Big Data Analytics

Link Analysis

SHYI-CHYI CHENG

Outline

- Introduction
- PageRank
- Topic-Sensitive PageRank
- Link Spam
- HITS
- Summary

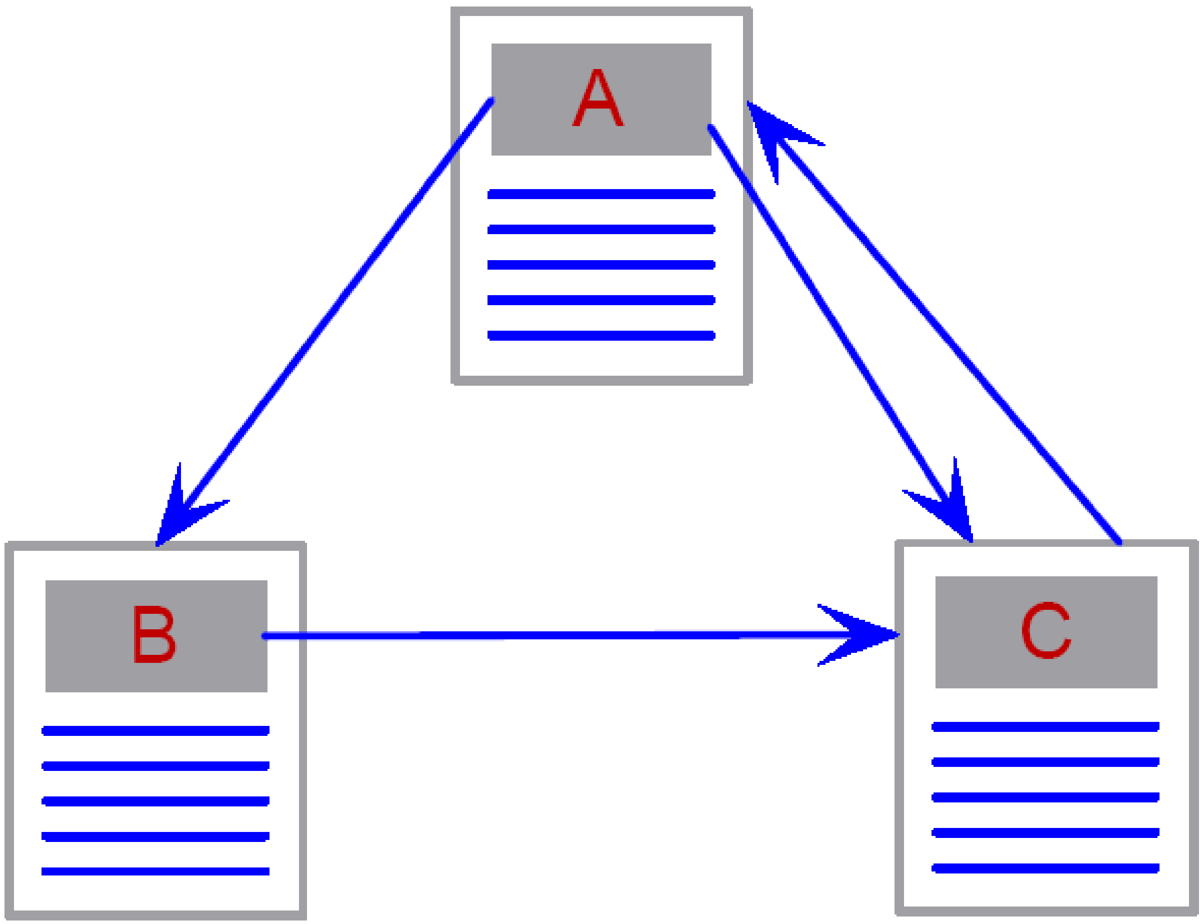
Motivation

How many articles did relate to the user query?

Why we need link analysis?

- The web is **not** just a collection of documents – its hyperlinks are important
- A link from page *A* to page *B* may indicate:
  - *A* is related to *B*, or
  - *A* is recommending, citing, voting for or endorsing *B*
- Link types
  - referential – *click here and get back home*, or
  - Informational – *click here to get more detail*
- Links effect the **Page Ranking**

Example Web Graph



Early Search Engines and Term Spam

- Early search engines mainly compare content similarity of the query and the indexed pages
  - IR techniques are applied: cosine, TF-IDF, …
- Term spam
  - Techniques for fooling search engines into believing your page is about something it is not
  - Content similarity is easily spammed

Google Solution

- PageRank was used to simulate where Web surfers
  - Pages that would have a large number of surfers were considered more "important" than pages that would rarely be visited.
- The content of a page was judged not only by the terms appearing on that page, but by the terms used in or near the links to that page.

Users of the Web "vote with their feet."

Link Analysis Strategies

- During 1997-1998, two most influential hyperlink based search algorithms PageRank and HITS were reported.
- Both algorithms are related to **social networks**. They exploit the hyperlinks of the Web to rank pages according to their levels of "prestige" or "authority".
  - **HITS**: Jon Kleinberg (Cornel University), at *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1998
  - **PageRank**: Sergey Brin and Larry Page, PhD students from Stanford University, at *Seventh International World Wide Web Conference* (WWW7) in April, 1998.

PageRank powers the Google search engine.

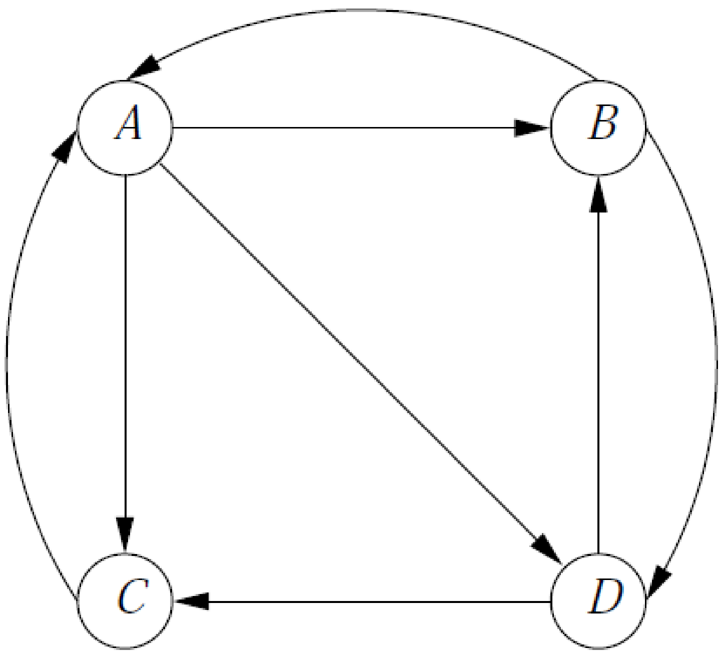
PageRank

- PageRank is a function that assigns a real number to each page in the Web
- PageRank relies on the democratic nature of the Web by using its vast link structure as an indicator of an individual page's value or quality.
- PageRank interprets a hyperlink from page x to page y as a vote, by page x, for page y.
- However, PageRank looks at more than the sheer number of votes; it also analyzes the page that casts the vote.
  - Votes casted by “important” pages weigh more heavily and help to make other pages more "important."

More specifically

- A hyperlink from a page to another page is an implicit conveyance of authority to the target page.
  - The more in-links that a page *i* receives, the more prestige the page *i* has.
- Pages that point to page *i* also have their own prestige scores.
  - A page of a higher prestige pointing to *i* is more important than a page of a lower prestige pointing to *i*.
  - In other words, a page is important if it is pointed to by other important pages.

Web as a Directed Graph



$$
\begin{array}{c}
 \\ A \\ B \\ C \\ D
\end{array}
\begin{array}{c}
\begin{array}{cccc} A & B & C & D \end{array} \\
\begin{bmatrix}
0 & 1/2 & 1 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 1/2 & 0 & 0
\end{bmatrix}
\end{array}
$$

Transition Matrix M

Probabilities of all Nodes v

- Suppose we start a random surfer at any of the n pages of the Web with equal probability
- Initially, Probability vector v = [1/n, …, 1/n]
- Loop

    $v = Mv$

  until end the surfing

Probability Estimation by Markov Process

- Conditions
  - The graph is strongly connected; that is, it is possible to get from any node to any other node.
  - There are no dead ends: nodes that have no arcs out.

