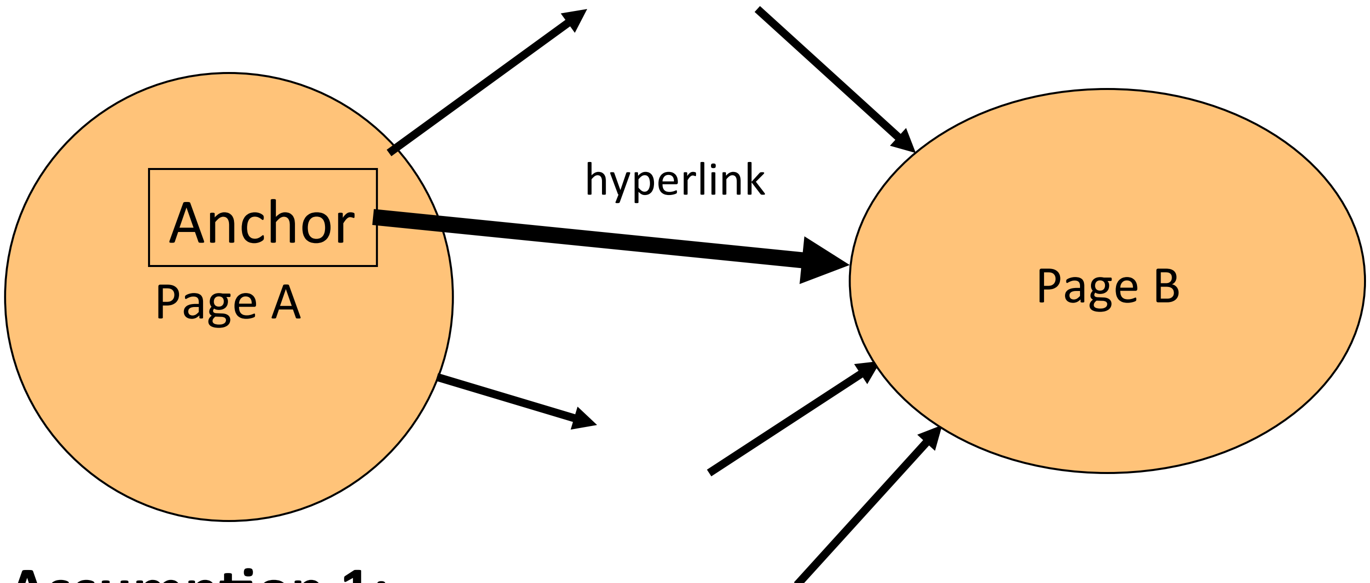
Networks and Link Analysis

Web Anchor Text

The Web as a Directed Graph

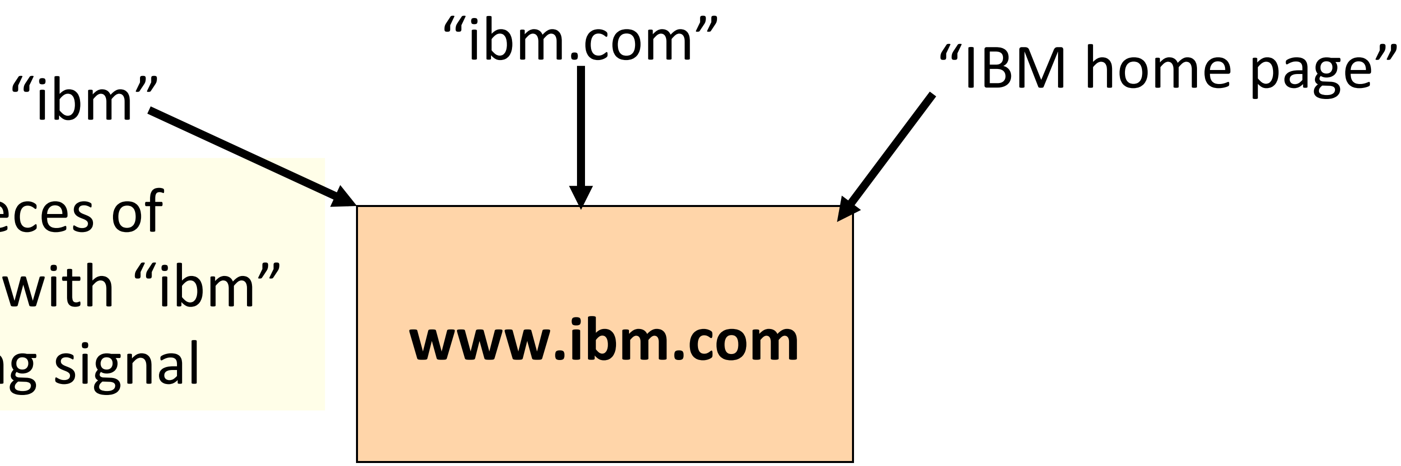


Assumption 1:
A hyperlink between pages denotes author perceived relevance (quality signal)

Assumption 2:
The anchor of the hyperlink describes the target page (textual context)

Anchor Text

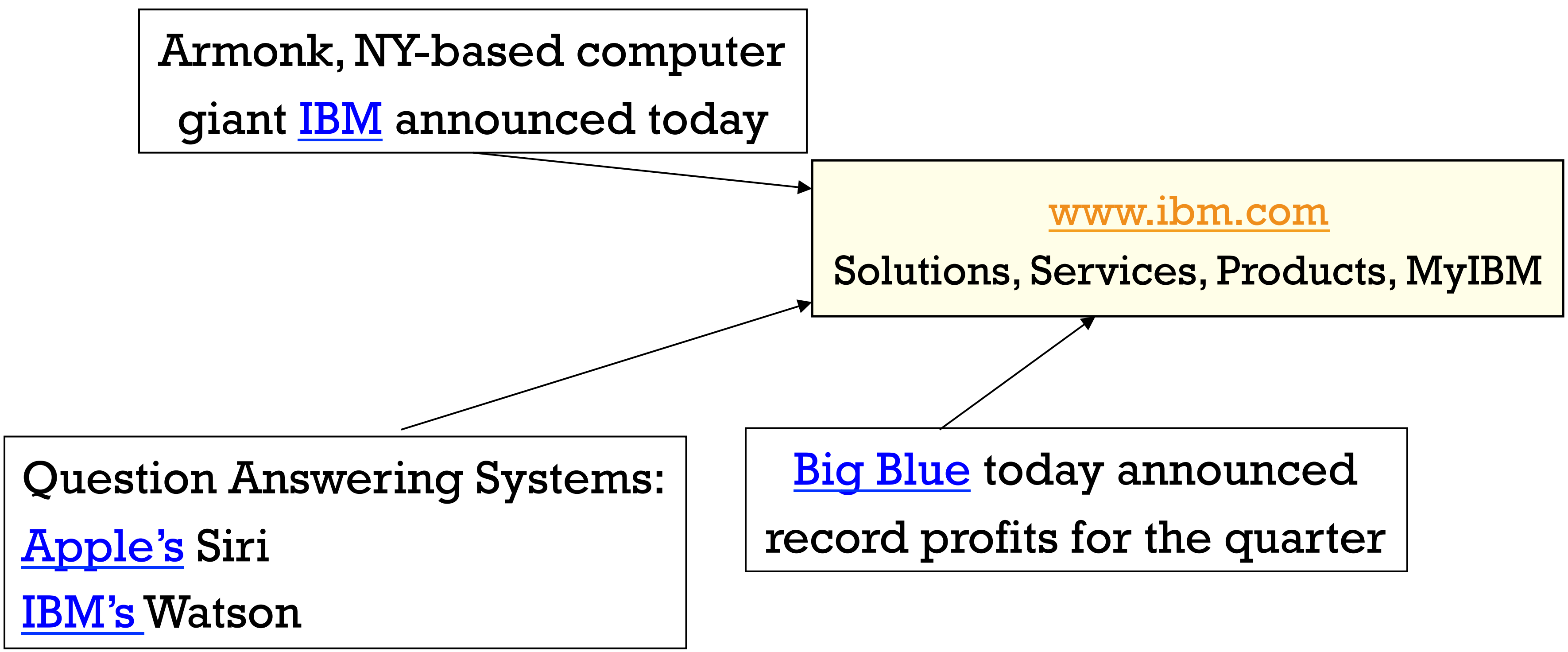
- For ***ibm*** how to distinguish between:
  - IBM's home page (mostly graphical)
  - IBM's copyright page (high term frequency for "ibm")
  - Rival's spam page (arbitrarily high term frequency)



