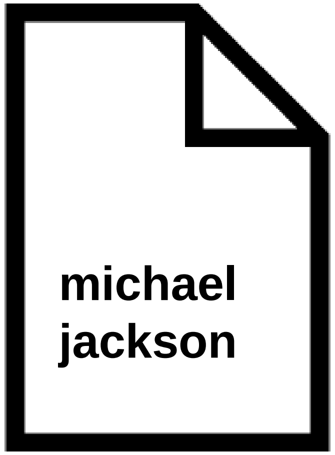
DD2476: Lecture 5

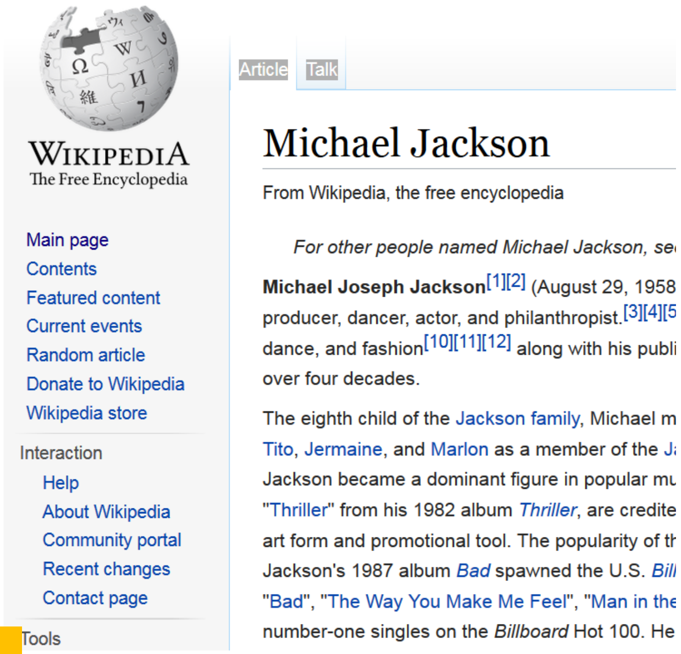
- Recap:
- **Vector-space model** to represent documents and queries
- **Tf-idf weighting**
- **Cosine similarity**

However, a problem...

michael jackson



Better cosine similarity with the query!



Michael Jackson

From Wikipedia, the free encyclopedia

For other people named Michael Jackson, se

Michael Joseph Jackson[1][2] (August 29, 1958
producer, dancer, actor, and philanthropist.[3][4][5
dance, and fashion[10][11][12] along with his publi
over four decades.

The eighth child of the Jackson family, Michael m
Tito, Jermaine, and Marlon as a member of the J
Jackson became a dominant figure in popular mu
"Thriller" from his 1982 album *Thriller*, are credite
art form and promotional tool. The popularity of th
Jackson's 1987 album *Bad* spawned the U.S. *Bill*
"Bad", "The Way You Make Me Feel", "Man in the
number-one singles on the *Billboard* Hot 100. He

Solution: Static quality scores

- We want top-ranking documents to be both **relevant** and **authoritative**
  - Relevance – cosine scores
  - Authority – query-independent property