What is a Markov Chain?

- A Markov chain has two components:

1) A network structure much like a web site, where each node is called a state.

2) A transition probability of traversing a link given that the chain is in a state.
    - For each state the sum of outgoing probabilities is one.

- A sequence of steps through the chain is called a *random walk*.

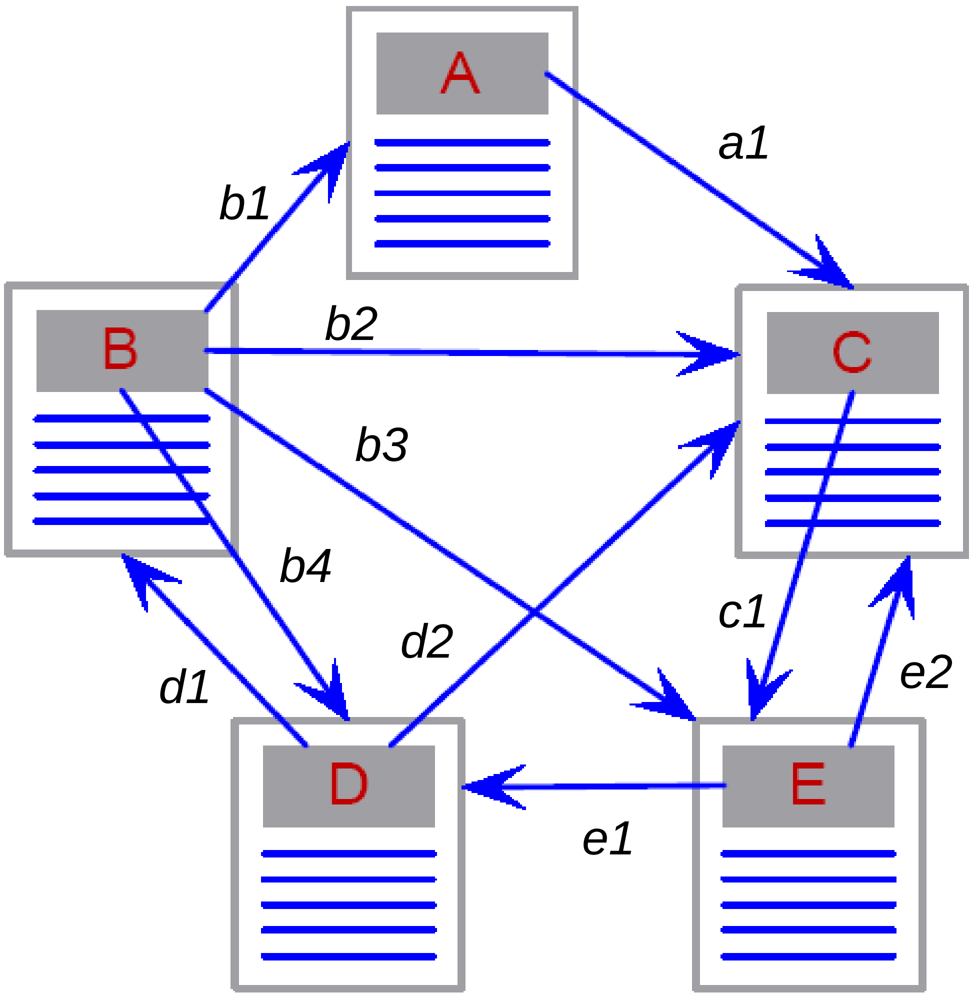
Markov chains

- Markov chains have been extensively studied by statisticians and have been applied in a wide variety of areas.

Markov Property:

$$P(S_t \mid S_{t-1}, S_{t-2}, \ldots, S_2, S_1) = P(S_t \mid S_{t-1}) \tag{3.3}$$

Markov Chain Example



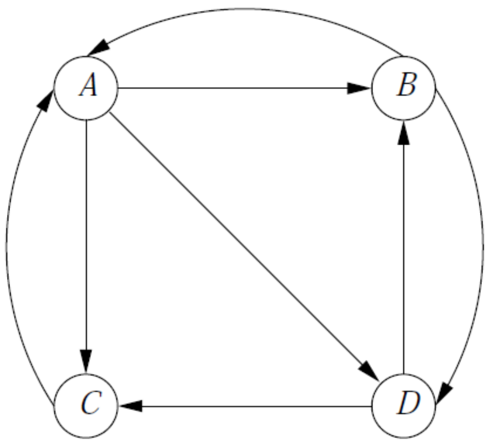
The Random Surfer

- Assume the web is a Markov chain.
- Surfers randomly click on links, where the probability of an outlink from page A is *1/m*, where *m* is the number of outlinks from *A*.
- The surfer occasionally gets *bored* and is *teleported* to another web page, say *B*, where *B* is equally likely to be any page.
- Using the theory of Markov chains it can be shown that if the surfer follows links for long enough, *the PageRank of a web page is the probability that the surfer will visit that page*.

Principal Eigenvector of M

- The limit is reached when multiplying the distribution by M another time does not change the distribution.
- The limiting v is an eigenvector of M
- We can compute the principal eigenvector of M by starting with the initial vector v0 and multiplying by M some number of times, until the vector we get shows little change at each round.

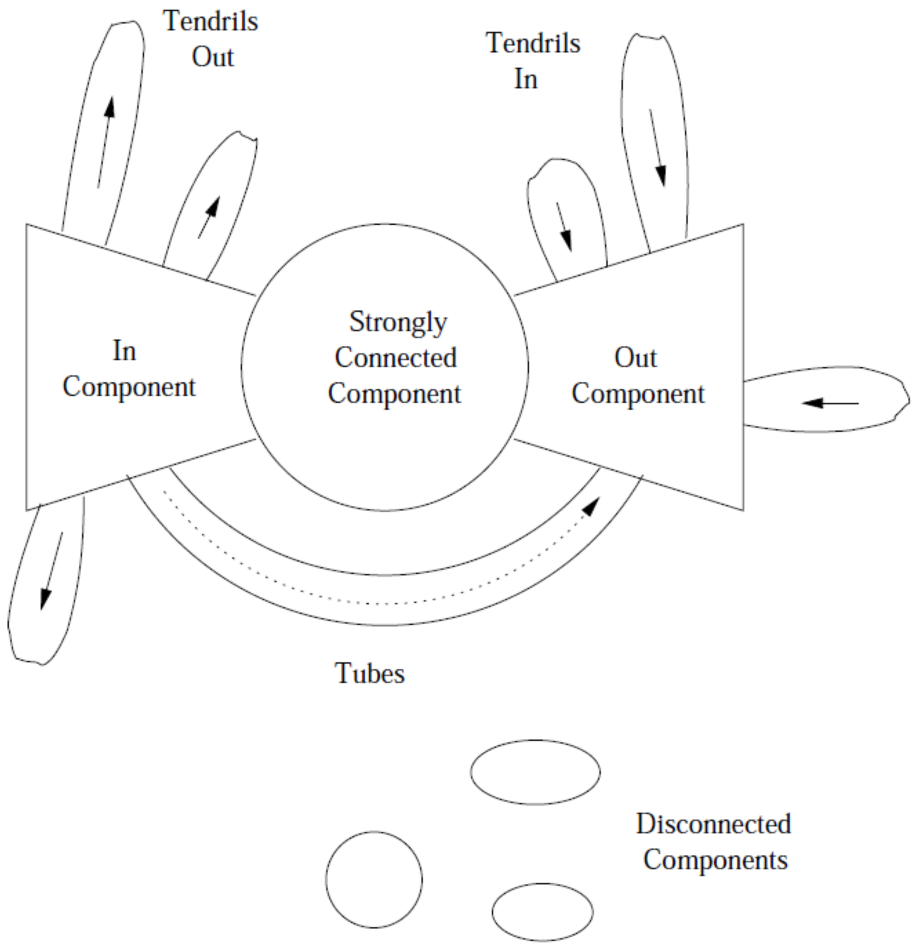
Example



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

$$\underset{v}{\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}} \quad \underset{Mv}{\begin{bmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix}} \quad \underset{M^2v}{\begin{bmatrix} 15/48 \\ 11/48 \\ 11/48 \\ 11/48 \end{bmatrix}} \quad \underset{M^3v}{\begin{bmatrix} 11/32 \\ 7/32 \\ 7/32 \\ 7/32 \end{bmatrix}} \quad \cdots \quad \underset{M^{56}v}{\begin{bmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{bmatrix}}$$