A million pieces of anchor text with "ibm" send a strong signal

Indexing anchor text

When indexing a document *D*, include anchor text from links pointing to *D*.



Indexing anchor text

- Can sometimes have unexpected side effects –
  - Google bombing
- Can score anchor text with weight depending on the authority of the anchor page's website
  - E.g., if we were to assume that content from cnn.com or yahoo.com is authoritative, then trust the anchor text from them

Many NLP Applications of Anchor Text

- Finding synonyms
  - Federal Reserve: “Fed”, “U.S. Federal Reserve Board", “U.S. Federal Reserve System”, “Federal Reserve Bank”
- Finding translations of named entities
- Providing constituent boundaries for parsers

Networks and Link Analysis

Web Anchor Text

Networks and Link Analysis

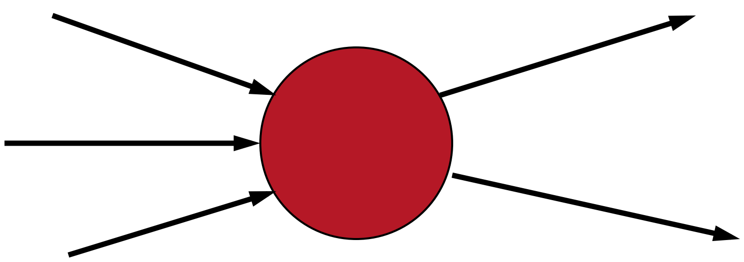
PageRank: Overview and Markov Chains

Combining link structure with text

- A good search result looks at more than just query-document text overlap
- One factor: page **popularity**.
  - Pages that are pointed to by lots of other pages are popular.
  - We can use link counts as a measure of static goodness,
  - Combine link counts with the text match score

Using link structure to measure page importance

- Simplest: use link counts as popularity measure
  - **Undirected popularity**:
    - Page score= **degree**: the number of in-links plus the number of out-links (3+2=5).
  - **Directed popularity**:
    - Page score = number of in-links (3).

