

Mining Big Data

Link Analysis (Chapter 5)

- Name me 3 top internet companies



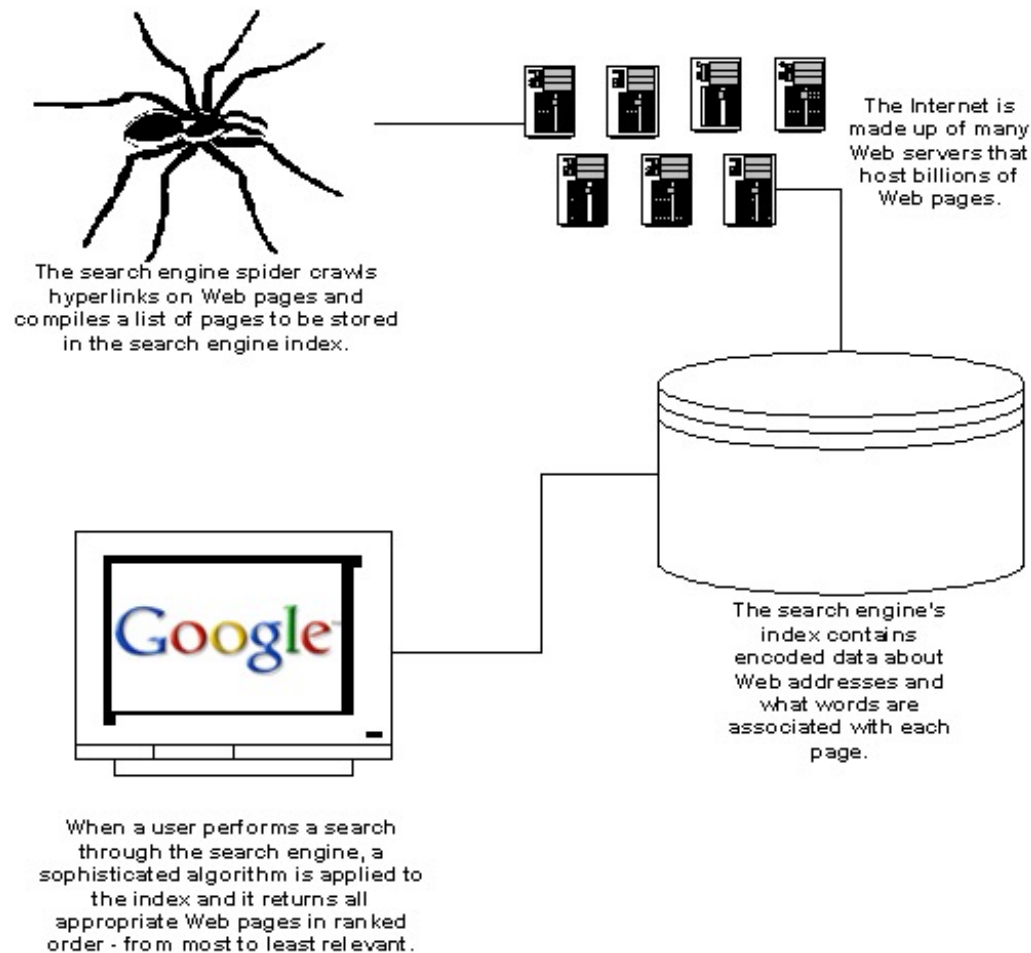
Google Search

I'm Feeling Lucky

Many search engines existed prior to Google



Web crawling



Old way of web retrieval/search

- A search query (list of terms) was issued
- The pages with those terms were ranked via the counts of the terms in a page