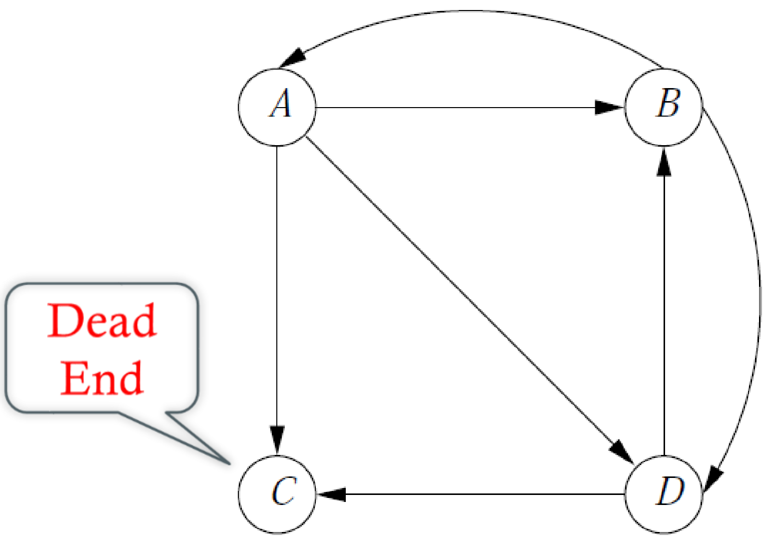
Structure of the Web



To use probability as measuring the importance of a page falsely concludes that nothing in the SCC or in-component is of any importance.

Avoiding Dead Ends

- A page with no link out is called a dead end

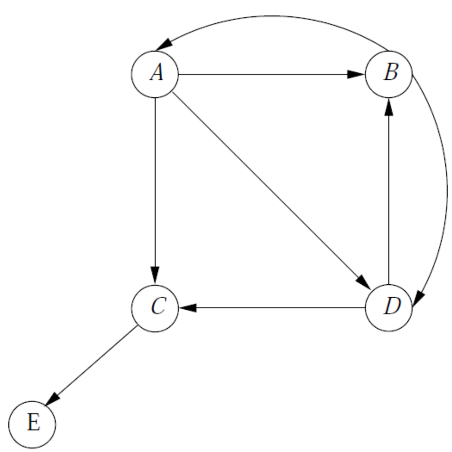


$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

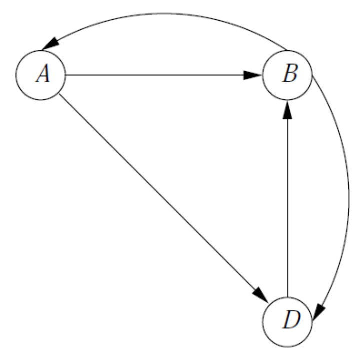
$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \begin{bmatrix} 3/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix} \begin{bmatrix} 5/48 \\ 7/48 \\ 7/48 \\ 7/48 \end{bmatrix} \begin{bmatrix} 21/288 \\ 31/288 \\ 31/288 \\ 31/288 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Two approaches to dealing with dead ends

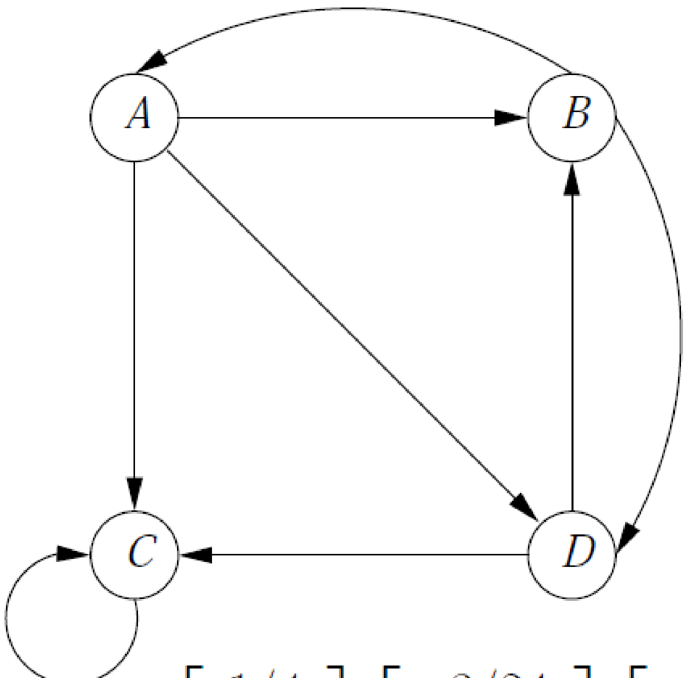
- We can drop the dead ends from the graph, and also drop their incoming arcs, recursively.
- We can modify the process by which random surfers are assumed to move about the Web.
  - The taxation method



Drop E and C



Spider Trap



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

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \begin{bmatrix} 3/24 \\ 5/24 \\ 11/24 \\ 5/24 \end{bmatrix} \begin{bmatrix} 5/48 \\ 7/48 \\ 29/48 \\ 7/48 \end{bmatrix} \begin{bmatrix} 21/288 \\ 31/288 \\ 205/288 \\ 31/288 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Revised Probability Estimation

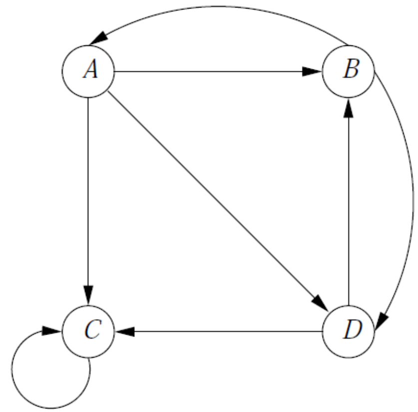
- The iterative step, where we compute a new vector estimate of PageRanks v′ from the current PageRank estimate v and the transition matrix M is

$$\mathbf{v}' = \beta M \mathbf{v} + (1-\beta)\mathbf{e}/n$$

where β is a chosen constant, usually in the range 0.8 to 0.9, **e** is a vector of all 1's with the appropriate number of components, and *n* is the number of nodes in the Web graph.

e *biases* the jump prefer some pages over others:
--e.g. **e** has 1 for your home page and 0 otherwise.
--e.g. **e** prefers the topics you are interested in.

Example



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

$\beta = 0.8$

$$\mathbf{v}' = \begin{bmatrix} 0 & 2/5 & 0 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 4/5 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v} + \begin{bmatrix} 1/20 \\ 1/20 \\ 1/20 \\ 1/20 \end{bmatrix}$$

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \begin{bmatrix} 9/60 \\ 13/60 \\ 25/60 \\ 13/60 \end{bmatrix} \begin{bmatrix} 41/300 \\ 53/300 \\ 153/300 \\ 53/300 \end{bmatrix} \begin{bmatrix} 543/4500 \\ 707/4500 \\ 2543/4500 \\ 707/4500 \end{bmatrix} \cdots \begin{bmatrix} 15/148 \\ 19/148 \\ 95/148 \\ 19/148 \end{bmatrix}$$

Issues of Computation of PageRank

- The transition matrix of the Web M is very sparse
- How to use MapReduce for parallel computing?

