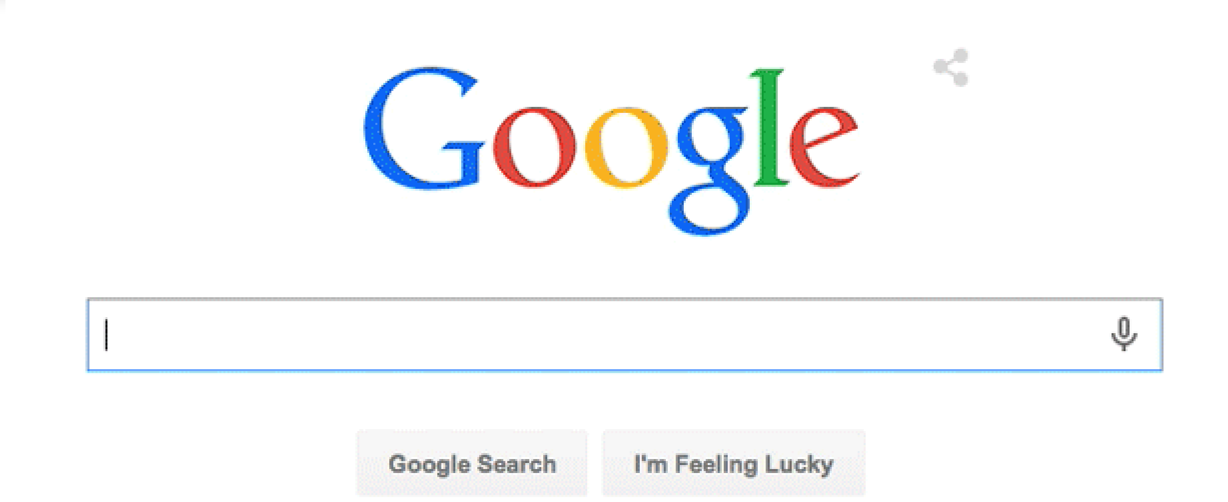
Sergey Brin and Lawrence Page:
The Anatomy of a Large-Scale Hypertextual Web Search Engine (1998)

Приложение марковских цепей: PageRank



thanks

Ryan Tibshirani

Associate Professor Department of Statistics and

Machine Learning Department Carnegie Mellon University

(in Pittsburgh, Pennsylvania**)**

Rodert Tibshirani

Professor in the Department of Statistics and Health Research and Policy at Stanford University

His most well-known contributions are the LASSO method

Recent News

- There are some news about that PageRank will be canceled by Google.
- There are large numbers of Search Engine Optimization (SEO).
- SEO use different trick methods to make a web page more important under the rating of PageRank.

PageRank algorithm

- Given webpages numbered 1,...n. The PageRank of webpage i is based on its linking webpages (webpages j that link to i), but we don't just count the number of linking webpages, i.e., don't want to treat all linking webpages equally

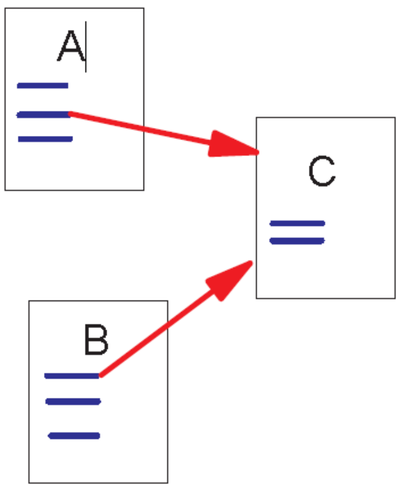
- Instead, we weight the links from different webpages:

Webpages that link to i, and have high PageRank scores themselves, should be given more weight

Webpages that link to i, but link to a lot of other webpages in general, should be given less weight