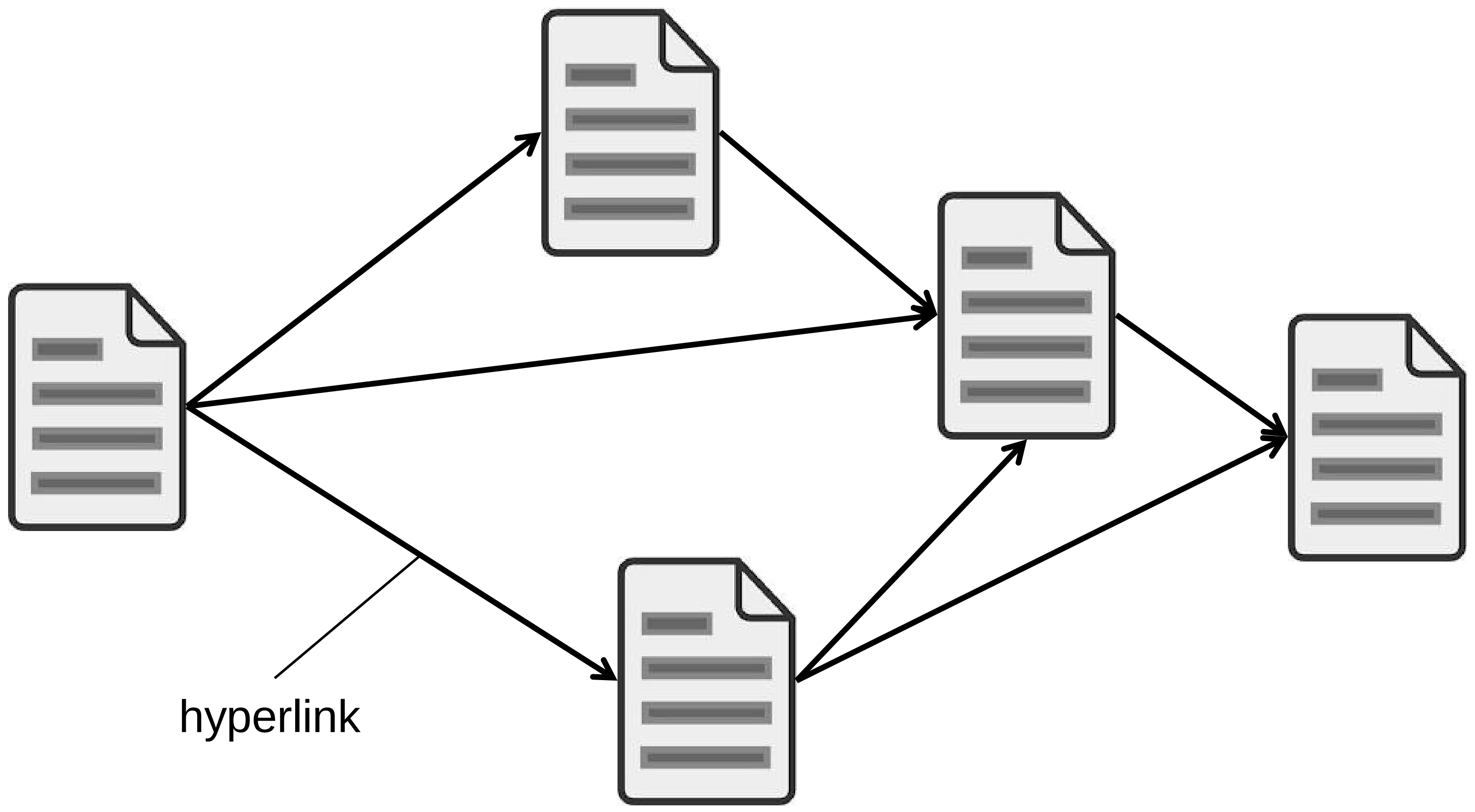
- Assign **query-independent quality score** g(*d*) in [0,1] to each document *d*

- net-score$(q,d) = w_1 * g(d) + w_2 * \cos(q,d)$
  - Two “signals” of user happiness

Static quality scores

- Which pages should be highly ranked? Alternatives:
  - Pages **visited by lots of users**
  - **Well-known quality** pages (Wikipedia, NY Times, …)
  - Pages with **many important in-links** (PageRank)
  - Personalised search: **Pages I've visited before**
  - Money talks: **Sponsored links**
  - (and recently): Veracity: **Pages with true information** are ranked highly, alternative facts less so.

