array}{c} 1/2 \\ 1/2 \\ 0 \end{array}} \end{array} \quad 0.8\begin{array}{c} y \\ \boxed{\begin{array}{c} 1/2 \\ 1/2 \\ 0 \end{array}} \end{array} \quad + 0.2\begin{array}{c} y \\ \boxed{\begin{array}{c} 1/3 \\ 1/3 \\ 1/3 \end{array}} \end{array}$$

$$0.8\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$\begin{array}{c} y \\ a \\ m \end{array}\begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

Random Teleports ($\beta = 0.8$)



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$\begin{array}{c} \\ y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

y		1	1.00	0.84	0.776	7/11
a	=	1	0.60	0.60	0.536	. . . 5/11
m		1	1.40	1.56	1.688	21/11

Matrix Formulation

- Suppose there are N pages
  - Consider a page j, with set of outlinks O(j)
  - We have M_{ij} = 1/|O(j)| when j!=i and M_{ij} = 0 otherwise
  - The random teleport is equivalent to
    - adding a teleport link from j to every other page with probability (1-β)/N
    - reducing the probability of following each outlink from 1/|O(j)| to β/|O(j)|
    - Equivalent: tax each page a fraction (1-β) of its score and redistribute evenly

PageRank

- Construct the N×N matrix **A** as follows
  - $A_{ij} = \beta M_{ij} + (1-\beta)/N$
- **A** is a stochastic matrix
- The PageRank vector **r** is the principal eigenvector of this matrix
  - satisfying **r** = **Ar**
- Equivalently, **r** is the stationary distribution of the random walk with teleports

Dealing with Dead-ends

- Pages with no outlinks are "dead ends" for the random surfer
- Teleport
  - Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly
- Prune and propagate
  - Preprocess the graph to eliminate dead-ends
  - Might require multiple passes
  - Compute PageRank on reduced graph
  - Approximate values for deadends by propagating values from reduced graph

Some Problems with PageRank

- Measures generic popularity of a page
  - Biased against topic-specific authorities
  - Ambiguous queries e.g., jaguar
- Uses a single measure of importance
  - Other models e.g., hubs-and-authorities
- Susceptible to Link spam
  - Artificial link topographies created in order to boost PageRank

Today's Agenda

- PageRank
- **Topic-Specific PageRank**
- HITS (Hypertext-Induced Topic Selection)
- Spam Detection Algorithms: TrustRank

Topic-sensitive PageRank

- **r = βMr+ (1-β)p**
- **Conventional PageRank: p** is a uniform vector with values **1/N**
- Topic-sensitive PageRank uses a **non-uniform** personalization vector **p**
- Not simply a post-processing step of the PageRank computation
- Personalization vector **p** introduces bias in all iterations of the iterative computation of the PageRank vector

Personalization Vector

- In the random-walk model, the personalization vector represents the addition of a set of transition edges, where the probability of an artificial edge **(u,v)** is **$(1-\beta)p_v$**
- Given a graph the result of the PageRank computation only depends on β and **p** : **PR(β,p)**

Topic-sensitive PageRank: Overall Approach

- Preprocessing
  - Fix a set of **k** topics
  - For each topic c_j compute the PageRank scores of page **u** wrt to the **j**-th topic: **r(u,j)**
- Query-time processing:
  - For query q compute the total score of page **u** wrt **q** as **score(u,q)** = $\Sigma_{j=1\ldots k}$ **Pr(c_j|q) r(u,j)**

Topic-sensitive PageRank: Preprocessing

- Create **k** different biased PageRank vectors using some pre-defined set of k categories **(c_1,…,c_k)**
  - E.g., Open Directory (DMOZ)'s 16 top level categories like sports, medicine, etc.
- **T_j**: set of URLs in the **j**-th category
- Use non-uniform personalization vector **$p=w_j$** such that:

$$w_j(v) = \begin{cases} \frac{1}{T_j}, v \in T_j \\ 0, \text{o/w} \end{cases}$$

Topic-sensitive PageRank: Query-time Processing

- D_j: class term vectors consisting of all the terms appearing in the **k** pre-selected categories

$$\Pr(c_j \mid q) = \frac{\Pr(c_j)\Pr(q \mid c_j)}{\Pr(q)} \propto \Pr(c_j)\prod_i \Pr(q_i \mid c_j)$$

- How can we compute $P(c_j)$?
  - Can be fixed as uniform
  - Can be biased to a particular set of categories if we have that information about the user
- How can we compute $\Pr(q_i \mid c_j)$?
  - From the class term vector D_j

Today's Agenda

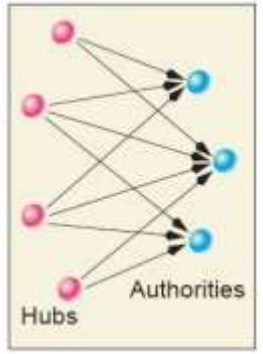
- PageRank
- Topic-Specific PageRank
- **HITS (Hypertext-Induced Topic Selection)**
- Spam Detection Algorithms: TrustRank