Link Structure of the Web

- 150 million web pages → 1.7 billion links

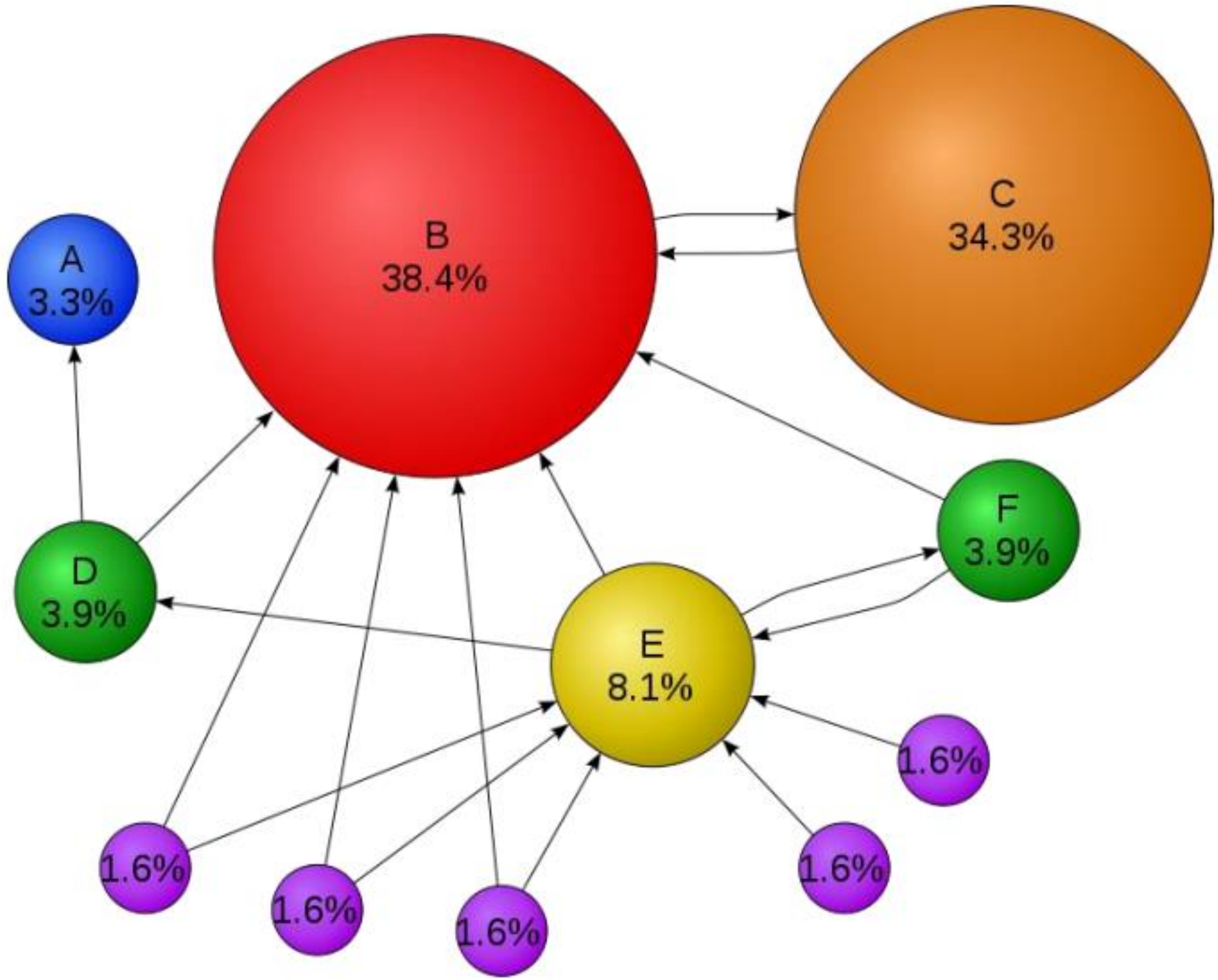


Backlinks and Forward links:
- A and B are C's backlinks
- C is A and B's forward link

Intuitively, a webpage is important if it has a lot of backlinks.

What if a webpage has only one link off www.yahoo.com?

PageRanks for a simple network



BrokenRank (almost PageRank) definition

Let $L_{ij} = 1$ if webpage j links to webpage i (written $j \rightarrow i$),

and $L_{ij} = 0$ otherwise

Also let $m_j = \sum_{k=1}^{n} L_{kj}$, the total number of webpages that j links to

First we define something that's almost PageRank, but not quite, because it's broken. The BrokenRank p_i of webpage i is

$$p_i = \sum_{j \rightarrow i} \frac{p_j}{m_j} = \sum_{j=1}^{n} L_{ij} \frac{p_j}{m_j},$$

Does this match our ideas from the last slide? Yes: for $j \rightarrow i$, the weight is $\frac{p_j}{m_j}$ — this increases with p_j, but decreases with m_j

BrokenRank in matrix notation

$$p = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}, \quad L = \begin{pmatrix} L_{11} & L_{12} & \dots & L_{1n} \\ L_{21} & L_{22} & \dots & L_{2n} \\ \vdots & & & \\ L_{n1} & L_{n2} & \dots & L_{nn} \end{pmatrix},$$

$$M = \begin{pmatrix} m_1 & 0 & \dots & 0 \\ 0 & m_2 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & m_n \end{pmatrix}$$

Now re-express definition on the previous page: the BrokenRank vector p is defined as $p = LM^{-1}p$

BrokenRank as a Markov chain

Think of a Markov Chain as a random process that moves between states numbered $1, \ldots n$ (each step of the process is one move). Recall that for a Markov chain to have an $n \times n$ transition matrix P, this means $\mathrm{P}(\text{go from } i \text{ to } j) = P_{ij}$

Suppose $p^{(0)}$ is an n-dimensional vector giving initial probabilities. After one step, $p^{(1)} = P^T p^{(0)}$ gives probabilities of being in each state (why?)

Now consider a Markov chain, with the states as webpages, and with transition matrix A^T. Note that $(A^T)_{ij} = A_{ji} = L_{ji}/m_i$, so we can describe the chain as

$$\mathrm{P}(\text{go from } i \text{ to } j) = \begin{cases} 1/m_i & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

Stationary distribution

A stationary distribution of our Markov chain is a probability vector p (i.e., its entries are ≥ 0 and sum to 1) with $p = Ap$