The Web as a directed graph

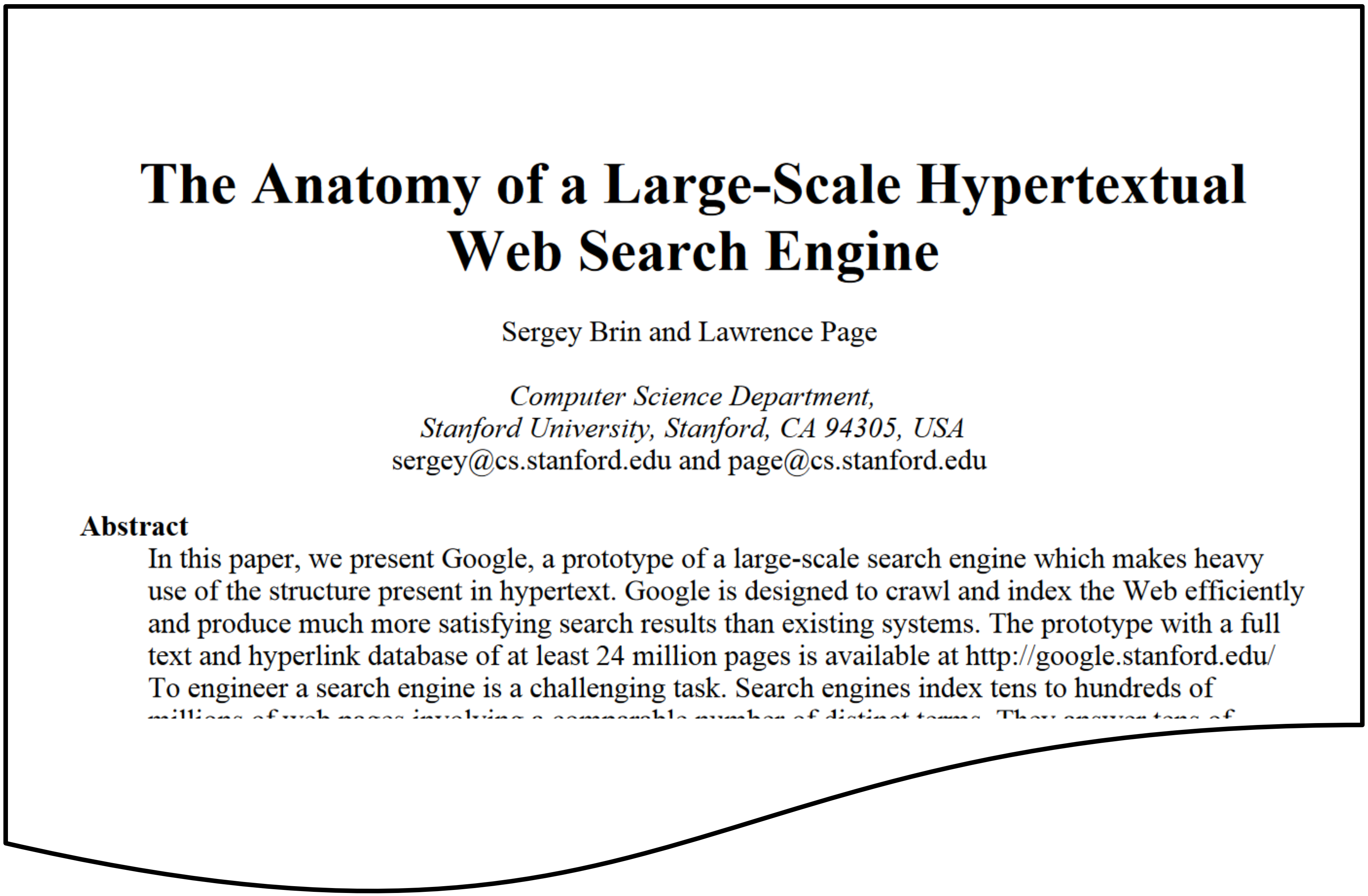


Using link structure for ranking

- **Assumption:** A link from X to Y signals that X’s author perceives Y to be an authoritative page.
  - X “casts a vote” on Y.
- **Simple suggestion:** Rank = number of in-links
- However, there are problems with this naive approach.

PageRank: basic ideas

- WWW's particular structure can be exploited
  - pages have links to one another
  - the more in-links, the higher rank
  - in-links from pages having **high rank** are **worth more** than links from pages having low rank
  - this idea is the cornerstone of **PageRank** (Brin & Page 1998)



The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/ To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of

PageRank – first attempt

$$PR(p) = \sum_{q \in in(p)} \frac{PR(q)}{L_q}$$

- *p* and *q* are pages
- *in(p)* is the set of pages linking to *p*
- L_q is the number of out-links from *q*

The random surfer model

- Imagine a **random surfer** that follow links
- The link to follow is selected with uniform probability
- If the surfer reaches a **sink** (a page without links), he **randomly restarts** on a new page
- Every once in a while, the surfer **jumps to a random page** (even if there are links to follow)

PageRank – second attempt

- With **probability *1-c*** the surfer is bored, **stops following links**, and **restarts** on a random page
- Guess: Google uses *c*=0.85

$$PR(p) = c\left(\sum_{q \in in(p)} \frac{PR(q)}{L_q}\right) + \frac{(1-c)}{N}$$

- Without this assumption, the surfer will get stuck in a subset of the web.