Hubs and Authorities

- Suppose we are given a collection of documents on some broad topic
  - e.g., stanford, evolution, iraq
  - perhaps obtained through a text search
- Can we organize these documents in some manner?
  - PageRank offers one solution
  - HITS (Hypertext-Induced Topic Selection) is another
    - proposed at approx the same time (1998)

HITS Model

- Interesting documents fall into two classes
- Authorities are pages containing useful information
  - course home pages
  - home pages of auto manufacturers
- Hubs are pages that link to authorities
  - course bulletin
  - list of US auto manufacturers

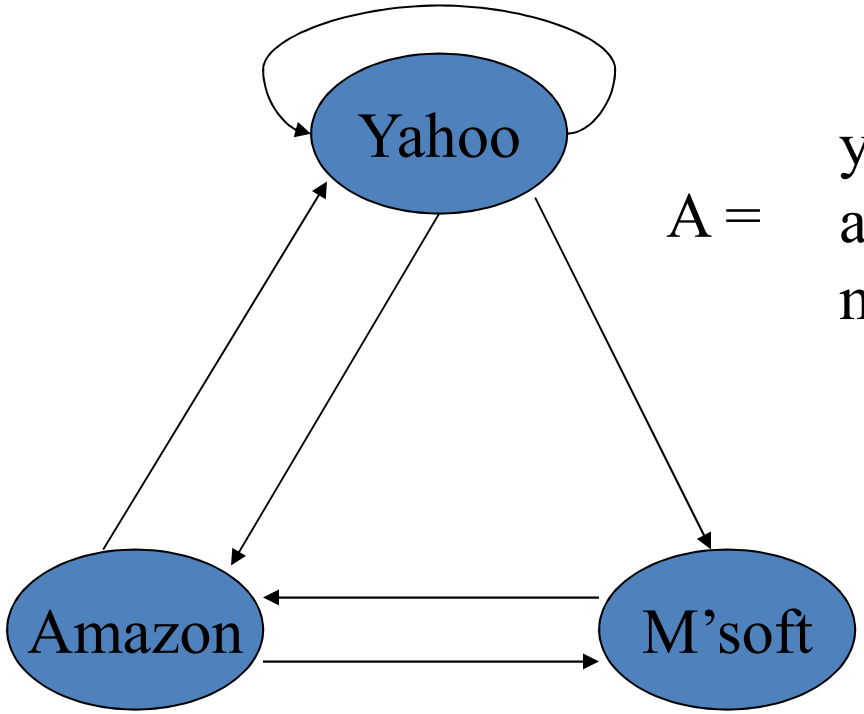


Mutually Recursive Definition

- A good hub links to many good authorities
- A good authority is linked from many good hubs
- Model using two scores for each node
  - Hub score and Authority score
  - Represented as vectors **h** and **a**

Transition Matrix *A*

- HITS uses a matrix *A*[*i*, *j*] = 1 if page *i* links to page *j*, 0 if not
- A^T, the transpose of *A*, is similar to the PageRank matrix *M*, but A^T has 1's where *M* has fractions



A =

	y	a	m
y	1	1	1
a	1	0	1
m	0	1	0

Hub and Authority Equations

- The hub score of page P is proportional to the sum of the authority scores of the pages it links to
  - $\mathbf{h} = \lambda A\mathbf{a}$
  - Constant λ is a scale factor
- The authority score of page P is proportional to the sum of the hub scores of the pages it is linked from
  - $\mathbf{a} = \mu A^T\mathbf{h}$
  - Constant μ is scale factor

Iterative Algorithm

- Initialize **h**, **a** to all 1's
- **Iterate until h and a converge**
  - **h** = **Aa**
  - Scale **h** so that its max entry is 1.0
  - **a** = $\mathbf{A}^T\mathbf{h}$
  - Scale **a** so that its max entry is 1.0

Example

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

a(yahoo)	=	1	1	1	1	...	1
a(amazon)	=	1	1	4/5	0.75	...	0.732
a(m'soft)	=	1	1	1	1	...	1
h(yahoo)	=	1	1	1	1	...	1.000
h(amazon)	=	1	2/3	0.71	0.73	...	0.732
h(m'soft)	=	1	1/3	0.29	0.27	...	0.268

Existence and Uniqueness

- $\mathbf{h} = \lambda A\mathbf{a}$ and $\mathbf{a} = \mu A^T \mathbf{h}$
- $\mathbf{h} = \lambda\mu AA^T \mathbf{h}$ **and** $\mathbf{a} = \lambda\mu A^T A\ \mathbf{a}$
- Under reasonable assumptions about **A**, the dual iterative algorithm converges to vectors **h*** and **a*** such that
  - **h*** is the principal eigenvector of the matrix AA^T
  - **a*** is the principal eigenvector of the matrix $A^T A$