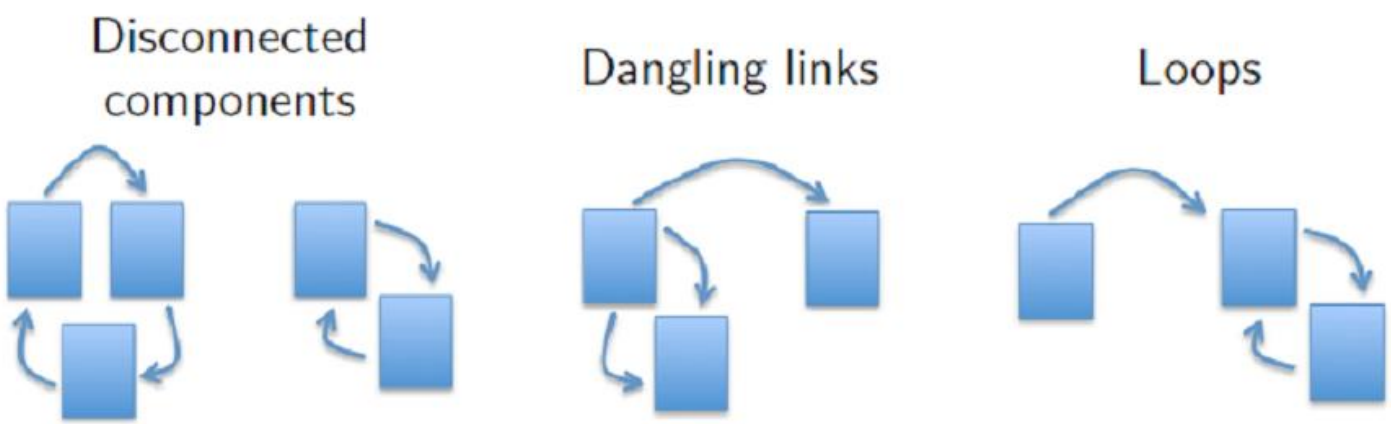
If the Markov chain is strongly connected, meaning that any state can be reached from any other state, then stationary distribution p exists and is unique. Furthermore, we can think of the stationary distribution as the of proportions of visits the chain pays to each state after a very long time (the ergodic theorem):

$$p_i = \lim_{t\to\infty} \frac{\#\text{ of visits to state } i \text{ in } t \text{ steps}}{t}$$

Our interpretation: the BrokenRank of p_i is the proportion of time our random surfer spends on webpage i if we let him go forever

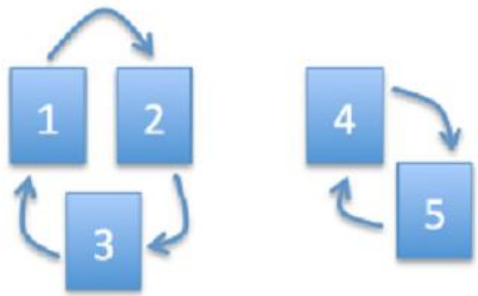
Why is BrokenRank broken?

There's a problem here. Our Markov chain—a random surfer on the web graph—is not strongly connected, in three cases (at least):



Actually, even for Markov chains that are not strongly connected, a stationary distribution always exists, but may nonunique

BrokenRank example



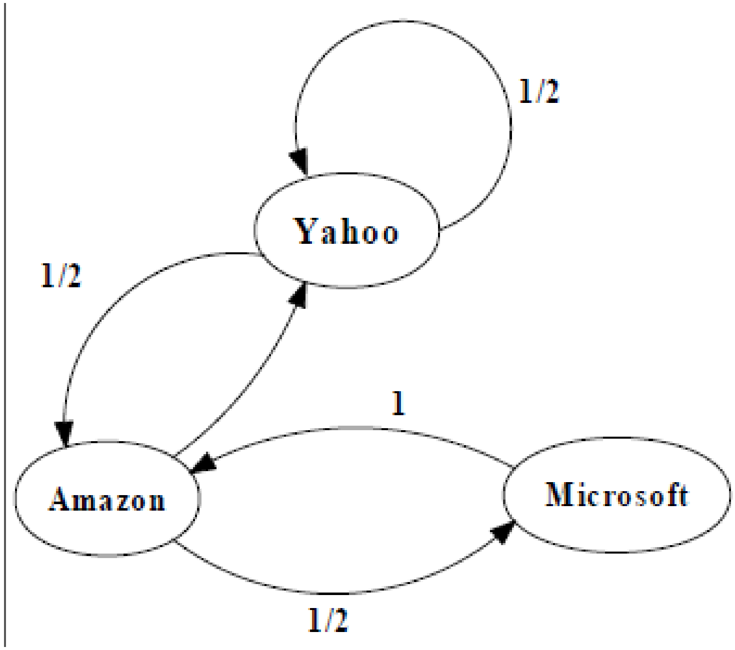
Here $A = LM^{-1} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

Here there are two eigenvectors of A with eigenvalue 1:

$$p = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad p = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

These are totally opposite rankings!

An example of Simplified PageRank



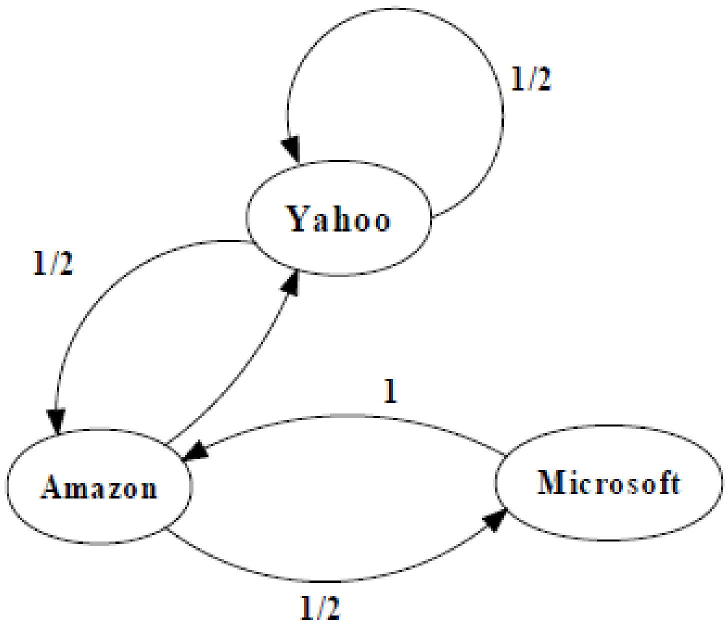
$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