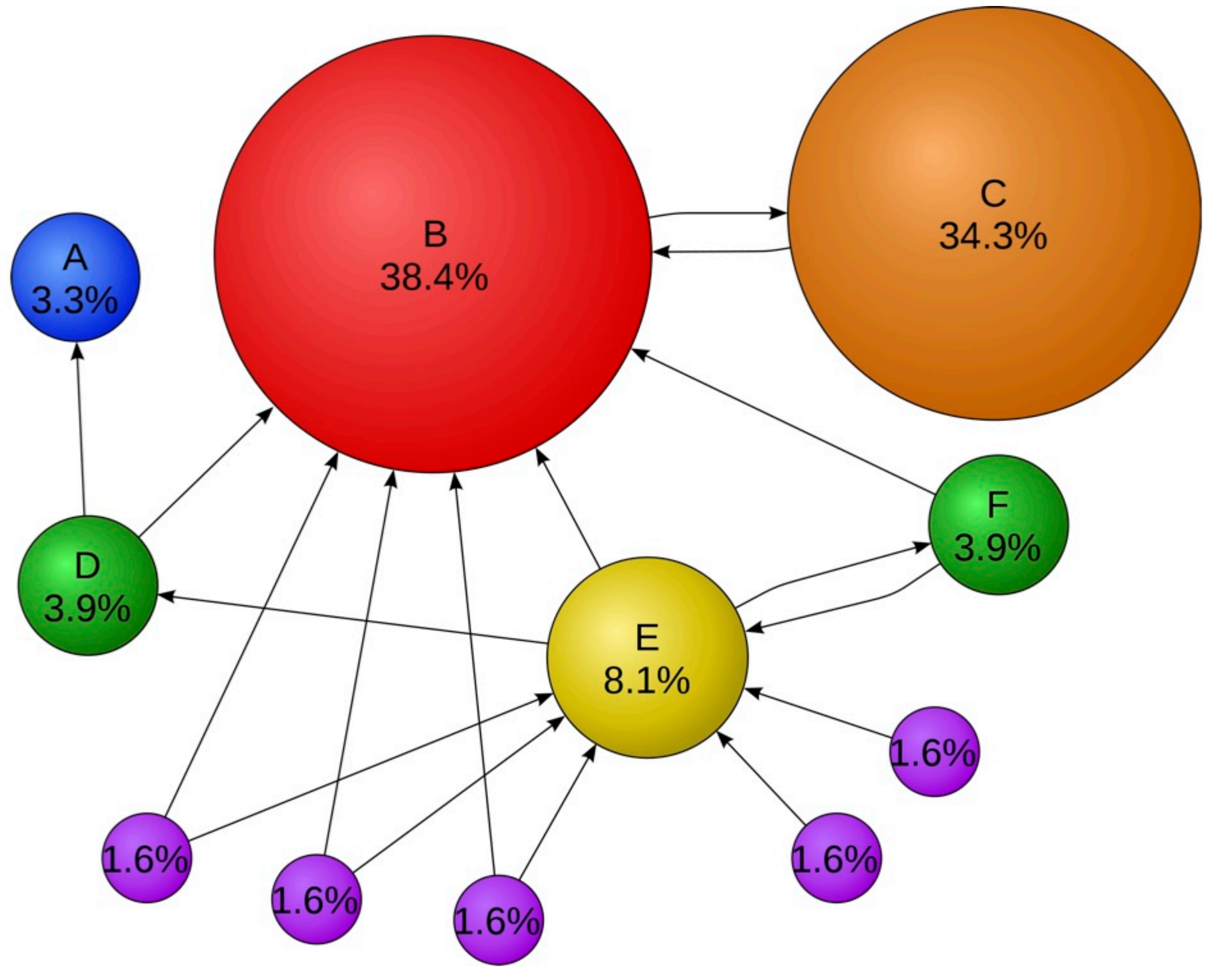


Spamming simple popularity

- Simple popularity heuristics can be spammed to give your page a high score, whether it's:
  - the number of in-links plus the number of out-links
  - number of in-links

Dan Jurafsky

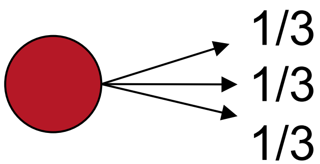
Intuition of PageRank



C has higher PageRank than E, even though E has more inlinks

PageRank scoring

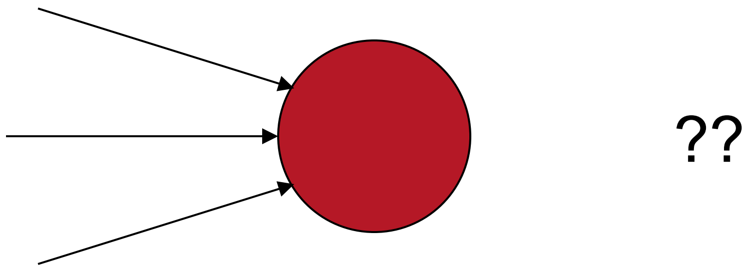
- Imagine a browser doing a random walk on web pages:
  - Start at a random page
  - At each step, go out of the current page along one of the links on that page, equiprobably



- "In the steady state" each page has a long-term visit rate - use this as the page's score.

