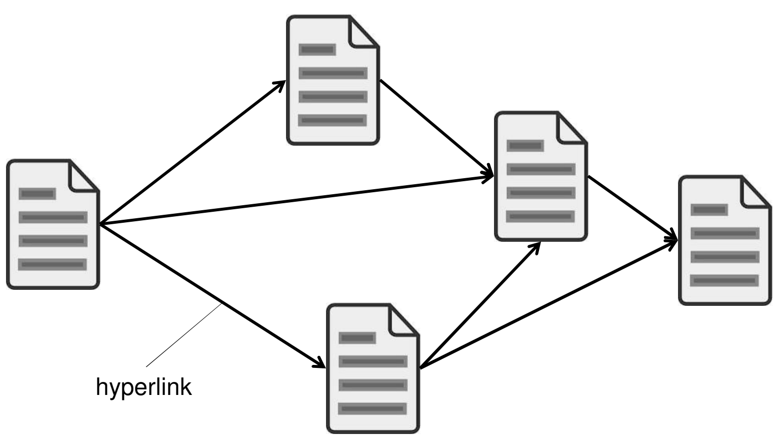
DD2476: Lecture 5

- Recap: Ranked retrieval
- We want top-ranking documents to be both **relevant** and **authoritative**
  - Relevance – cosine scores *cos(q,d)*
  - Authority – query-independent property *g*(*d*)
  - *net_score(q,d) = cos(q,d) + g(d)*
- **PageRank** is a way of estimating the authority of a page

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)
  - A "random surfer" that randomly follows links will spend more time on pages with high PageRank

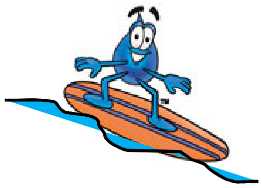
PageRank – first attempt

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

- 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

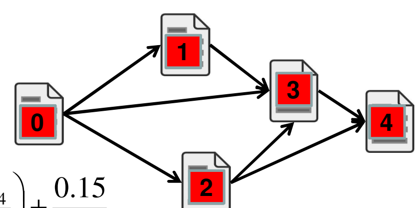
- With **probability *1-c*** the surfer is bored, **stops following links**, and **restarts** on a random page
- Guess: Google uses c=0.85

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

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

PageRank example

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



$$PR_3 = 0.85 \cdot \left(\frac{PR_0}{3} + PR_1 + \frac{PR_2}{2} + \frac{PR_4}{4}\right) + \frac{0.15}{5}$$

$$PR_2 = PR_1 = 0.85 \cdot \left(\frac{PR_0}{3} + \frac{PR_4}{4}\right) + \frac{0.15}{5}$$

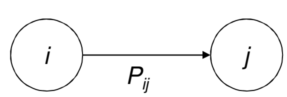
$$PR_0 = 0.85 \cdot \left(\frac{PR_4}{4}\right) + \frac{0.15}{5}$$

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 $1 \le i,j \le n$, the matrix entry P_{ij} tells us the probability of *j* being the next state, given we are currently in state *i*.

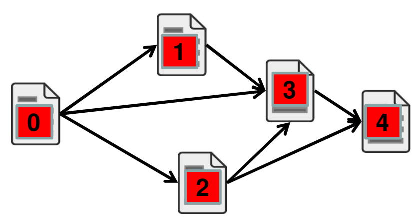


Ergodic Markov chains

- 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 everywhere $\leftrightarrow$ the graph is strongly connected $\leftrightarrow$ the Markov chain is ergodic $\leftrightarrow$ **unique stationary probabilities** exist

Transition matrices

$$P = \begin{bmatrix} 0 & 0.33 & 0.33 & 0.33 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0 \end{bmatrix}$$



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

G = cP+(1-c)J

$$\begin{bmatrix} 0.0300 & 0.3105 & 0.3105 & 0.3105 & 0.0300 \\ 0.0300 & 0.0300 & 0.0300 & 0.8800 & 0.0300 \\ 0.0300 & 0.0300 & 0.0300 & 0.4550 & 0.4550 \\ 0.0300 & 0.0300 & 0.0300 & 0.0300 & 0.8800 \\ 0.2425 & 0.2425 & 0.2425 & 0.2425 & 0.0300 \end{bmatrix}$$

Probability vectors

- A **probability** (row) **vector x** = (x_1, … x_n) tells us where the walk is at any point.
- E.g., (000…1…000) means we're in state *i*.

  1 i n

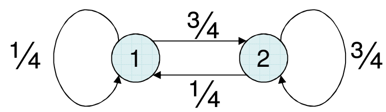
- More generally, the vector **x** = (x_1, … x_n) means the walk is in state *i* with probability x_i.
- **xG** gives the next time step.
- **x** is stationary if **x=xG**

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

Stationary probabilites

- Let **a** = (a_1, … a_n) denote the row vector of stationary probabilities.
  - If our current position is described by **a**, then the next step is distributed as **aG**.
  - But **a** is the steady state, so **a=aG**.
- Solving the matrix equation **x=xG** gives us **a**.
  - So **a** is the (left) eigenvector for **G**.
  - (Corresponds to the "principal" eigenvector of **G** with the largest eigenvalue.)
  - Transition probability matrices always have largest eigenvalue 1.