PageRank Calculation: first iteration

An example of Simplified PageRank



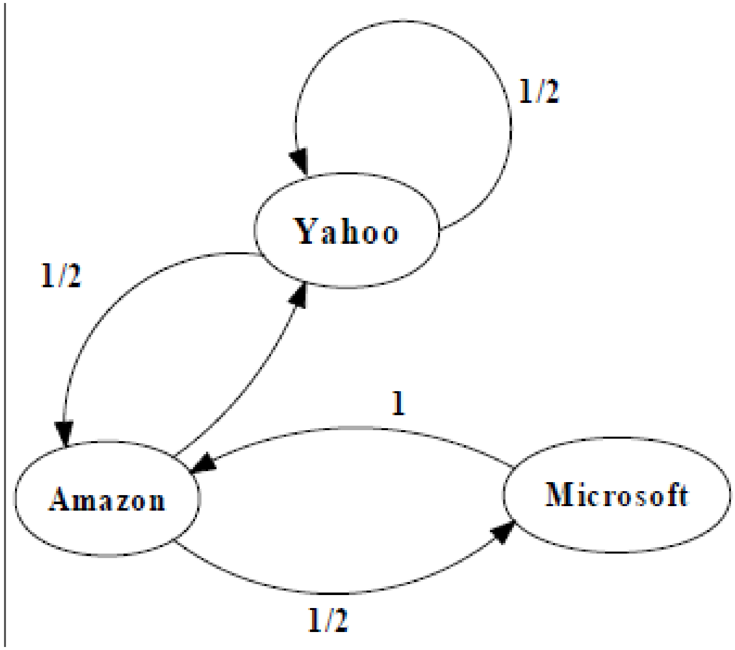
$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 5/12 \\ 1/3 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix}$$

PageRank Calculation: second iteration

An example of Simplified PageRank



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

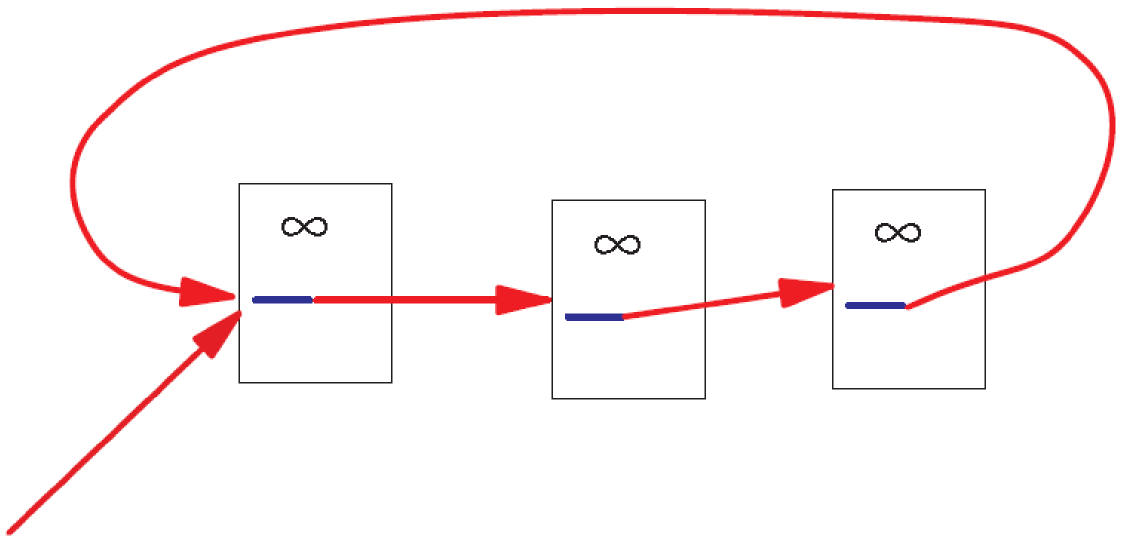
$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 3/8 \\ 11/24 \\ 1/6 \end{bmatrix} \begin{bmatrix} 5/12 \\ 17/48 \\ 11/48 \end{bmatrix} \ldots \begin{bmatrix} 2/5 \\ 2/5 \\ 1/5 \end{bmatrix}$$