Change Matrix Representation

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

Source	Degree	Destinations
A	3	B, C, D
B	2	A, D
C	1	A
D	2	B, C

PageRank Iteration Using MapReduce

- Partitioning the matrix M into k^2 blocks, while the vectors are still partitioned into k stripes.
  - Need k^2 Map tasks. Each task gets one square of the matrix M, say M_{ij}, and one stripe of the vector v, v_j.
  - v is transmitted over the network k times However, each piece of the matrix is sent only once.

$$\begin{bmatrix} \mathbf{v}_1' \\ \mathbf{v}_2' \\ \mathbf{v}_3' \\ \mathbf{v}_4' \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \end{bmatrix}$$

Topic-Sensitive PageRank

Motivation

- PageRank scores are independent of the queries
- Can we bias PageRank rankings to take into account query keywords?

Topic-sensitive PageRank

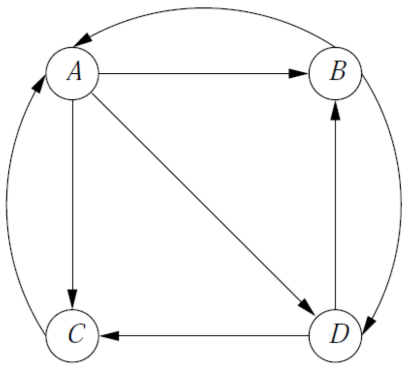
To create one vector for each of some small number of topics, biasing the PageRank to favor pages of that topic

Biased Random Walks

- Suppose S is the teleport set
  - a set of integers consisting of the row/column numbers for the pages we have identified as belonging to a certain topic
- Let $\mathbf{e}_S$ be a vector that has 1 in the components in S and 0 in other components.

$$\mathbf{v}' = \beta M \mathbf{v} + (1 - \beta)\mathbf{e}_S / |S|$$

Example



$$\beta M = \begin{bmatrix} 0 & 2/5 & 4/5 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix}$$

β = 0.8

$$\mathbf{v}' = \begin{bmatrix} 0 & 2/5 & 4/5 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v} + \begin{bmatrix} 0 \\ 1/10 \\ 0 \\ 1/10 \end{bmatrix}$$

S={B,D}

$$\begin{bmatrix} 0/2 \\ 1/2 \\ 0/2 \\ 1/2 \end{bmatrix} \begin{bmatrix} 2/10 \\ 3/10 \\ 2/10 \\ 3/10 \end{bmatrix} \begin{bmatrix} 42/150 \\ 41/150 \\ 26/150 \\ 41/150 \end{bmatrix} \begin{bmatrix} 62/250 \\ 71/250 \\ 46/250 \\ 71/250 \end{bmatrix} \cdots \begin{bmatrix} 54/210 \\ 59/210 \\ 38/210 \\ 59/210 \end{bmatrix}$$

Using Topic-Sensitive PageRank

- To integrate topic-sensitive PageRank into a search engine
  - Decide on the topics for which we shall create specialized PageRank vectors.
  - Pick a teleport set for each of these topics, and use that set to compute the topic-sensitive PageRank vector for that topic.
  - Find a way of determining the topic or set of topics that are most relevant for a particular search query.
  - Use the PageRank vectors for that topic or topics in the ordering of the responses to the search query.

The Largest Matrix Computation in the World

- Computing PageRank can be done via matrix multiplication, where the matrix has 3 billion rows and columns.
- The matrix is sparse as average number of outlinks is between 7 and 8.
- Setting $\beta = 0.15$ or above requires at most 100 iterations to convergence.
- Researchers still trying to speed-up the computation.

Monte Carlo Methods in Computing PageRank

- Rather than following a single long random walk, the random surfer can follow many sampled random walks.
- Each walk starts at a random page and either teleports with probability T or continues choosing a link with uniform probability.
- The PR of a page is the proportion of times a sample random walk ended at that page.
- Rather than starting at a random page, start each walk a fixed number of times from each page.

Inlinks and Outlinks

- The number of incoming links to a web page is correlated to the PageRank, but … this measure is a noisy metric!
- PageRank is biased against new pages (*Googlearch*y), why?
- The long tail of queries, gives less popular web pages some visibililty.
- A single outlink to one in the community minimises the loss of PageRank within a community.

Discussion

- Inferring topics from words have been extensively studied for decades
- We can build topic classifiers in advance to guess the topics that are interested

Link Spam

Motivation

- PageRank solves the term spam problem
- Spammers turned to use the boosting techniques to fool the PageRank algorithm into overvaluing certain pages