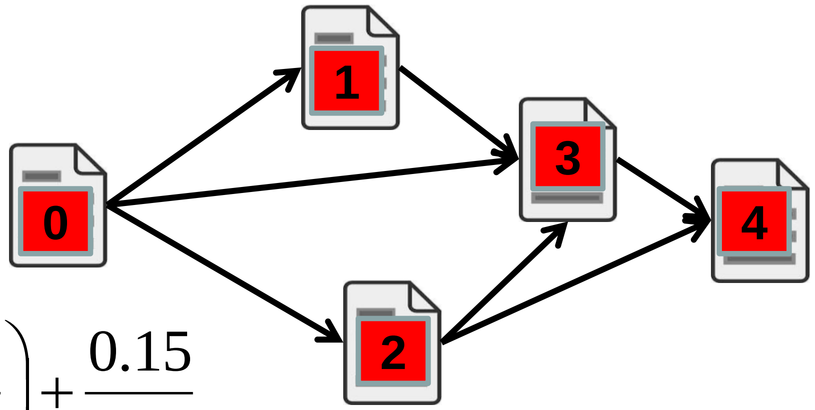
PageRank example

$$PR_4 = 0.85 \cdot \left(\frac{PR_2}{2} + PR_3 \right) + \frac{0.15}{5}$$

$$PR_3 = 0.85 \cdot \left(\frac{PR_0}{3} + PR_1 + \frac{PR_2}{2} + \frac{PR_4}{5} \right) + \frac{0.15}{5}$$

$$PR_2 = PR_1 = 0.85 \cdot \left(\frac{PR_0}{3} + \frac{PR_4}{5} \right) + \frac{0.15}{5}$$

$$PR_0 = 0.85 \cdot \left(\frac{PR_4}{5} \right) + \frac{0.15}{5}$$

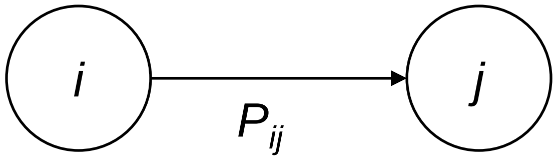


PageRank- interpretations

- Authority / popularity / relative information value
- PR_p = the probability that the random surfer will be at page *p* at any given point in time
- This is called the **stationary probability**
- How do we compute it?

Random surfer as a Markov chain

- The random surfer model suggests a a **Markov chain** formulation
  - A Markov chain consists of n states, plus an $n \times n$ transition probability matrix **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.



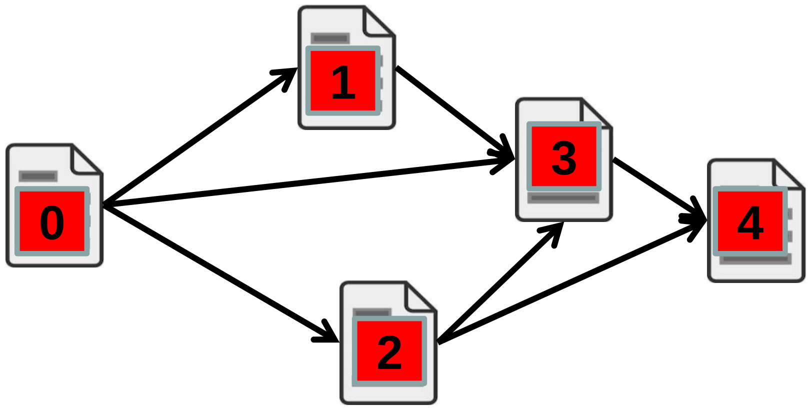
$$\begin{tikzcd} i \arrow[r, "P_{ij}"'] & j \end{tikzcd}$$

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.**
- Our transition matrix is non-zero positive everywhere $\leftrightarrow$
  the graph is strongly connected $\leftrightarrow$
  the Markov chain is ergodic $\leftrightarrow$
  unique stationary probabilities exist

Transition matrices

$$P = \begin{bmatrix} 0 & 0.33 & 0.33 & 0.33 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix}$$