PageRank and HITS

- PageRank and HITS are two solutions to the same problem
  - What is the value of an inlink from S to D?
  - In the PageRank model, the value of the link depends on the links **into** S
  - In the HITS model, it depends on the value of the other links **out of** S

Comparison of PageRank & HITS

	PR	HITS
Matrix construction	static	query time
Matrix size	huge	moderate
Stochastic matrix	yes	no
Dampening by random jumps	yes	no
Outdegree normalization	yes	no
Preference for bipartite cores	no	yes
Score stability to perturbations	yes	no
Resilience to topic drift	n/a	no
Resilience to spam	no	no

Today's Agenda

- PageRank
- Topic-Specific PageRank
- HITS (Hypertext-Induced Topic Selection)
- **Spam Detection Algorithms: TrustRank**

What is Web Spam?

- Spamming = any deliberate action solely in order to boost a web page's position in search engine results, incommensurate with page's real value
- Spam = web pages that are the result of spamming
- This is a very broad defintion
  - SEO industry might disagree!
  - SEO = search engine optimization
- Approximately 10-15% of web pages are spam

Web Spam Taxonomy

- We follow the treatment by Gyongyi and Garcia-Molina [2004]
- Boosting techniques
  - Techniques for achieving high relevance/importance for a web page
- Hiding techniques
  - Techniques to hide the use of boosting
    - From humans and web crawlers

Boosting Techniques

- Term spamming
  - Manipulating the text of web pages in order to appear relevant to queries
- Link spamming
  - Creating link structures that boost PageRank or hubs and authorities scores

Term Spamming

- Repetition
  - of one or a few specific terms e.g., free, cheap, viagra
  - Goal is to subvert TF.IDF ranking schemes
- Dumping
  - of a large number of unrelated terms
  - e.g., copy entire dictionaries
- Weaving
  - Copy legitimate pages and insert spam terms at random positions
- Phrase Stitching
  - Glue together sentences and phrases from different sources

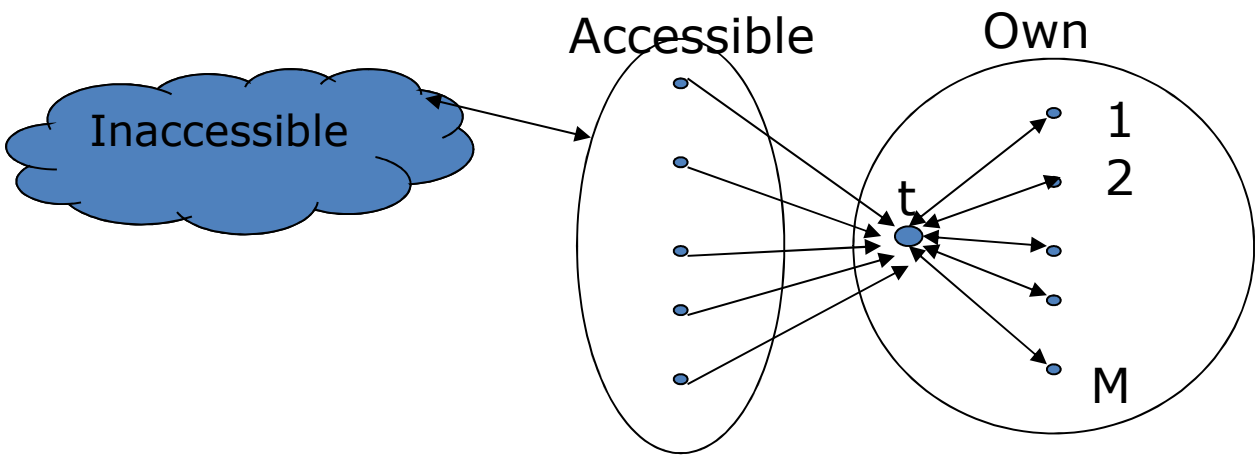
Link Spam

- Three kinds of web pages from a spammer's point of view
  - Inaccessible pages
  - Accessible pages
    - e.g., web log comments pages
    - spammer can post links to his pages
  - Own pages
    - Completely controlled by spammer
    - May span multiple domain names

Link Farms

- Spammer's goal
  - Maximize the PageRank of target page t
- Technique
  - Get as many links from accessible pages as possible to target page t
  - Construct "link farm" to get PageRank multiplier effect