Convergence after some iterations

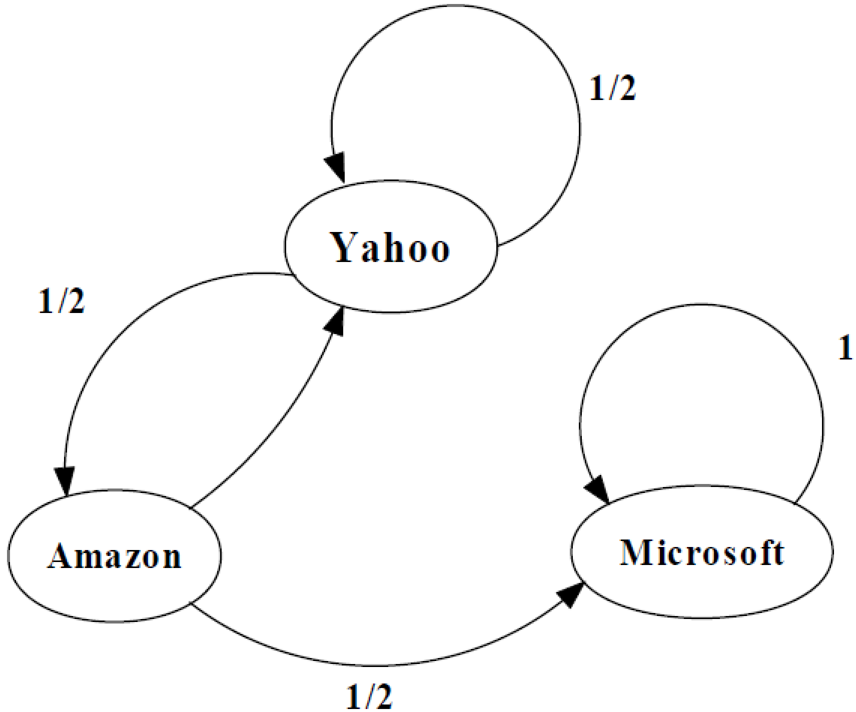
A Problem with Simplified PageRank

A loop:



During each iteration, the loop accumulates rank but never distributes rank to other pages!

An example of the Problem

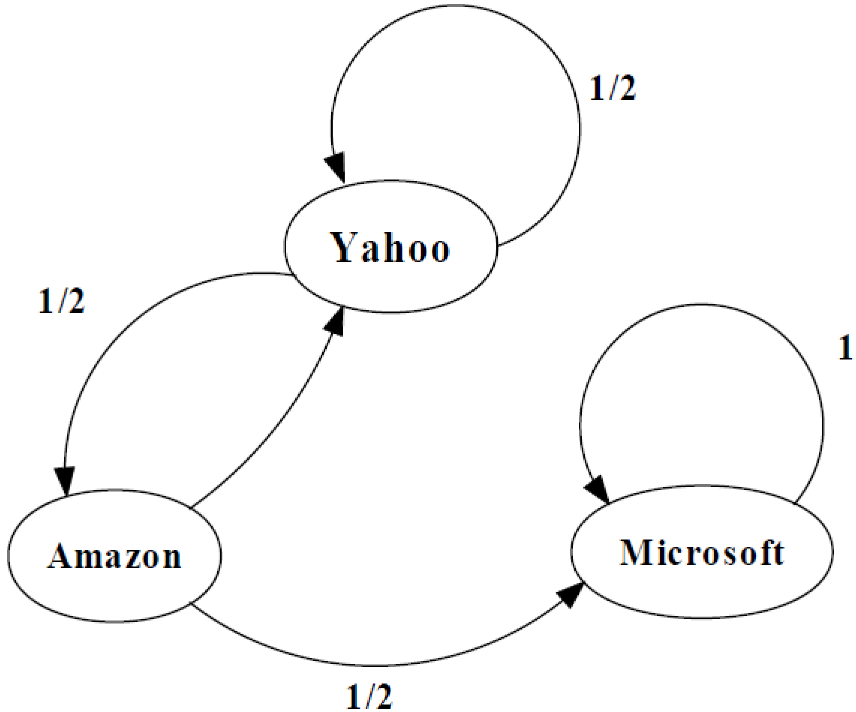


$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 1/3 \\ 1/6 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

An example of the Problem

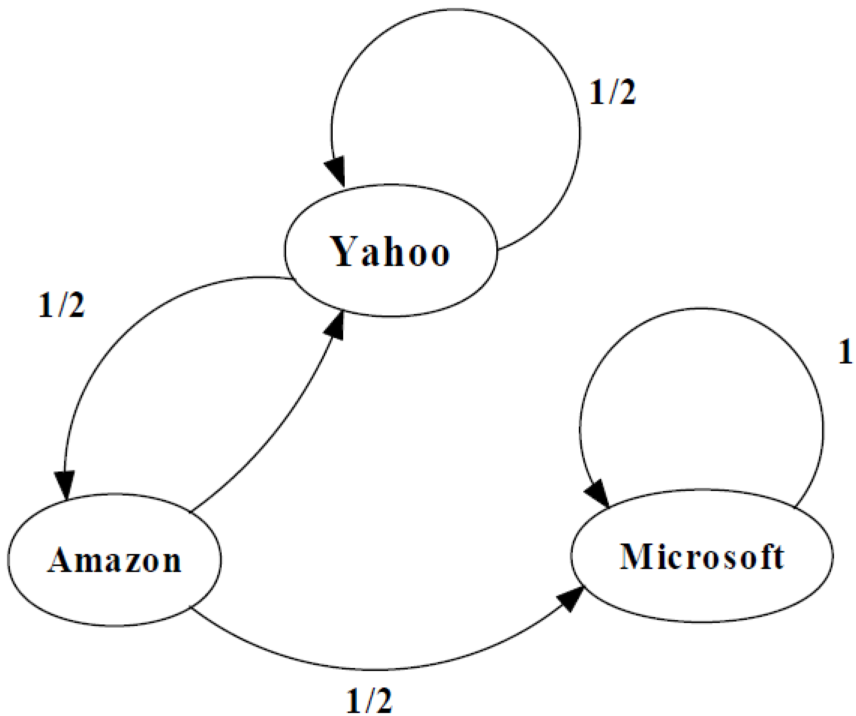


$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 1/4 \\ 1/6 \\ 7/12 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/6 \\ 1/2 \end{bmatrix}$$