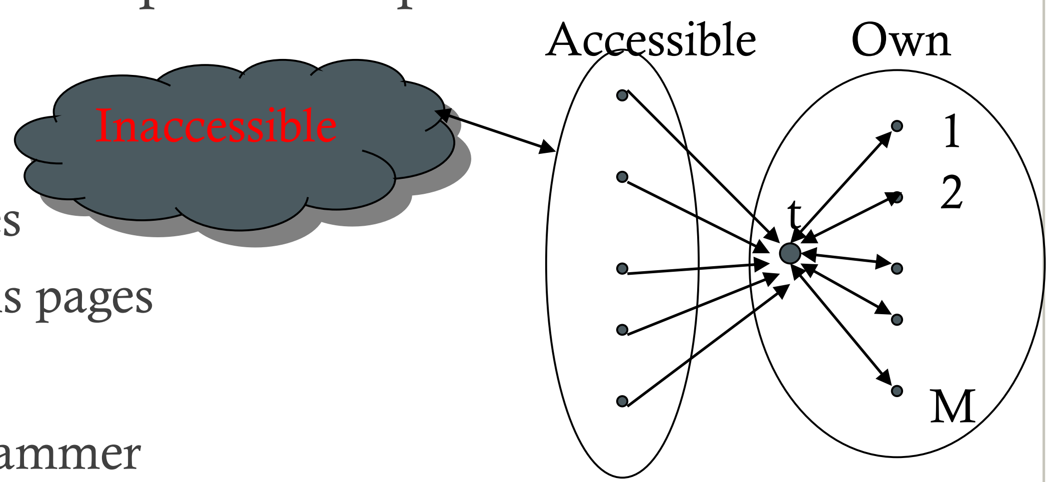
Web Spam Taxonomy

- Defined 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

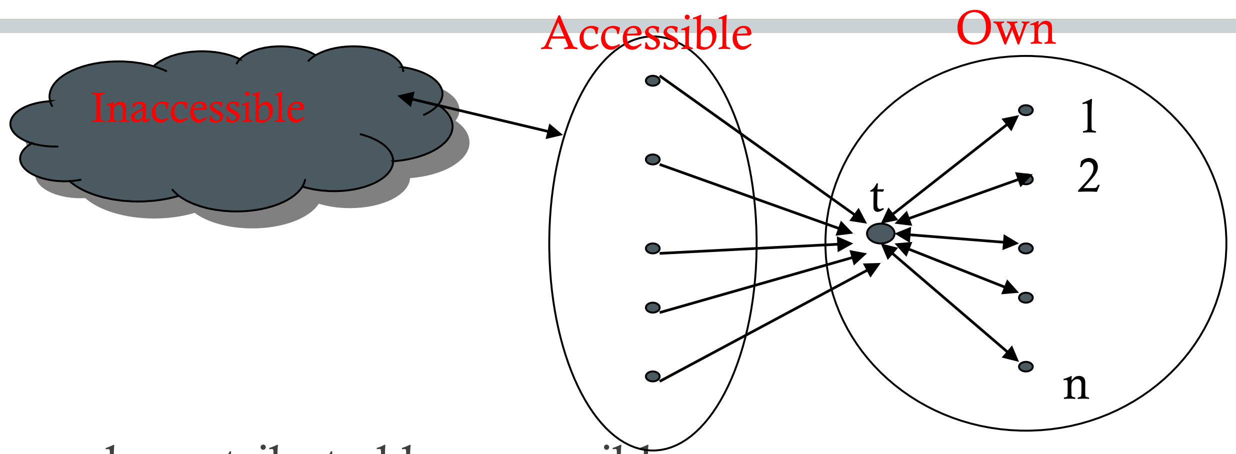
Link Spam by Boosting

- Creating link structures that boost page rank or hubs and authorities scores

- 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



Spam Farm Analysis



Suppose rank contributed by accessible pages = x

Let page rank of target page = y

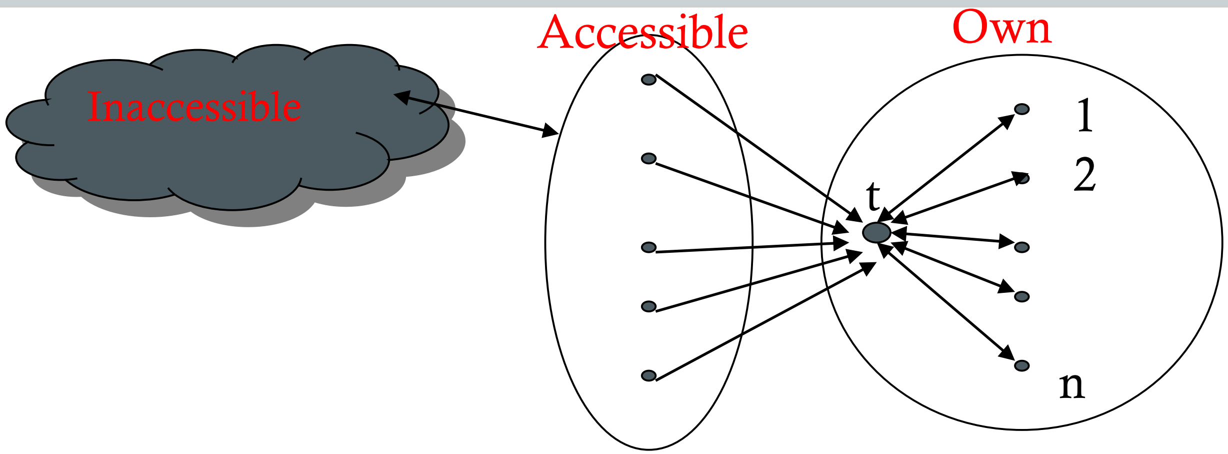
Rank of each "farm" page = $\beta y/m + (1-\beta)/n$

$y = x + \beta m[\beta y/m + (1-\beta)/n] + (1-\beta)/n$

$\quad = x + \beta^2 y + \beta(1-\beta)m/n + (1-\beta)/n$

$y = x/(1-\beta^2) + cm/n \quad$ where $c = \beta/(1+\beta)$

Example



- $y = x/(1-\beta^2) + cm/n$ where $c = \beta/(1+\beta)$
- For $\beta = 0.85$, $1/(1-\beta^2) = 3.6$
  - Multiplier effect for "acquired" page rank
  - By making m large, we can make y as large as we want

Hiding techniques

- Content hiding
  - Use same color for text and page background
- Cloaking
  - Return different pages to crawlers and browsers
- Redirection
  - Alternative to cloaking
  - Redirects are followed by browsers but not crawlers

Combating Link Spam

- Open research area
- Two approaches
  - TrustRank
    - a variation of topic-sensitive PageRank designed to lower the score of spam pages.
  - Spam mass
    - a calculation that identifies the pages that are likely to be spam and allows the search engine to eliminate those pages or to lower their PageRank strongly.

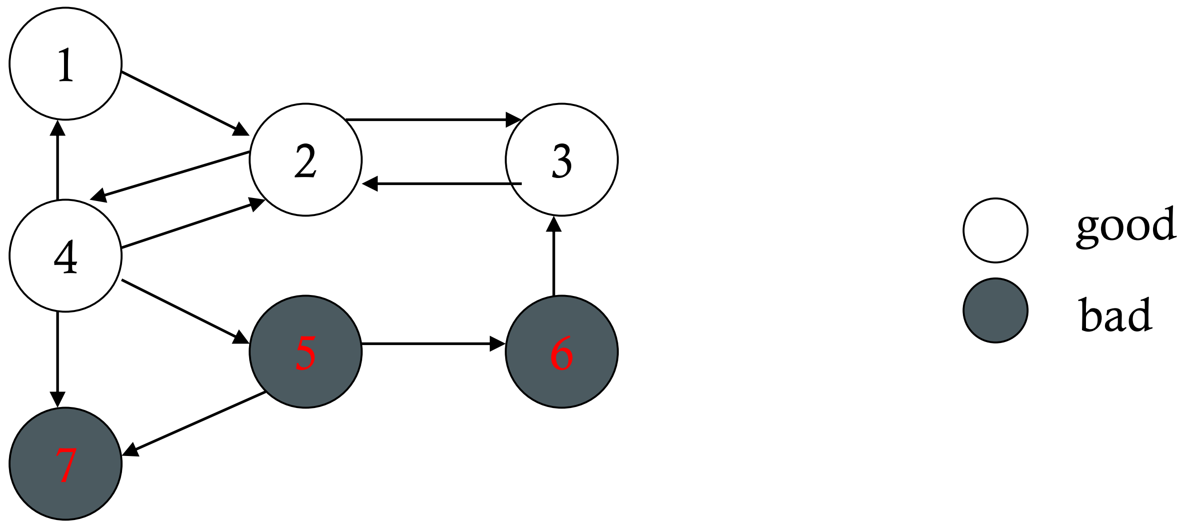
TrustRank idea

- 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

Example



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 Page Rank
  - Within a scaling factor, trust rank = biased page rank 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 page rank
  - Assume high page rank pages are close to other highly ranked pages
  - We care more about high page rank "good" pages

Inverse page rank

- 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 page rank”
  - Construct graph G’ by reversing each edge in web graph G
  - Page Rank in G’ is inverse page rank in G
- Pick top k pages by inverse page rank

Spam Mass

- To measure each page the fraction of its PageRank that comes from spam
- Suppose page p has PageRank r and TrustRank t
  - The spam mass of p is (r–t)/r
- Example

Spam

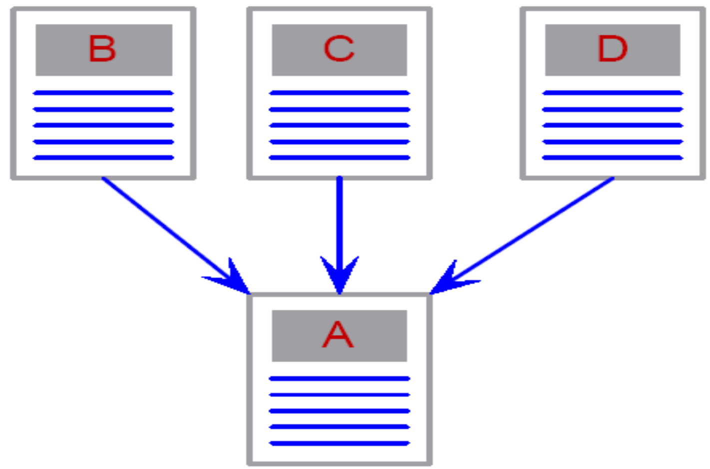
Node	PageRank	TrustRank	Spam Mass
A	$3/9$	$54/210$	0.229
B	$2/9$	$59/210$	-0.264
C	$2/9$	$38/210$	0.186
D	$2/9$	$59/210$	-0.264

Hubs and Authorities

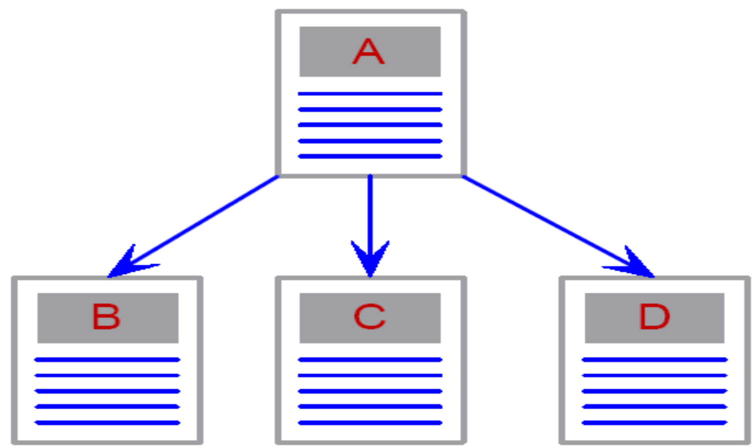
HITS (hyperlink-induced topic search)