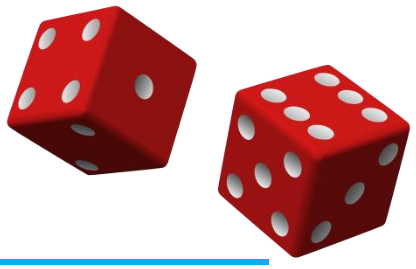


$$J = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix}$$

$G = cP+(1-c)J$

$$\begin{bmatrix} 0.0300 & 0.3105 & 0.3105 & 0.3105 & 0.0300 \\ 0.0300 & 0.0300 & 0.0300 & 0.8800 & 0.0300 \\ 0.0300 & 0.0300 & 0.0300 & 0.4550 & 0.4550 \\ 0.0300 & 0.0300 & 0.0300 & 0.0300 & 0.8800 \\ 0.2000 & 0.2000 & 0.2000 & 0.2000 & 0.2000 \end{bmatrix}$$

Monte Carlo simulation



- Probabilistic simulation can be used to estimate functions
- Common example: What is the expected value of a roll of two dice?
- Exact solution can be calculated with the formula

$$E[z] = \sum z\, p(z)$$

- We can estimate E[z] by throwing two dice a large number of times and calculate the average number of eyes shown.

Monte Carlo simulation to approximate PageRank

- Avrachenkov et al. describe a number of simulation algorithm based on the random surfer model
- D = document id
- Consider a random walk $\{D_t\}_{t\geq 0}$ that starts from a randomly chosen page.
- At each step t:
  - Prob c: D_t = one of the documents with edges from D_{t-1}
  - Prob (1 – c): The random walk terminates

1. MC end-point with random start

- Simulate N runs of the random walk $\{D_t\}_{t\geq 0}$ initiated at a **randomly chosen page**
- PageRank of page j = 1,…,n:

  $\boldsymbol{\pi_j = (\#\text{walks which end at } j)/N}$
- **Worst case: N = $O(n^2)$**
- **Mean case: N = O(n)**

2. MC end-point with cyclic start

- Simulate $N = mn$ runs of the random walk $\{D_t\}_{t\geq 0}$ initiated at **each page exactly m times**
- PageRank of page $j = 1,\dots,n$:

$$\boldsymbol{\pi_j = (\#\text{walks which end at } j)/N}$$

3. MC complete path

- Simulate N = mn runs of the random walk $\{D_t\}_{t \geq 0}$ of length T, initiated at **each page exactly m times**
- PageRank of page j = 1,…,n:
- **π_j = (#visits to node j during walks)/(NT_j)**

4. MC complete path stopping at dangling nodes

- Simulate N = mn runs of the random walk $\{D_t\}_{t \geq 0}$ initiated at **each page exactly m times** and **stopping when it reaches a dangling node**
- PageRank of page j = 1,…,n:

  π_j = (#visits to node j during walks)/ (total #visits during walks)

5. MC complete path with random start

- Simulate N runs of the random walk $\{D_t\}_{t\geq 0}$ initiated at a **randomly chosen page** and **stopping when it reaches a dangling node**
- PageRank of page j = 1,...,n:

  π_j = (#visits to node j during walks)/ (total #visits during walks)

Monte Carlo advantages

- **Power interation:**
  - precision improves linearly for all docs
  - computationally expensive
  - must be redone when new pages are added

- **Monte Carlo method:**
  - precision improves faster for high-rank docs
  - parallel implementation possible
  - continuous update

Approximating PageRank

- Power iteration is slow – every iteration requires N^2 multiplications
  - with our Davis corpus, that's $3*10^8$ multiplications
- However:
  - many of these multiplications are unnecessary
  - random jumps can be computed faster
  - if there are few sinks, jumps from sinks can be approximated faster

Another approximation method

```
for every i:
    for every link i→j:
        x′[j] += x[i]*c/out[i]
    x′[i] += (1-c)/N
    x′[i] += s/N/N
x = x′
x′= 0
```

where *s* is the number of sinks
Iterate until convergence.
At the end, one needs to normalize (divide every x[i] by $\Sigma_{\text{all k}}$ x[k])

Assignment 2

- Ranked search using the vector space model and tf-idf
- Computing PageRank by power interation
- Approximating PageRank by Monte-Carlo algorithms
- Putting it all together
- Review 6 March 10-12 and 13-15.