Not quite enough

- The web is full of dead-ends.
  - Random walk can get stuck in dead-ends.
  - Makes no sense to talk about long-term visit rates.



Teleporting

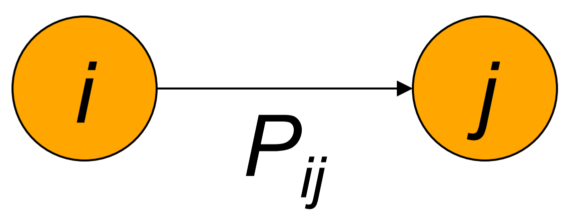
- At a dead end, jump to a random web page.
- At any non-dead end, with probability 10%, jump to a random web page.
  - With remaining probability (90%), go out on a random link.
  - 10% - a parameter.

Result of teleporting

- Now cannot get stuck locally.
- There is a long-term rate, the Pagerank, at which any page is visited

Markov chains

- A Markov chain:
  - *N* states,
  - *An N×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*.

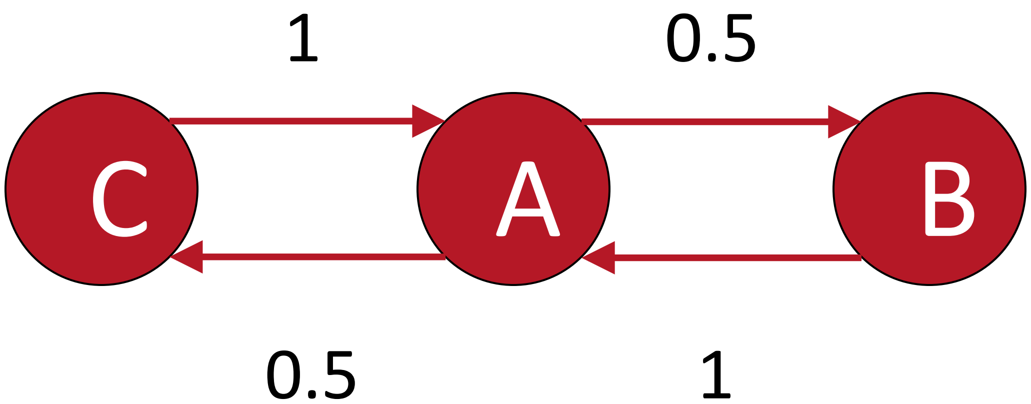


- For all i,

$$\sum_{j=1}^{n} P_{ij} = 1.$$

Dan Jurafsky

Markov chains



- Transition probability matrix P

$$P = \begin{pmatrix} 0 & 0.5 & 0.5 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Random Surfers and Markov chains

- Markov chains are abstractions of random walks.
- Each state
  - represents one web page
- Each transition probability
  - represents the probability of moving from one page to another
- We can derive the transition probability P from the adjacency matrix A of the web graph.

Teleporting, more formally

- If a node has no out-links, the random surfer teleports:
  - the transition probability to each node in the N-node graph is 1/N
- If a node has K>0 outgoing links:
  - with probability $0 < \alpha < 1$ the surfer teleports to a random node
    - probability is α/N
  - with probability $1 - \alpha$ the surfer takes a normal random walk
    - probability is $(1 - \alpha)/K$