Link Farm Analysis



Suppose rank contributed by accessible pages = x

Let PageRank of target page = y

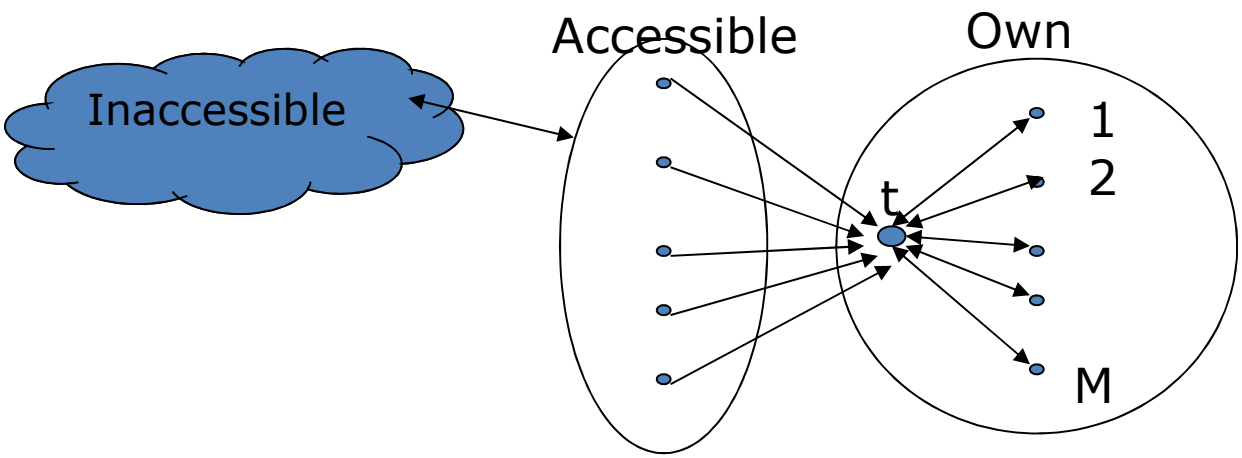
Rank of each "farm" page = $\beta y/M + (1-\beta)/N$

$y = x + \beta M[\beta y/M + (1-\beta)/N] + (1-\beta)/N$

$= x + \beta^2 y + \beta(1-\beta)M/N + (1-\beta)/N$ Very small; ignore

$y = x/(1-\beta^2) + cM/N$ where $c = \beta/(1+\beta)$

Analysis



- $y = x/(1-\beta^2) + cM/N$ where $c = \beta/(1+\beta)$
- For $\beta = 0.85$, $1/(1-\beta^2) = 3.6$
  - Multiplier effect for "acquired" PageRank
  - By making M large, we can make y as large as we want

Detecting Spam

- Term spamming
  - Analyze text using statistical methods e.g., Naïve Bayes classifiers
  - Similar to email spam filtering
  - Also useful: detecting approximate duplicate pages
- Link spamming
  - One approach: TrustRank

TrustRank Idea

- Basic principle: approximate isolation
  - It is rare for a "good" page to point to a "bad" (spam) page
- Sample a set of "seed pages" from the web
- Have an oracle (human) identify the good pages and the spam pages in the seed set
  - Expensive task, so must make seed set as small as possible

Trust Propagation

- Call the subset of seed pages that are identified as “good” the “trusted pages”
- Set trust of each trusted page to 1
- Propagate trust through links
  - Each page gets a trust value between 0 and 1
  - Use a threshold value and mark all pages below the trust threshold as spam

Rules for Trust Propagation

- Trust attenuation
  - The degree of trust conferred by a trusted page decreases with distance
- Trust splitting
  - The larger the number of outlinks from a page, the less scrutiny the page author gives each outlink
  - Trust is “split” across outlinks

Simple Model

- Suppose trust of page p is t(p)
  - Set of outlinks O(p)
- For each $q \in O(p)$, p confers the trust
  - $\beta t(p)/|O(p)|$ for $0<\beta<1$
- Trust is additive
  - Trust of p is the sum of the trust conferred on p by all its inlinked pages
- Note similarity to Topic-Specific PageRank
  - Within a scaling factor, trust rank = biased PageRank with trusted pages as teleport set

Picking the Seed Set

- Two conflicting considerations
  - Human has to inspect each seed page, so seed set must be as small as possible
  - Must ensure every “good page” gets adequate trust rank, so need make all good pages reachable from seed set by short paths

Approaches to Picking Seed Set

- Suppose we want to pick a seed set of k pages
- PageRank
  - Pick the top k pages by PageRank
  - Assume high PageRank pages are close to other highly ranked pages
  - We care more about high PageRank “good” pages

Inverse PageRank

- Pick the pages with the maximum number of outlinks
- Can make it recursive
  - Pick pages that link to pages with many outlinks
- Formalize as "inverse PageRank"
  - Construct graph G' by reversing each edge in web graph G
  - PageRank in G' is inverse PageRank in G
- Pick top k pages by inverse PageRank

Spam Mass

- In the TrustRank model, we start with good pages and propagate trust
- Complementary view: what fraction of a page's PageRank comes from "spam" pages?
- In practice, we don't know all the spam pages, so we need to estimate

Spam Mass Estimation

- $r(p)$ = PageRank of page p
- $r^{+}(p)$ = PageRank of p with teleport into "good" pages only
- $r^{-}(p) = r(p) - r^{+}(p)$
- Spam mass of p = $r^{-}(p)/r(p)$
- Pages could be ranked based on their spam mass too

Good Pages

- For spam mass, we need a large set of "good" pages
  - Need not be as careful about quality of individual pages as with TrustRank
- One reasonable approach
  - .edu sites
  - .gov sites
  - .mil sites