Authority vs. Hub

- **A** is an **authority**



- **A** is a **hub**



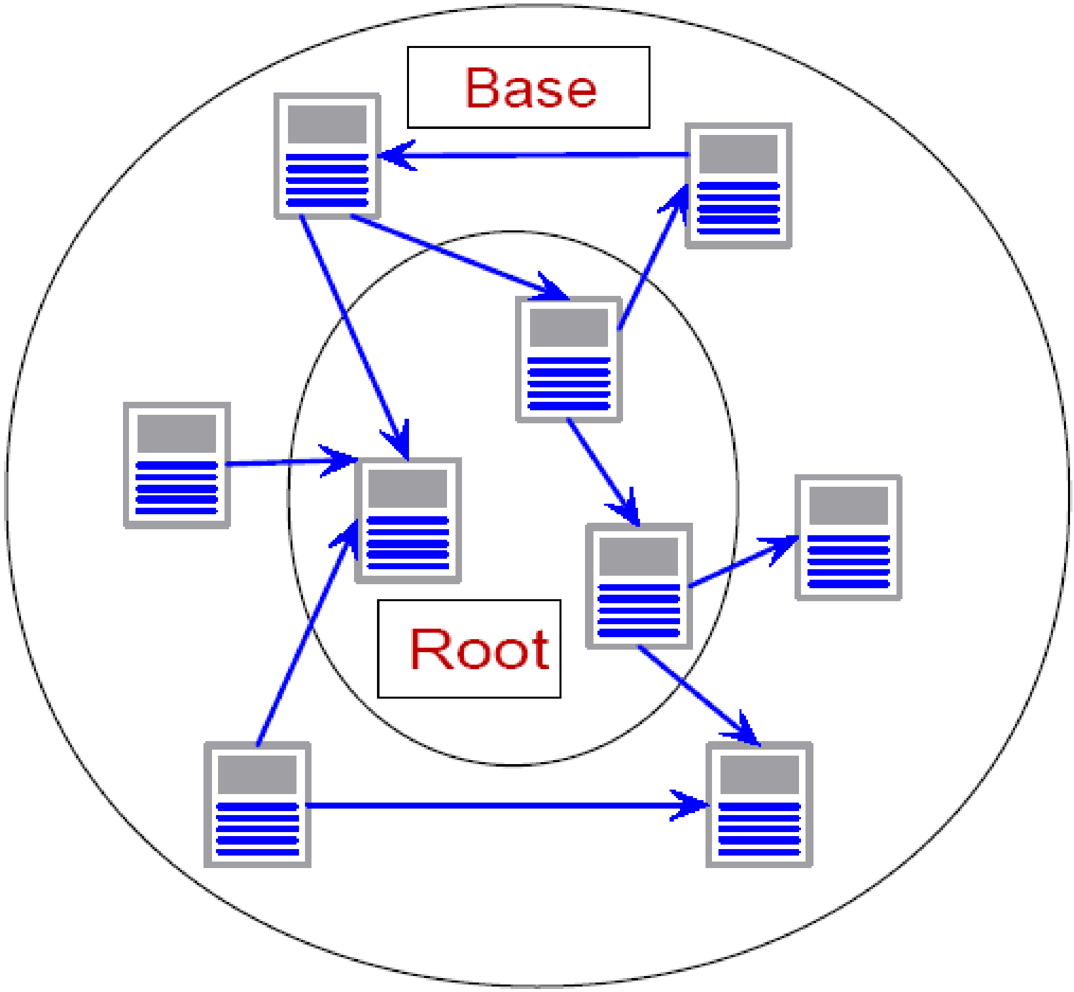
The Intuition Behind HITS

- HITS views important pages as having two flavors of importance
- Authority pages are valuable because they provide information about a topic.
- Hub pages are valuable because they tell you where to go to find out about that topic.

Pre-processing for HITS

1) Collect the top *t* pages (say t = 200) based on the input query; call this the **root set**.
2) Extend the root set into a **base set** as follows, for all pages *p* in the root set:
   1) add to the root set all pages that *p* points to, and
   2) add to the root set up-to *q* pages that point to *p* (say *q* = 50).
3) Delete all links within the same web site in the base set resulting in a **focused sub-graph**.

Expanding the Root Set



Formalizing Hubbiness and Authority

- Vectors **h** and **a** measure the hubbiness and authority of pages, respectively

- Iterate until Convergence

$$\mathrm{h} = \lambda L \mathrm{a}$$

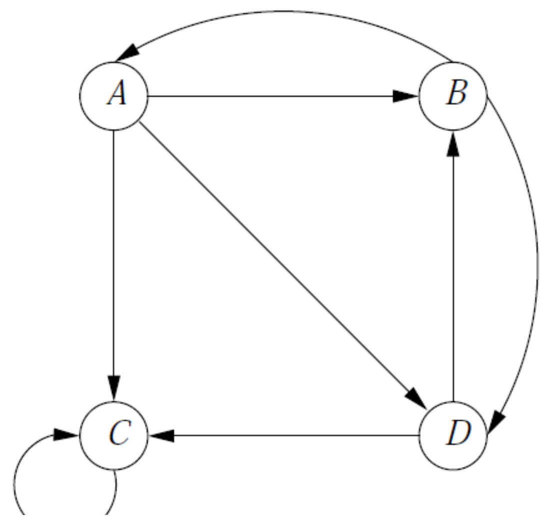
$$\mathrm{a} = \mu L^{T} \mathrm{h}$$

$$L_{ij} = \begin{cases} 1 & if\,(i,j) \in E \\ 0 & otherwise \end{cases}$$

  - λ and μ are unknown constant representing the scaling factors needed

  - *L is* the link matrix of the Web *E*

Example



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

Example

First two iterations

$$
\underset{\mathbf{h}}{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}} \quad
\underset{L^{\mathrm{T}}\mathbf{h}}{\begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ 1 \end{bmatrix}} \quad
\underset{\mathbf{a}}{\begin{bmatrix} 1/2 \\ 1 \\ 1 \\ 1 \\ 1/2 \end{bmatrix}} \quad
\underset{L\mathbf{a}}{\begin{bmatrix} 3 \\ 3/2 \\ 1/2 \\ 2 \\ 0 \end{bmatrix}} \quad
\underset{\mathbf{h}}{\begin{bmatrix} 1 \\ 1/2 \\ 1/6 \\ 2/3 \\ 0 \end{bmatrix}}
$$

$$
\underset{L^{\mathrm{T}}\mathbf{h}}{\begin{bmatrix} 1/2 \\ 5/3 \\ 5/3 \\ 3/2 \\ 1/6 \end{bmatrix}} \quad
\underset{\mathbf{a}}{\begin{bmatrix} 3/10 \\ 1 \\ 1 \\ 9/10 \\ 1/10 \end{bmatrix}} \quad
\underset{L\mathbf{a}}{\begin{bmatrix} 29/10 \\ 6/5 \\ 1/10 \\ 2 \\ 0 \end{bmatrix}} \quad
\underset{\mathbf{h}}{\begin{bmatrix} 1 \\ 12/29 \\ 1/29 \\ 20/29 \\ 0 \end{bmatrix}}
$$

Example

The limits are:

$$\mathbf{h} = \begin{bmatrix} 1 \\ 0.3583 \\ 0 \\ 0.7165 \\ 0 \end{bmatrix} \qquad \mathbf{a} = \begin{bmatrix} 0.2087 \\ 1 \\ 1 \\ 0.7913 \\ 0 \end{bmatrix}$$

Computation of HITS

- The computation of authority scores and hub scores is the same as the computation of the PageRank scores, using power iteration.
- If we use $\mathbf{a}_k$ and $\mathbf{h}_k$ to denote authority and hub vectors at the kth iteration, the iterations for generating the final solutions are

$$\mathrm{a}_k = L^T L \mathrm{a}_{k-1}, \mathrm{a}_0 = [1,1,\ldots,1]$$

$$\mathrm{h}_k = L^T L \mathrm{h}_{k-1}, \mathrm{h}_0 = [1,1,\ldots,1]$$

The HITS algorithm

HITS-Iterate(G)

$\boldsymbol{a}_0 = \boldsymbol{h}_0 = (1, 1, \ldots, 1)$;

$k = 1$

Repeat

$\boldsymbol{a}_k = \boldsymbol{L}^T \boldsymbol{L}\boldsymbol{a}_{k-1}$;

$\boldsymbol{h}_k = \boldsymbol{L}\boldsymbol{L}^T \boldsymbol{h}_{k-1}$;

normalize $\boldsymbol{a}_k$;

normalize $\boldsymbol{h}_k$;

$k = k + 1$;

until $\boldsymbol{a}_k$ and $\boldsymbol{h}_k$ do not change significantly;

return $\boldsymbol{a}_k$ and $\boldsymbol{h}_k$

Strengths and weaknesses of HITS

- Strength: its ability to rank pages according to the query topic, which may be able to provide more relevant authority and hub pages.