Deriving transition probability matrix P from adjacency matrix A

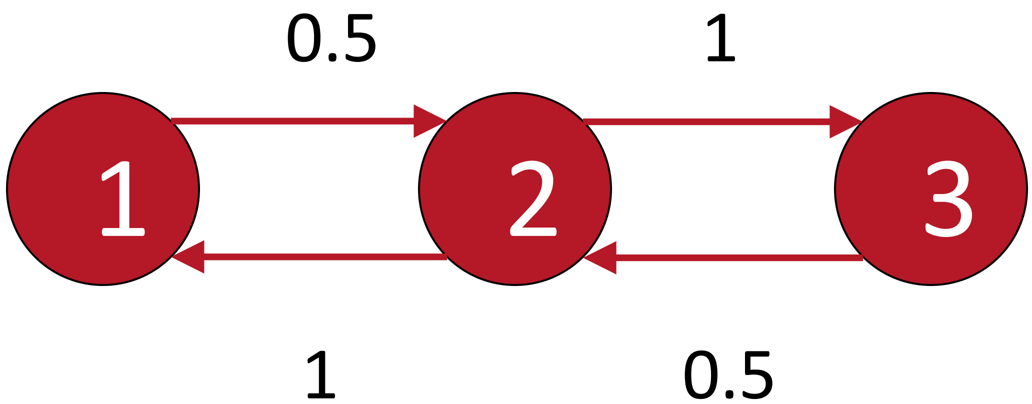
- A is the adjacency matrix of the web graph
  - A_{ij} is 1 if there is a hyperlink from page *i* to page *j*

- If a row of A has no 1's, then replace each element by 1/N.

For all other rows proceed as follows.

- Divide each 1 in A by the number of 1's in its row. Thus, if there is a row with three 1's, then each of them is replaced by 1/3
- Multiply the resulting matrix by $(1-\alpha)$
- Add α/N to every entry of the resulting matrix, to obtain P.

Computing P with teleportation



$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

P[1,*] = (1-α) (0 1 0) + α (1/N 1/N 1/N)

P[1,*] = 0.5 (0 1 0) + 0.5(1/3 1/3 1/3)

$$P_{\alpha=0} = \begin{pmatrix} 0 & 1 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 1 & 0 \end{pmatrix}$$

$$P_{\alpha=0.5} = \begin{pmatrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{pmatrix}$$

Networks and Link Analysis

PageRank: Overview and Markov Chains

Networks and Link Analysis

PageRank: Computation

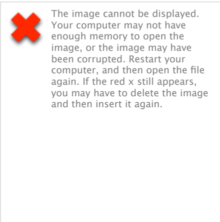
The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

Computing PageRank: The probability of being in a state

- A probability (row) vector $\mathbf{x} = (x_1, \ldots x_n)$ tells us where the walk is at any point.
- E.g., $(\underset{1}{0}00\ldots\underset{i}{1}\ldots00\underset{n}{0})$ means we're in state *i*.

More generally, the vector $\mathbf{x} = (x_1, \ldots x_n)$ means the walk is in state *i* with probability x_i.

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

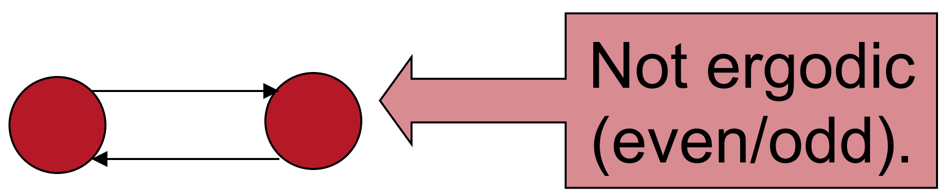


Computing PageRank: Change in probability vector

- If the probability vector is **x** $= (x_1, \ldots x_n)$ at this step, what is it at the next step?
- Recall that row *i* of transition matrix P tells us where we go next from state *i*.
- So from **x**, our next state is distributed as **xP**.

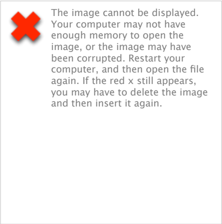
Ergodic Markov chains

- 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.



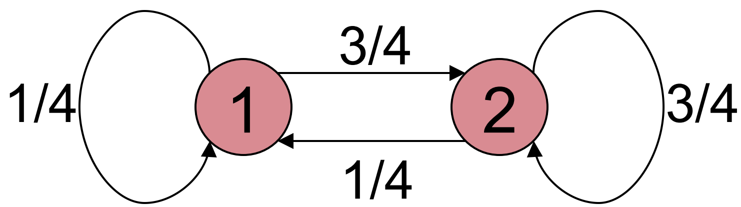
Ergodic Markov chains

- For any ergodic Markov chain, there is a unique long-term visit rate for each state.
  - A steady-state probability distribution $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots \boldsymbol{\pi}_n)$.
  - Over a long time-period, we visit each state in proportion to this rate.
  - Thus $\boldsymbol{\pi}_i$ is the PageRank of state i.
- It doesn't matter where we start.