Take-away Messages

- Linkage on the web is an important phenomena
- Links contain a lot of knowledge
- Knowledge in the link structure can be used to rank webpages
  - Using PageRank
  - Using Topic Sensitive PageRank
  - Using HITS
- Ways of webpage spamming
- Ways of handling webpage spamming
  - Trust Rank
  - Spam mass

Further Reading

- A nice summary of link analysis algorithms from John Kleinberg
  - http://dl.acm.org/citation.cfm?id=345982
- Chapter 5: "Link Analysis" from Mining of Massive Datasets
  - http://infolab.stanford.edu/~ullman/mmds.html
- Chapter 7 (Social Network Analysis) from Mining the Web
  - http://www.cse.iitb.ac.in/soumen/mining-the-web/
- J.M. Kleinberg: Authoritative Sources in a Hyperlinked Environment, JACM 46(5), 1999
- S Brin, L. Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine, WWW 1998
- M. Najork, H. Zaragoza, M. Taylor: HITS on the Web: How does it Compare?, SIGIR 2007
- R. Lempel, S. Moran: SALSA: The Stochastic Approach for Link-Structure Analysis, ACM TOIS 19(2), 2001.
- G. Jeh, J. Widom: SimRank: a Measure of Structural-Context Similarity, KDD 2002
- Taher Haveliwala: Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search, IEEE Trans. on Knowledge and Data Engineering, 2003.
- G. Jeh, J. Widom: Scaling personalized web search, WWW 2003.
- D. Fogaras, B. Racz, K. Csalogany, A. Benczur: Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments, Internet Mathematics 2(3): 333-358, 2006.

Preview of Lecture 5: LSI and EM

- Singular Value Decomposition (SVD)
- Latent Semantic Indexing (LSI)
- K-Means
- Expectation Maximization (EM)

Disclaimers

- This course will represent opinions of the instructor. It does not reflect views of Microsoft or any other entity.
- Algorithms, techniques, features, etc mentioned here might or might not be in use by Microsoft or any other company.
- Lot of material covered in this course is borrowed from slides across many universities and conference tutorials. These are gratefully acknowledged.

Thanks!

Computing PageRank

- Key step is matrix-vector multiplication
  - $\mathbf{r}^{new} = \mathbf{A}\mathbf{r}^{old}$
- Easy if we have enough main memory to hold $\mathbf{A}$, $\mathbf{r}^{old}$, $\mathbf{r}^{new}$
- Say N = 1 billion pages
  - We need 4 bytes for each entry (say)
  - 2 billion entries for vectors, approx 8GB
  - Matrix A has N^2 entries
    - 10^{18} is a large number!

Rearranging the Equation

$\mathbf{r} = \mathbf{Ar}$, where

$A_{ij} = \beta M_{ij} + (1-\beta)/N$

$r_i = \sum_{1 \le j \le N} A_{ij} r_j$

$r_i = \sum_{1 \le j \le N} [\beta M_{ij} + (1-\beta)/N] r_j$

$= \beta \sum_{1 \le j \le N} M_{ij} r_j + (1-\beta)/N \sum_{1 \le j \le N} r_j$

$= \beta \sum_{1 \le j \le N} M_{ij} r_j + (1-\beta)/N$, since $|\mathbf{r}| = 1$

$\mathbf{r} = \beta \mathbf{Mr} + [(1-\beta)/N]_N$

where $[x]_N$ is an N-vector with all entries x

Sparse Matrix Formulation

- We can rearrange the PageRank equation:
  - $\mathbf{r} = \beta\mathbf{M}\mathbf{r} + [(1-\beta)/N]_N$
  - $[(1-\beta)/N]_N$ is an N-vector with all entries $(1-\beta)/N$
- **M** is a sparse matrix!
  - 10 links per node, approx 10N entries
- So in each iteration, we need to:
  - Compute $\mathbf{r}^{new} = \beta\mathbf{M}\mathbf{r}^{old}$
  - Add a constant value $(1-\beta)/N$ to each entry in $\mathbf{r}^{new}$

Sparse Matrix Encoding

- Encode sparse matrix using only nonzero entries
  - Space proportional roughly to number of links
  - say 10N, or 4*10*1 billion = 40GB
  - still won't fit in memory, but will fit on disk

source node	degree	destination nodes
0	3	1, 5, 7
1	5	17, 64, 113, 117, 245
2	2	13, 23

Basic Algorithm

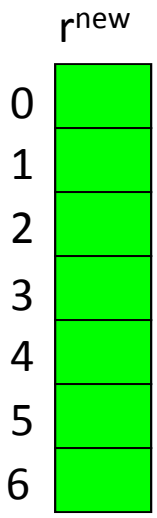
- Assume we have enough RAM to fit $\mathbf{r}^{new}$, plus some working memory
  - Store $\mathbf{r}^{old}$ and matrix $\mathbf{M}$ on disk