An example of the Problem



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

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 5/24 \\ 1/8 \\ 2/3 \end{bmatrix} \begin{bmatrix} 1/6 \\ 5/48 \\ 35/48 \end{bmatrix} \dots \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Random Walks in Graphs

- The Random Surfer Model
  - – The simplified model: the standing probability distribution of a random walk on the graph of the web. simply keeps clicking successive links at random
- The Modified Model
  - – The modified model: the “random surfer” simply keeps clicking successive links at random, but periodically “gets bored” and jumps to a random page based on the distribution of E

Modified Version of PageRank

$$R'(u) = c_1 \sum_{v \in B_u} \frac{R'(v)}{N_v} + c_2 E(u)$$

E(u): a distribution of ranks of web pages that "users" jump to when they "gets bored" after successive links at random.

PageRank definition

PageRank is given by a small modification of BrokenRank:

$$p_i = \frac{1-d}{n} + d \sum_{j=1}^{n} \frac{L_{ij}}{m_j} p_j,$$

where $0 < d < 1$ is a constant (apparently Google uses $d = 0.85$)

In matrix notation, this is

$$p = (\frac{1-d}{n} E + dLM^{-1}) p,$$

where E is the $n \times n$ matrix of 1s, subject to the constraint $\sum_{i=1}^{n} p_i = 1$

PageRank as a Markov chain

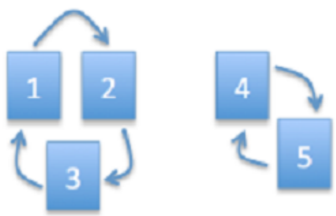
Let $A = \frac{1-d}{n}E + dLM^{-1}$, and consider as before a Markov chain with transition matrix A^T

Well $(A^T)_{ij} = A_{ji} = (1-d)/n + dL_{ji}/m_i$, so the chain can be described as

$$\mathrm{P}(\text{go from } i \text{ to } j) = \begin{cases} (1-d)/n + d/m_i & \text{if } i \to j \\ (1-d)/n & \text{otherwise} \end{cases}$$

Hence this is like a random surfer with random jumps. Fortunately, the random jumps get rid of our problems: our Markov chain is now strongly connected. Therefore the stationary distribution (i.e., PageRank vector) p is unique

PageRank example

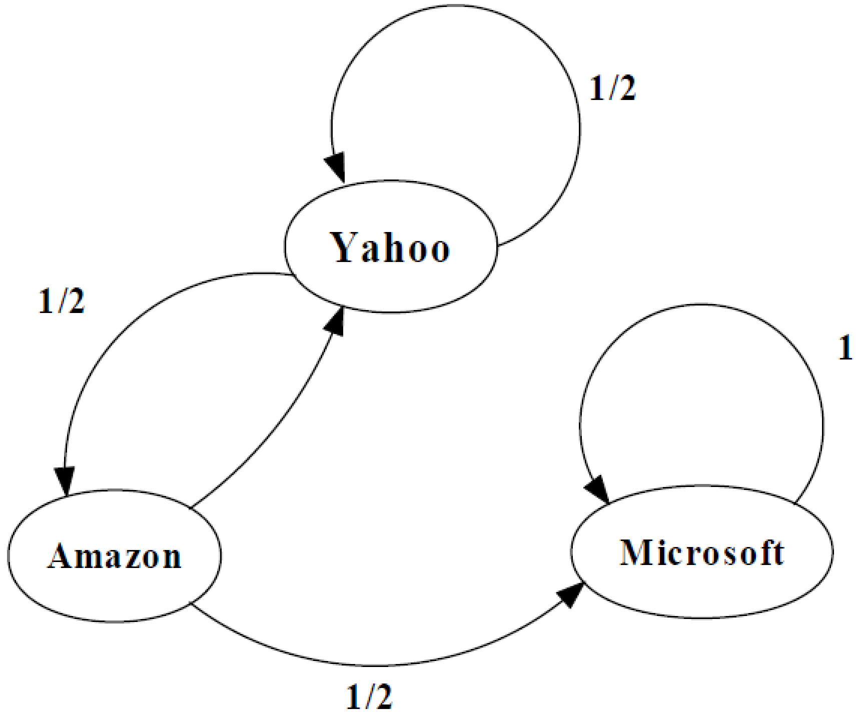


With $d = 0.85$, $A = \frac{1-d}{n}E + dLM^{-1}$

$$= \frac{0.15}{5} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} + 0.85 \cdot \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0.03 & 0.03 & 0.88 & 0.03 & 0.03 \\ 0.88 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.88 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.88 \\ 0.03 & 0.03 & 0.03 & 0.88 & 0.03 \end{pmatrix}$$

Now only one eigenvector of A with eigenvalue 1: $p = \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{pmatrix}$

An example of Modified PageRank



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

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$C_1 = 0.8 \qquad C_2 = 0.2$

$$\begin{bmatrix} 0.333 \\ 0.333 \\ 0.333 \end{bmatrix} \begin{bmatrix} 0.333 \\ 0.200 \\ 0.467 \end{bmatrix} \begin{bmatrix} 0.280 \\ 0.200 \\ 0.520 \end{bmatrix} \begin{bmatrix} 0.259 \\ 0.179 \\ 0.563 \end{bmatrix} \ldots \begin{bmatrix} 7/33 \\ 5/33 \\ 21/33 \end{bmatrix}$$

Computing the PageRank vector

Computing the PageRank vector p via traditional methods, i.e., an eigendecomposition, takes roughly n^3 operations. When $n = 10^{10}$, $n^3 = 10^{30}$.