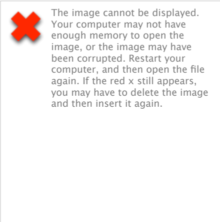


Steady state example

- The steady state looks like a vector of probabilities $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots \boldsymbol{\pi}_n)$:
  - $\boldsymbol{\pi}_i$ is the probability that we are in state *i*.

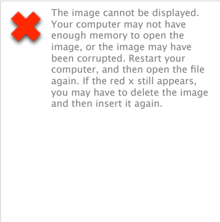


For this example, $\boldsymbol{\pi}_1 = 1/4$ and $\boldsymbol{\pi}_2 = 3/4$.



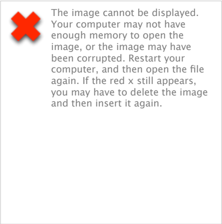
How do we compute this vector?

- Let $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots \boldsymbol{\pi}_n)$ denote the row vector of steady-state probabilities
- If our current position is described by **π**, then the next step is distributed as **πP**.
- But **π** is the steady state, so **π=πP**.
- Solving this matrix equation gives us **π**.
  - So **π** is the (left) eigenvector for **P**.
  - (Corresponds to the "principal" eigenvector of **P** with the largest eigenvalue.)
  - Transition probability matrices always have largest eigenvalue 1.



The power iteration method of computing π

- Recall, regardless of where we start, we eventually reach the steady state $\boldsymbol{\pi}$.
- Start with any distribution (say **x**=(*10…0*)).
- After one step, we're at **xP**;
- after two steps at $\mathbf{xP}^2$, then $\mathbf{xP}^3$ and so on.
- "Eventually" means for "large" k, $\mathbf{xP}^k = \boldsymbol{\pi}$.
- Algorithm: multiply **x** by increasing powers of **P** until the product looks stable.



Example of power iteration

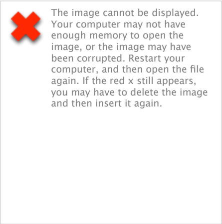
$$P_{\alpha=0.5} = \begin{pmatrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{pmatrix}$$

- Let's say surfer starts in state 1:

$$\vec{x}_0 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

$$\vec{x}_1 = \vec{x}_0 P = \begin{pmatrix} 1/6 & 2/3 & 1/6 \end{pmatrix}$$

$$\vec{x}_2 = \vec{x}_1 P = \begin{pmatrix} 1/6 & 2/3 & 1/6 \end{pmatrix} \begin{pmatrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{pmatrix} = \begin{pmatrix} 1/3 & 1/3 & 1/3 \end{pmatrix}$$



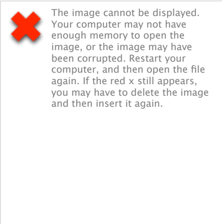
Power iteration example (continued)

$\vec{x}_0$	1	0	0
$\vec{x}_1$	1/6	2/3	1/6
$\vec{x}_2$	1/3	1/3	1/3
$\vec{x}_3$	1/4	1/2	1/4
$\vec{x}_4$	7/24	5/12	7/24
…	…	…	…
$\vec{x} = \vec{\pi}$	5/18	4/9	5/18

Node 1 PageRank

Node 2 PageRank

Node 3 PageRank



PageRank summary

- Preprocessing:
  - Given graph of links, build matrix **P**.
  - From it compute the PageRank vector $\boldsymbol{\pi}$.
  - The PageRank of page i, $\boldsymbol{\pi}_i$ is between 0 and 1
- Query processing:
  - Retrieve pages meeting query.
  - Rank them by their PageRank.
  - Order is query-*independent*.

Networks and Link Analysis

PageRank: Computation