Basic Algorithm:

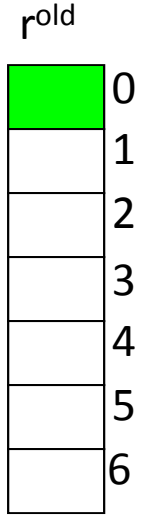
- Initialize: $\mathbf{r}^{old} = [1/N]_N$
- Iterate:
  - Update: Perform a sequential scan of $\mathbf{M}$ and $\mathbf{r}^{old}$ to update $\mathbf{r}^{new}$
  - Write out $\mathbf{r}^{new}$ to disk as $\mathbf{r}^{old}$ for next iteration
  - Every few iterations, compute $|\mathbf{r}^{new}-\mathbf{r}^{old}|$ and stop if it is below threshold
    - Need to read in both vectors into memory

Update Step

Initialize all entries of $\mathbf{r}^{new}$ to $(1-\beta)/N$
For each page p (out-degree n):
    Read into memory: p, n, $dest_1,\ldots,dest_n$, $r^{old}(p)$
    for j = 1..n:
        $r^{new}(dest_j) \mathrel{+}= \beta * r^{old}(p)/n$



src	degree	destination
0	3	1, 5, 6
1	4	17, 64, 113, 117
2	2	13, 23



Analysis

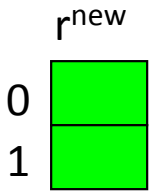
- In each iteration, we have to:
  - Read $\mathbf{r}^{old}$ and $\mathbf{M}$
  - Write $\mathbf{r}^{new}$ back to disk
  - IO Cost = $2|\mathbf{r}| + |\mathbf{M}|$
- What if we could not even fit $\mathbf{r}^{new}$ in memory?
  - 10 billion pages

Block-based Update Algorithm

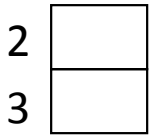
Analysis of Block Update

- Similar to nested-loop join in databases
  - Break $\mathbf{r}^{new}$ into k blocks that fit in memory
  - Scan **M** and $\mathbf{r}^{old}$ once for each block
- k scans of **M** and $\mathbf{r}^{old}$
  - $k(|\mathbf{M}| + |\mathbf{r}|) + |\mathbf{r}| = k|\mathbf{M}| + (k+1)|\mathbf{r}|$
- Can we do better?
- Hint: **M** is much bigger than **r** (approx 10-20x), so we must avoid reading it k times per iteration

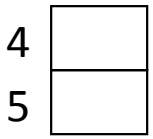
Block-Stripe Update algorithm



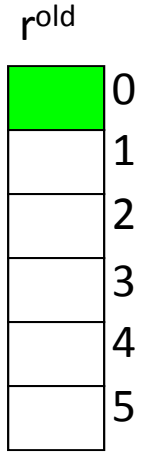
src	degree	destination
0	4	0, 1
1	2	0
2	2	1



0	4	3
2	2	3



0	4	5
1	2	5
2	2	4



Block-Stripe Analysis

- Break **M** into stripes
  - Each stripe contains only destination nodes in the corresponding block of $\mathbf{r}^{new}$
- Some additional overhead per stripe
  - But usually worth it
- Cost per iteration
  - $|\mathbf{M}|(1+\varepsilon) + (k+1)|\mathbf{r}|$