Fortunately, much faster way to compute the eigenvector of A with eigenvalue 1: begin with any initial distribution $p^{(0)}$, and compute

$$\begin{aligned} p^{(1)} &= Ap^{(0)} \\ p^{(2)} &= Ap^{(1)} \\ &\vdots \\ p^{(t)} &= Ap^{(t-1)}, \end{aligned}$$

Computing the PageRank vector

Then $p^{(t)} \to p$ as $t \to \infty$. In practice, we just repeatedly multiply by A until there isn't much change between iterations

E.g., after 100 iterations, operation count: $100n^2 \ll n^3$ for large n

There are still important questions remaining about computing the PageRank vector p

1. How can we perform each iteration quickly (multiply by A quickly)?
2. How many iterations does it take (generally) to get a reasonable answer?

Computing the PageRank vector

Broadly, the answers are:

1. Use the sparsity of web graph (how?)

2. Not very many if A large spectral gap (difference between its first and second largest absolute eigenvalues); the largest is 1, the second largest is $\leq d$

Software package for PageRank calculations

NetworkX

Docs » Reference » Reference » Algorithms » Link Analysis

Link Analysis

PageRank

PageRank analysis of graph structure.

`pagerank` (G[, alpha, personalization, ...])	Return the PageRank of the nodes in the graph.
`pagerank_numpy` (G[, alpha, personalization, ...])	Return the PageRank of the nodes in the graph.
`pagerank_scipy` (G[, alpha, personalization, ...])	Return the PageRank of the nodes in the graph.
`google_matrix` (G[, alpha, personalization, ...])	Return the Google matrix of the graph.

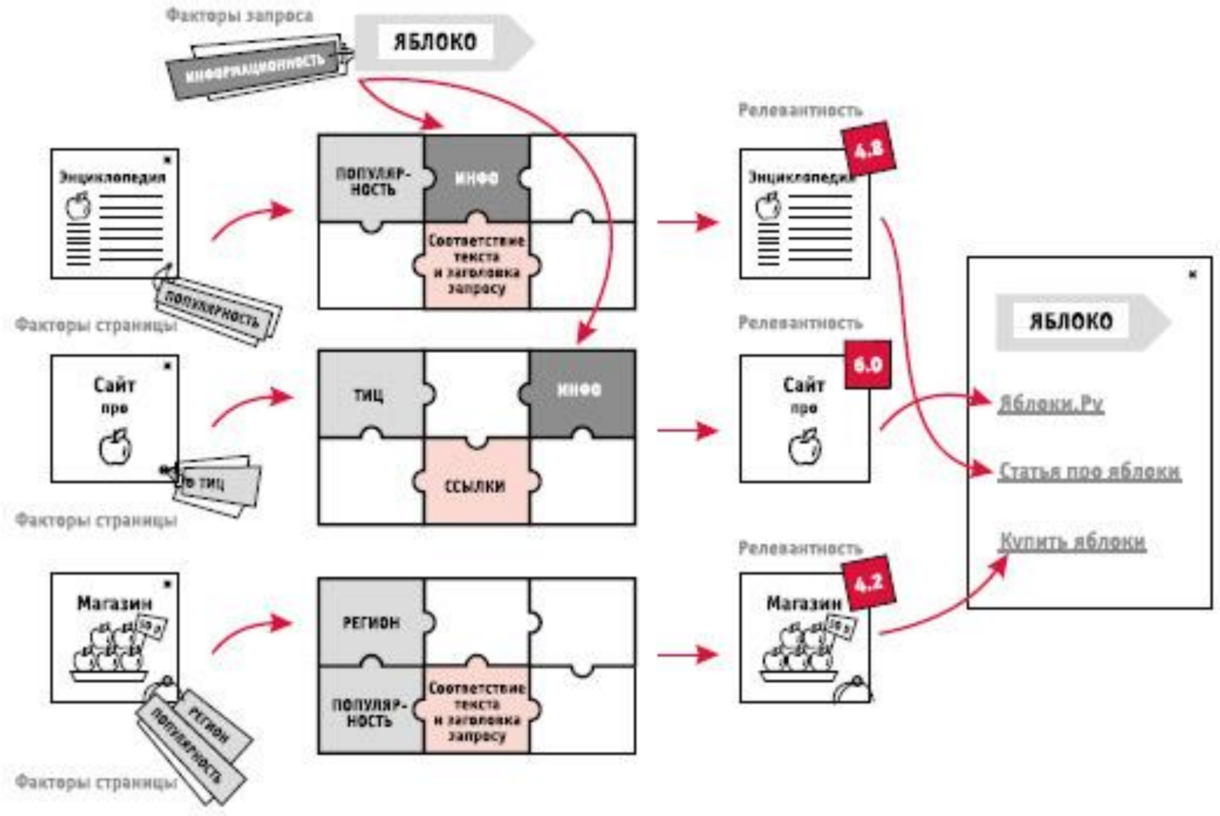
Searching with PageRank

- Two search engines:
  - Title-based search engine
  - Full text search engine

- Title-based search engine
  - Searches only the “Titles”
  - Finds all the web pages whose titles contain all the query words
  - Sorts the results by PageRank
  - Very simple and cheap to implement
  - Title match ensures high precision, and PageRank ensures high quality