Example



For this example, the stationary probabilities are $a_1 = ¼$ and $a_2 = ¾$.

$$\begin{bmatrix} ¼ & ¾ \end{bmatrix} \begin{bmatrix} ¼ & ¾ \\ ¼ & ¾ \end{bmatrix} = \begin{bmatrix} ¼\cdot¼ + ¾\cdot¼ & ¼\cdot¾ + ¾\cdot¾ \end{bmatrix} = \begin{bmatrix} ¼ & ¾ \end{bmatrix}$$

Power iteration

- Recall, regardless of where we start, we eventually reach the stationary vector **a**.
- Start with any distribution (say **x**=(*10…0*)).
  - After one step, we're at **xG**;
  - after two steps at (**xG)G** , then **((xG)G)G** and so on.
- "Eventually", for "large" k, $\mathbf{xG}^k = \mathbf{a}$.

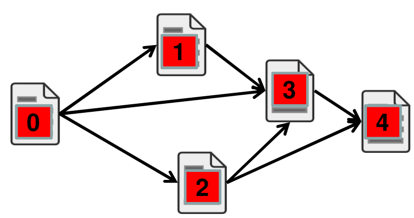
Power iteration algorithm

```
Let x=(0,…,0) and x' an initial
  state, say (1,0,…,0)

while ( |x-x'| > ε ):
     x = x'
     x'= xG
```

This algorithm converges **very** slowly.

Example



- Stationary probabilities:
  (0.102, 0.131, 0.131, 0.298, 0.339)

  found after 17 iterations starting from
  (0.2, 0.2, 0.2, 0.2, 0.2)

Approximating PageRank

- Power iteration is slow – every iteration requires N^2 multiplications
  - with our Wikipedia corpus, that's 10^{12} multiplications
- However:
  - many of these multiplications are unnecessary
  - random jumps can be computed faster
  - if there are few sinks (like in Wikipedia), jumps from sinks can be approximated faster

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 or a fixed number of times (e.g. 1000)
Caveat: this works well for Wikipedia but probably not for the web

DD2476 Search Engines and Information Retrieval Systems Lecture 5 Part 2: Monte Carlo Approximations of PageRank

Hedvig Kjellström
hedvig@kth.se
www.csc.kth.se/DD2476

Today

- Monte Carlo methods
  - Bishop, *Pattern Recognition and Machine Learning*, ch 11
- Five Monte Carlo approximations to PageRank
  - Avrachenkov et al, *SIAM* 2007, sec 1-2

Monte Carlo Methods

Approximate Solutions

- Huge #docs -> exact inference very expensive
  - Matrix factorization takes us part of the way
  - But eventually…
- Better solution: find approximation
- One way: Monte Carlo sampling

The Monte Carlo principle

- State space *z*
- Imagine that we can sample $z^{(l)}$ from the pdf $p(z)$ but that we do not know its functional form
- Might want to estimate for example:

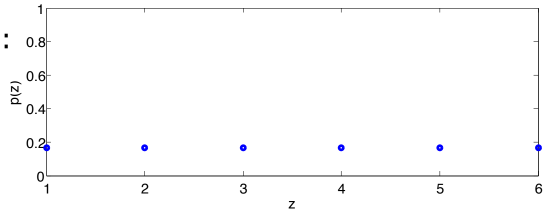
$$E[z] = \sum z\, p(z)$$

- $p(z)$ can be approximated by a histogram over $z^{(l)}$:

$$\hat{q}(z) = \frac{1}{L} \sum_{l=1}^{L} \delta_{z^{(l)}=z}$$

Example: Dice Roll

- The probability of outcomes of dice rolls: $p(z) = \frac{1}{6}$
- Exact solution:

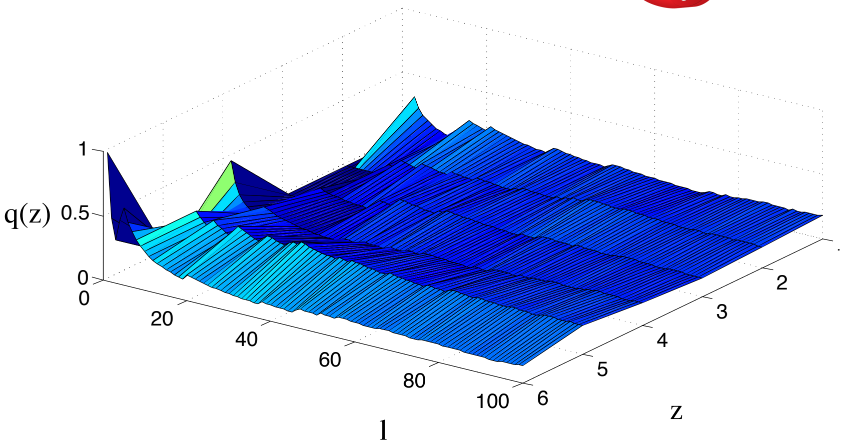


What would happen if the dice was bad?