- Weaknesses:
  - It is easily spammed. It is in fact quite easy to influence HITS since adding out-links in one's own page is so easy.
  - Topic drift. Many pages in the expanded set may not be on topic.
  - Inefficiency at query time: The query time evaluation is slow. Collecting the root set, expanding it and performing eigenvector computation are all expensive operations

Applications of HITS

- Search engine querying (speed an issue)
- Finding web communities.
- Finding related pages.
- Populating categories in web directories.
- Citation analysis.

Summary

- In this chapter, we introduced
  - Term Spam
  - The Google Solution to Term Spam
  - PageRank
  - Transition Matrix of the Web
  - Computing PageRank on Strongly Connected Web Graphs
  - The Random Surfer Model
  - Dead Ends

Summary

- In this chapter, we introduced
  - Spider Traps
  - Taxation Schemes
  - Taxation and Random Surfers
  - Efficient Representation of Transition Matrices
  - Topic-Sensitive PageRank
  - Link Spam
  - Hubs and Authorities
  - Recursive Formulation of the HITS Algorithm

Homework 6:Page Rank Implementation with Spark

報告要準時交

Why PageRank?

- Good example of a more complex algorithm
  - Multiple stages of map & reduce
- Benefits from Spark's in-memory caching
  - Multiple iterations over the same data

Basic Idea

- Give pages ranks (scores) based on links to them
  - Links from many pages ➔ high rank
  - Link from a high-rank page ➔ high rank

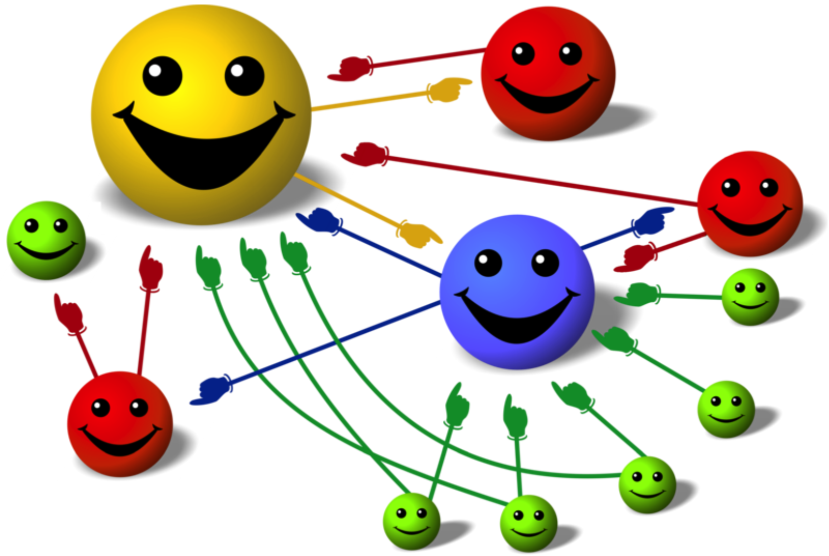
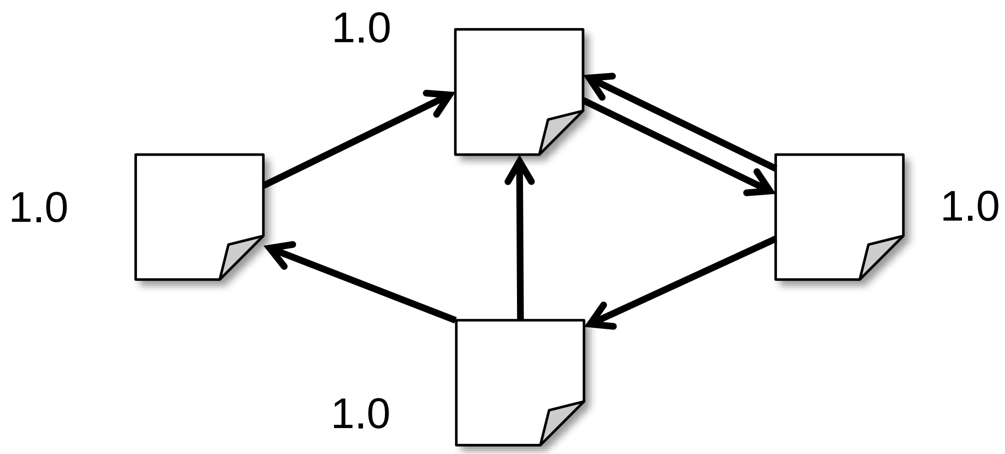


Image: en.wikipedia.org/wiki/File:PageRank-hi-res-2.png

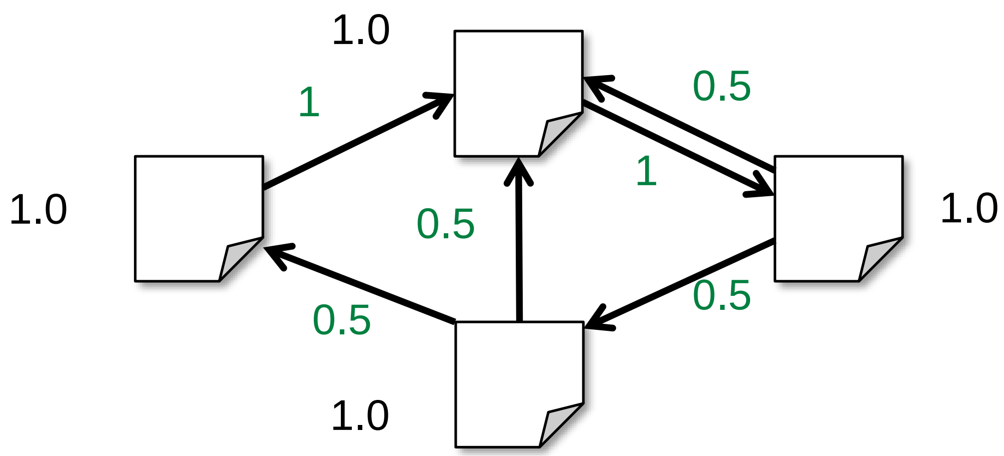
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to $0.15 + 0.85 \times$ contribs



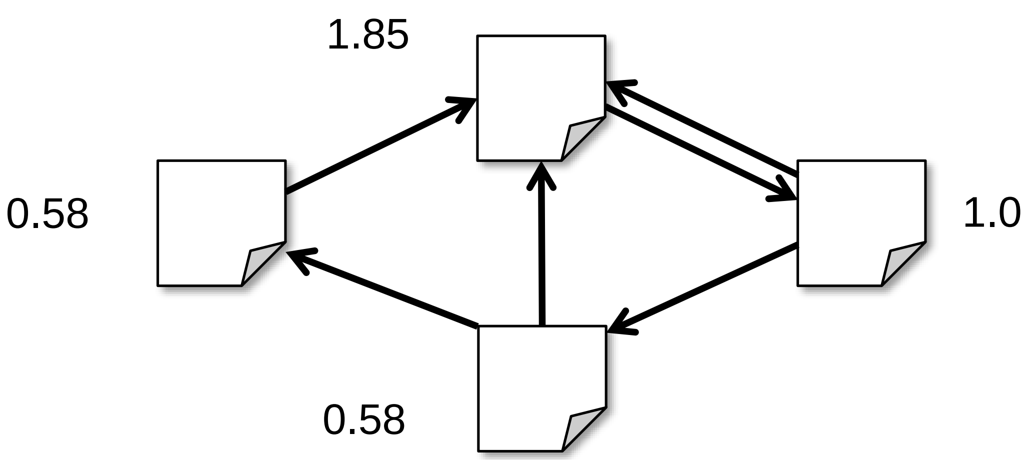
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to $0.15 + 0.85 \times$ contribs