- Full text search engine
  - Called Google
  - Examines all the words in every stored document and also performs PageRank (Rank Merging)
  - More precise but more complicated

Matrixnet: алгоритм Яндекс



Задача обучения классификатора

X — множество *объектов*;
Y — множество *ответов*;
$y\colon X \to Y$ — неизвестная зависимость (target function).

Дано:
$\{x_1, \dots, x_\ell\} \subset X$ — *обучающая выборка* (training sample);
$y_i = y(x_i)$, $i = 1, \dots, \ell$ — известные ответы.

Найти:
$a\colon X \to Y$ — алгоритм, решающую функцию (decision function), приближающую y на всём множестве X.

Весь курс машинного обучения — это конкретизация:

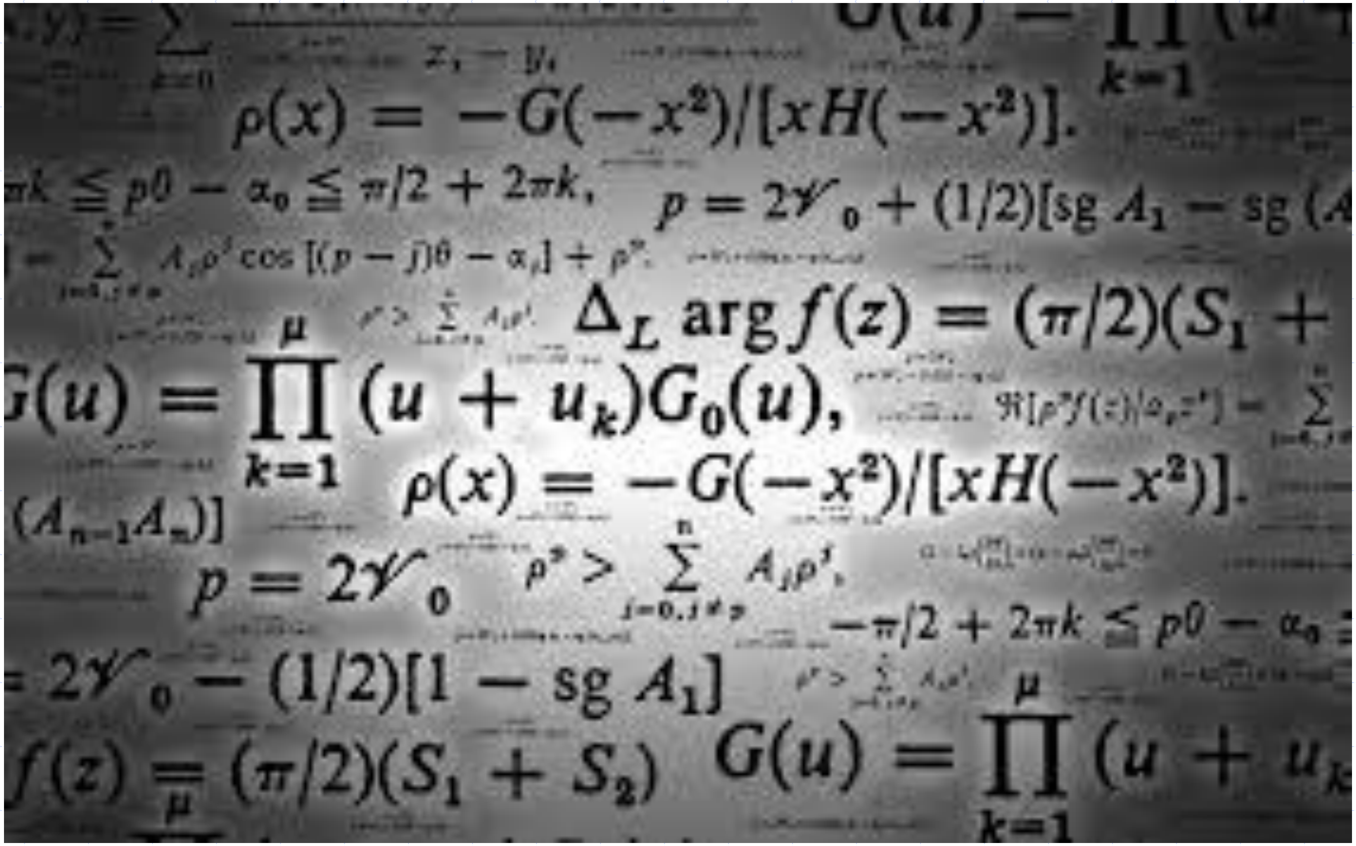
- как задаются объекты и какими могут быть ответы;
- в каком смысле «a приближает y»;
- как строить функцию a.

Обучение PageRank

f(q,d) – вектор признаков, зависящий от запроса q и страницы d;

ответы – это ранги страниц d относительно запросов q.

Математика поможет:



Спасибо за терпение!