SALSA: Random Walk on Hubs and Authorities

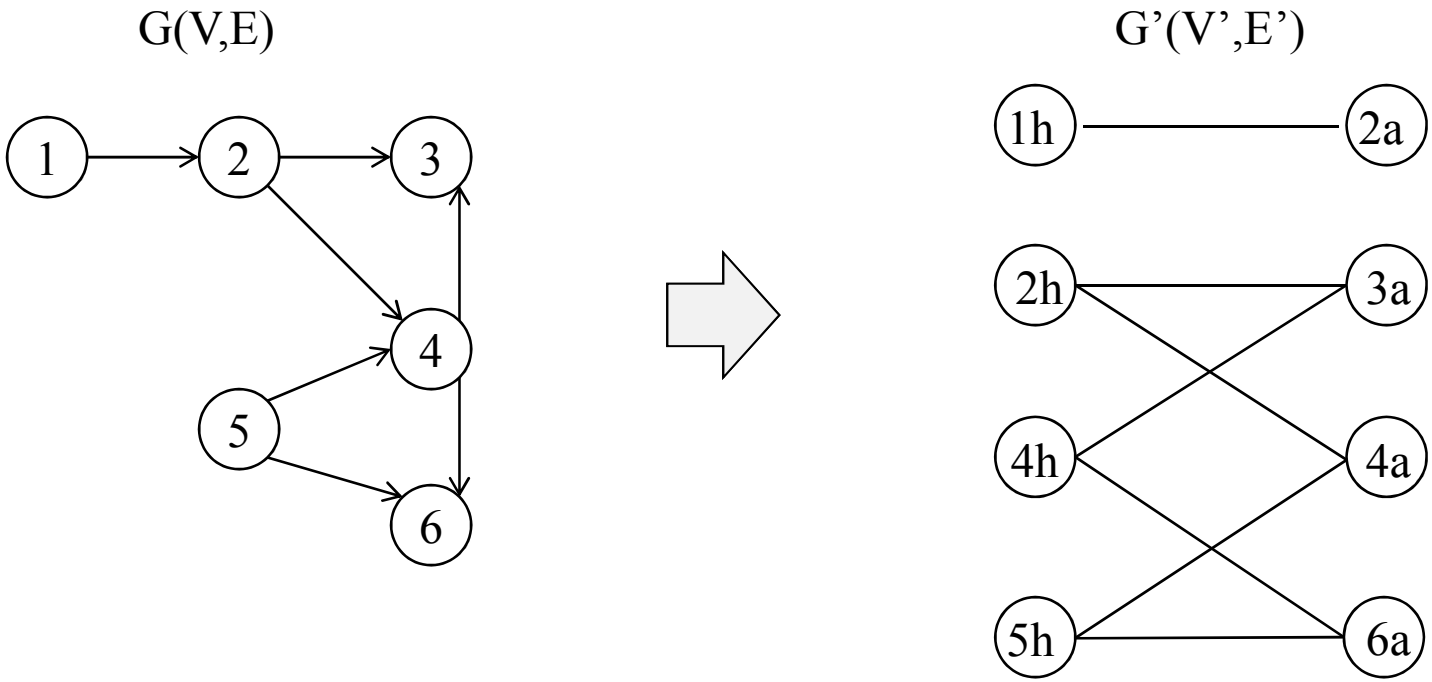
[Lempel/Moran, TOIS 2001]

View each node v of the link graph as two nodes v_h and v_a

Construct bipartite undirected graph G'(V',E') from link graph G(V,E):

$V' = \{v_h \mid v \in V \text{ and outdegree}(v)>0\} \cup \{v_a \mid v \in V \text{ and indegree}(v)>0\}$

$E' = \{(v_h, w_a) \mid (v,w) \in E\}$



SALSA: Random Walk on Hubs and Authorities

[Lempel/Moran: TOIS 2001]

→ Perform PageRank-like random walks for both Hubs and Authorities:

Stochastic hub matrix H:

$$h_{ij} = \sum_k \frac{1}{\text{degree}(i_h)} \cdot \frac{1}{\text{degree}(k_a)}$$

for i, j and k ranging over all nodes with $(i_h,k_a), (j_h,k_a) \in E'$

Stochastic authority matrix A:

$$a_{ij} = \sum_k \frac{1}{\text{degree}(i_a)} \cdot \frac{1}{\text{degree}(k_h)}$$

for i, j and k ranging over all nodes with $(k_h,i_a), (k_h,j_a) \in E'$

The corresponding Markov chains are ergodic in each connected component.

Stationary solution: $\pi[v_h] \sim$ outdegree(v) for H, $\pi[v_a] \sim$ indegree(v) for A

Further extension with random jumps: PHITS (Probabilistic HITS)

More Link Analysis Variants

Hub-Averaging:

$$a(q) = \sum_{p \in IN(q)} h(p) \quad \text{and} \quad h(p) = \frac{1}{|OUT(p)|} \sum_{q \in OUT(p)} a(q)$$

Authority-Threshold (only k best authorities per hub):

$$a(q) = \sum_{p \in IN(q)} h(p) \quad \text{and} \quad h(p) = \frac{1}{k} \sum_{q \in OUT-k(p)} a(q)$$

$$\text{with} \quad \text{OUT} - \text{k}(p) = \text{argmax} - \text{k}_q \{a(q) \mid q \in OUT(p)\}$$

Max-Authority (authority threshold with k=1):

$$a(q) = \sum_{p \in IN(q)} h(p) \quad \text{and} \quad h(p) = a(\text{argmax}_q \{a(q) \mid q \in OUT(p)\})$$

Breadth-First-Search (transitive citations up to depth k):

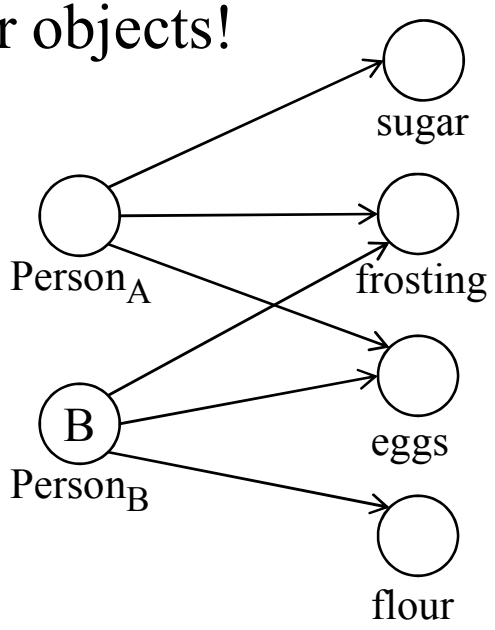
$$a(q) = \sum_{j=1}^{k} \left(\frac{1}{2}\right)^{j-1} |N^{(j)}(q)|$$