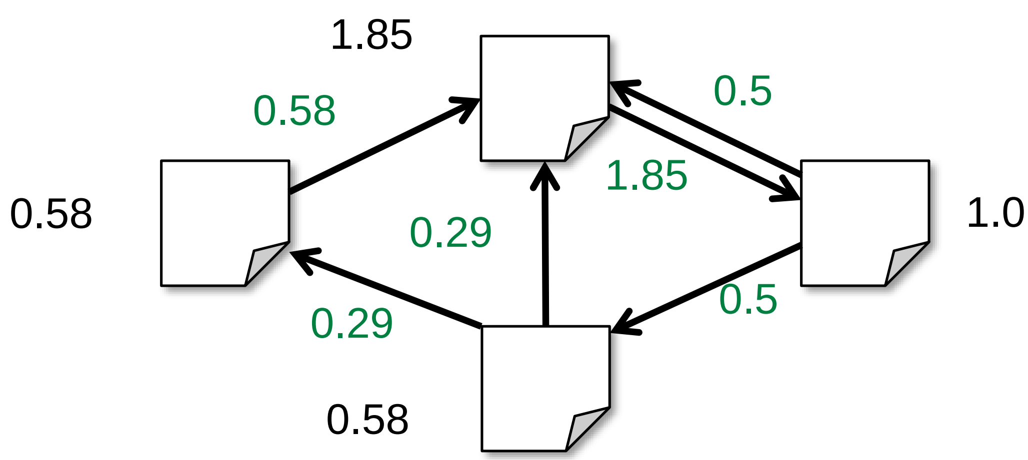
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to $0.15 + 0.85 \times$ contribs



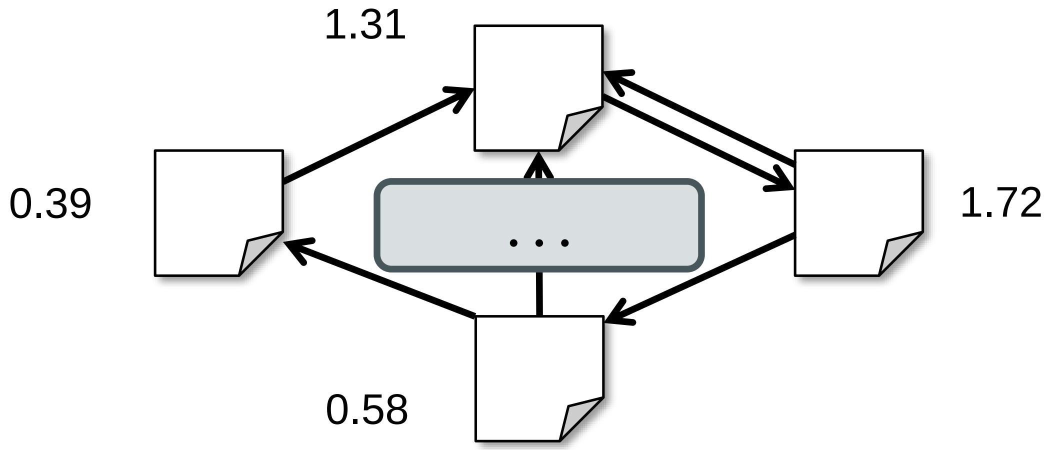
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute rank_p / $|\text{neighbors}_p|$ to its neighbors
3. Set each page's rank to $0.15 + 0.85 \times \text{contribs}$



Algorithm

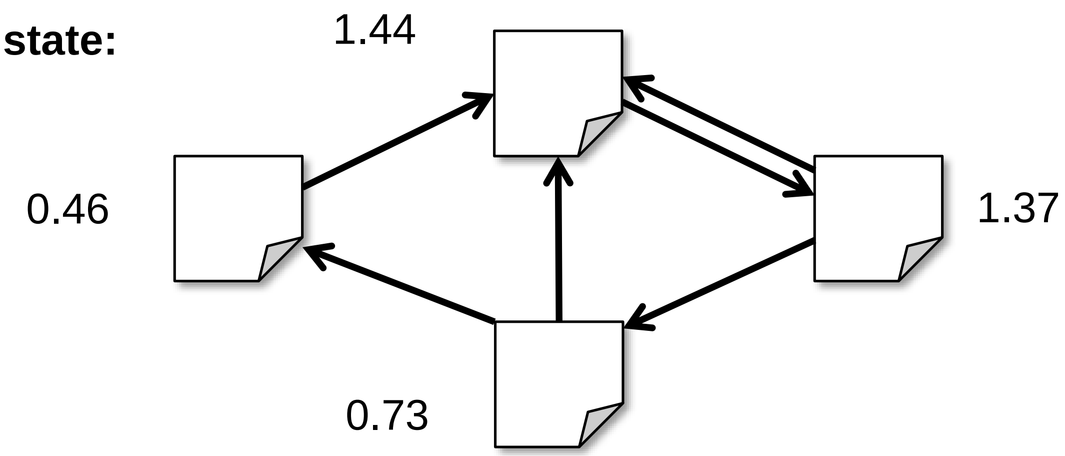
1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to $0.15 + 0.85 \times$ contribs



Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to $0.15 + 0.85 \times$ contribs

Final state:



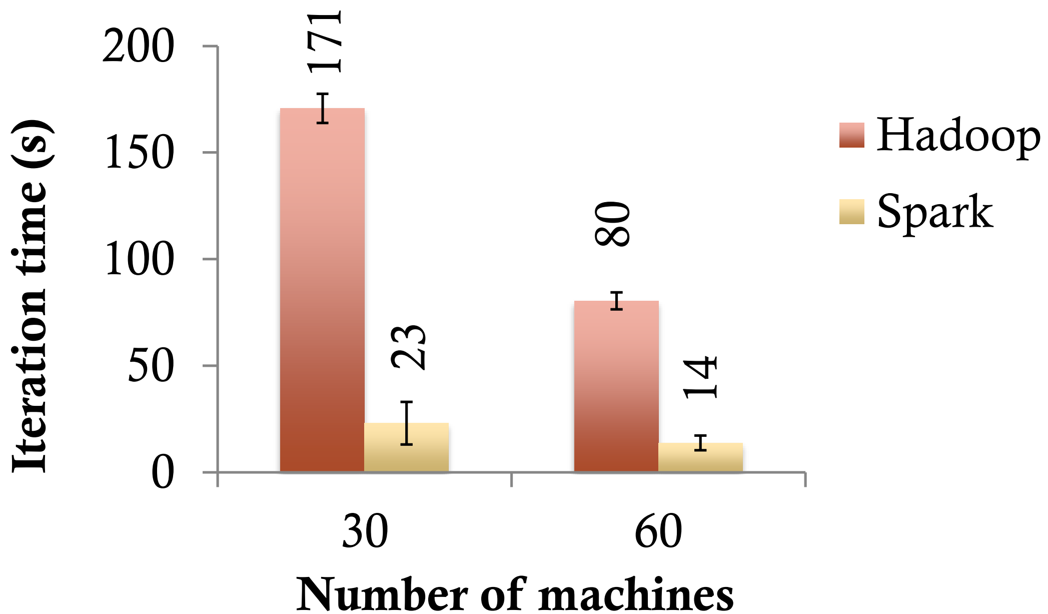
Scala Implementation

```
val links = // RDD of (url, neighbors) pairs
var ranks = // RDD of (url, rank) pairs

for (i <- 1 to ITERATIONS) {
  val contribs = links.join(ranks).flatMap {
    case (url, (links, rank)) =>
      links.map(dest => (dest, rank/links.size))
  }
  ranks = contribs.reduceByKey(_ + _)
                  .mapValues(0.15 + 0.85 * _)
}

ranks.saveAsTextFile(...)
```

PageRank Performance



Any Questions?