- Monte Carlo approximation:
  - Roll a dice a number of times, might get

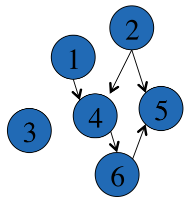
$$z^{(1)} = 6 \quad z^{(2)} = 4 \quad z^{(3)} = 1 \quad z^{(4)} = 6 \quad z^{(5)} = 6$$

Example: Dice Roll



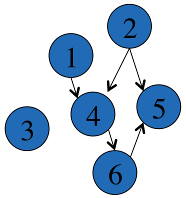
- The Law of Large Numbers

What is *p* and *q* for PageRank?



- Discuss with your neighbor (5 mins)
  - Graph of connected documents
  - Look at each document *z*, compute PageRank
- Quest: Find $p(z)$ = prob that the document *z* is visited = PageRank score of document *z*
- Monte Carlo approach: find approximate PageRank $\hat{q}(z)$ by sampling from $p(z)$

How do we sample from *p* without knowing *p*?



- Discuss with your neighbor (5 mins)

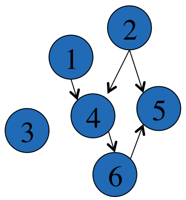
- Simulate a "random surfer" walking in the graph
  - Equal probability c/<#links> of selecting any of the <#links> links in a document D
  - Probability (1 - c) of not following links, but jumping to an unlinked document in the graph

- Record location $z^{(l)}$ at each step *l*

$$\hat{q}(z) = \frac{1}{L}\sum_{l=1}^{L} \delta_{z^{(l)}=z}$$

Five Monte Carlo Approximations to PageRank

Monte Carlo Idea



z above same as D here

- 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, and D_t = random node

x in first hour and q above same as π here

- Endpoint D_T is distributed as PageRank π when T→∞
- Sample from π = do many random walks (with limited T)

Advantages

- Exact method: precision improves linearly for all docs
- Monte Carlo method: precision improves faster for high-rank docs

- Exact method: computationally expensive
- Monte Carlo method: parallel implementation possible

- Exact method: must be redone when new pages are added
- Monte Carlo method: continuous update

1. MC end-point with random start

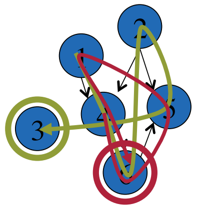
- Simulate N runs of the random walk $\{D_t\}_{t\geq 0}$ initiated at a **randomly chosen page**
- PageRank of page j = 1,...,n:

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

- **$N = O(n^2)$**, remember Law of Large Numbers

- Example:

  1 link 4 link 6 jump 2 link 5 jump 3
  4 link 6 link 5 jump 1 link 4 link 6
  π = [0, 0, 0.5, 0, 0, 0.5]
  2 walks not enough



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,...,n:

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

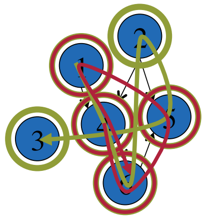
3. MC complete path

- 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,...,n:

  $$\pi_j = \textbf{(\#visits to node j during walks)/N}$$

- Example:

  1 link 4 link 6 jump 2 link 5 jump 3
  4 link 6 link 5 jump 1 link 4 link 6
  π = [1/6, 1/12, 1/12, 1/4, 1/6, 1/4]
  2 walks not enough



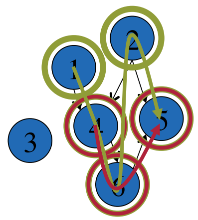
4. MC complete path stopping at dangling nodes

- 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:

  $$\pi_j = \textbf{(\#visits to node j during walks)/ (total \#visits during walks)}$$

- Example:

  1 link 4 link 6 jump 2 link 5
  4 link 6 link 5
  π = [1/8, 1/8, 0, 1/4, 1/4, 1/4]
  2 walks not enough



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:

$$\boldsymbol{\pi_j = (\#\text{visits to node } j \text{ during walks}) / (\text{total } \#\text{visits during walks})}$$

Next

- Assignment 1 left? Email Johan or Hedvig
- Lecture 6 (March 7, 13.15-15.00)
  - B1
  - Readings: Manning Chapter 9
- Lecture 7 (March 18, 13.15-15.00)
  - B1
  - Readings: Manning Chapter 11, 12
- Computer hall session (March 18, 15.00-19.00)
  - Gul (Osquars Backe 2, level 4)
  - Examination of computer assignment 2