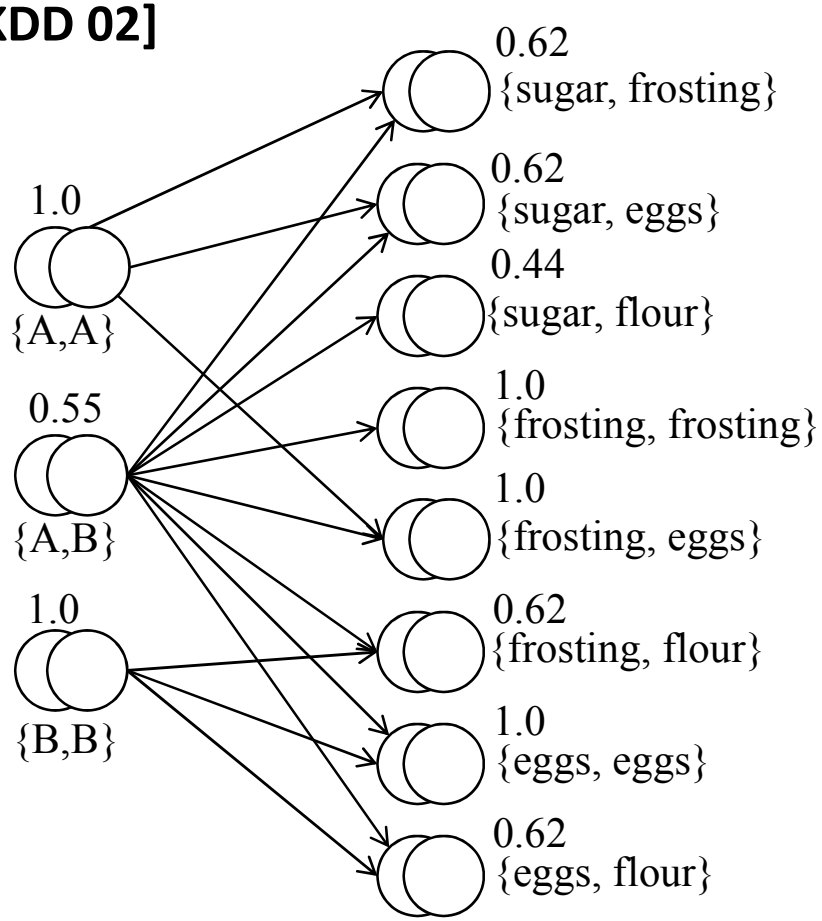
where $N^{(j)}(q)$ are nodes that have a path to q by alternating o OUT and i IN steps with j=o+i

SimRank: Measuring Structural Context Similarity

[Jeh, Widom: KDD 02]

Idea: Two objects are similar if they relate to objects that are similar objects!





$$s(A,B) = \frac{c_1}{|O(A)| \cdot |O(B)|} \sum_{i=1}^{|O(A)|} \sum_{j=1}^{|O(B)|} s(O_i(A), O_j(B))$$

$$s(c,d) = \frac{c_2}{|I(c)| \cdot |I(d)|} \sum_{i=1}^{|I(c)|} \sum_{j=1}^{|I(d)|} s(I_i(c), I_j(d))$$

$O_i(a)$ – i-th object in a's inlink neighborhood

$I_i(a)$ – i-th object in a's outlink neighborhood

and constants $c_1, c_2 \in [0,1]$

Personalized PageRank

Goal: Efficient computation and efficient storage of user-specific personalized PageRank vectors (PPR)

PageRank equation: $p = \alpha\, C\, p + (1-\alpha)\, r$

Linearity Theorem:

Let r_1 and r_2 be **personal preference vectors** for random-jump targets, and let p_1 and p_2 denote the corresponding PPR vectors.

Then for all $\beta_1, \beta_2 \geq 0$ with $\beta_1 + \beta_2 = 1$ the following holds:

$\beta_1\, p_1 + \beta_2\, p_2 = \alpha\, C\, (\beta_1\, p_1 + \beta_2\, p_2) + (1-\alpha)\, (\beta_1\, r_1 + \beta_2\, r_2)$

Corollary:

For preference vector r with m non-zero components and **base vectors** e_k (k=1..m) with $(e_k)_i = 1$ for i=k, 0 for i≠k, we obtain:

$r = \sum_{k=1..m} \beta_k\, e_k$ with constants $\beta_1 \ldots \beta_m$

and $p = \sum_{k=1..m} \beta_k\, p_k$ for PPR vector p with $p_k = \alpha\, C\, p_k + (1-\alpha)\, e_k$

[for further optimizations see Jeh/Widom: WWW 2003]