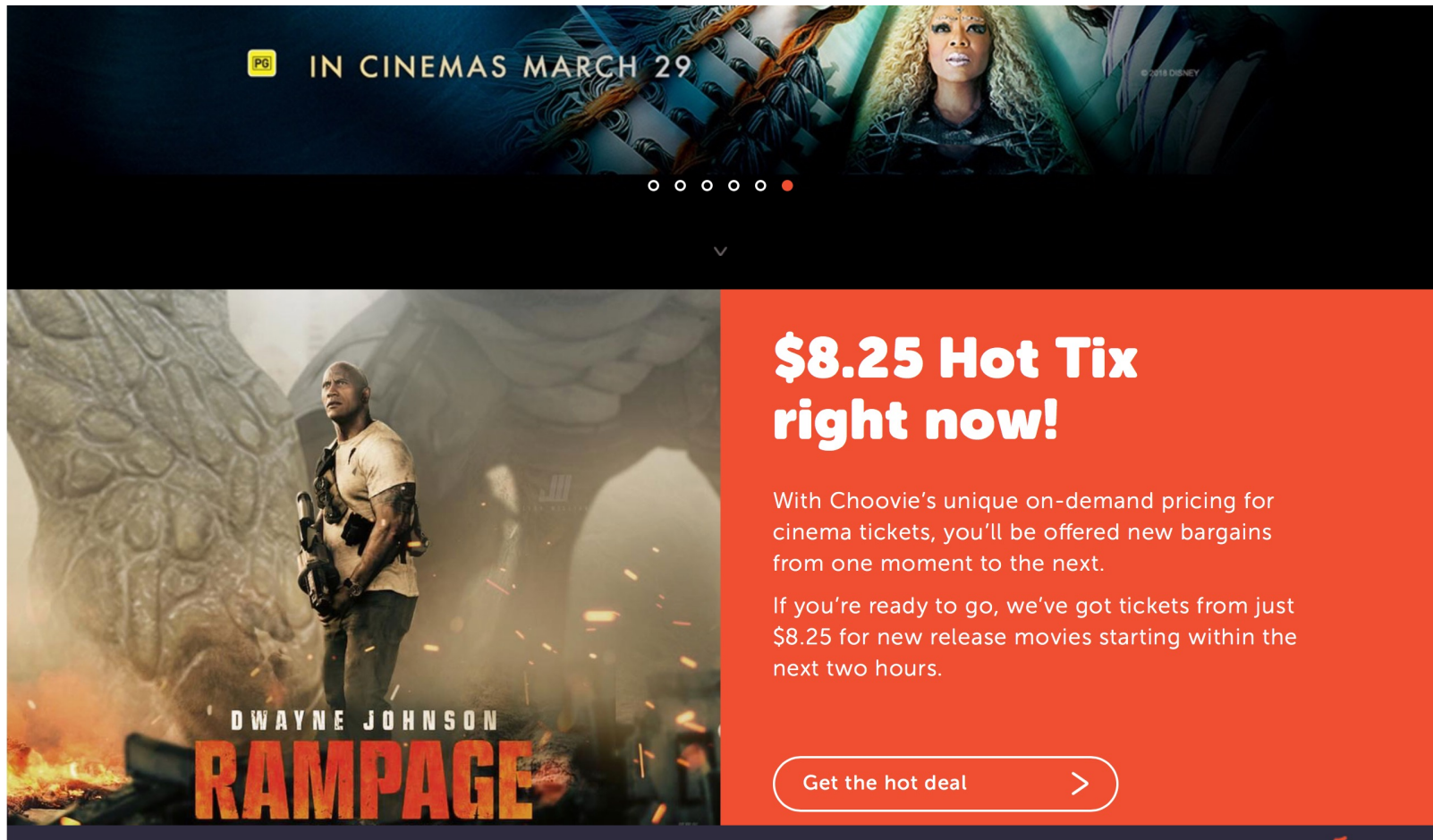
How to fool a search engine?

- You can pay a third party company (often called **spammers**) to boost any page of yours to be top ranked
- business is harder now as the search engine tech improves

Fake movie ticket selling website

- Say you have a **fake** movie ticket selling website that takes money from customers online but never issues a ticket.
- You want the website to be **highly ranked** to fish customers.
- If you get caught, and the website is banned, you will just put up a new one which will also be highly ranked again.

How to boost its ranking?



The advertisement is a horizontal banner. The top section is dark blue with a yellow 'PG' rating icon and the text 'IN CINEMAS MARCH 29'. Below this is a row of five white circles, with the fifth one being red. The main image shows Dwayne Johnson in a white t-shirt and black vest, holding a gun, standing in front of a large, ancient stone structure. The title 'DWAYNE JOHNSON RAMPAGE' is at the bottom left. The right side of the banner has a red background with white text.

PG IN CINEMAS MARCH 29

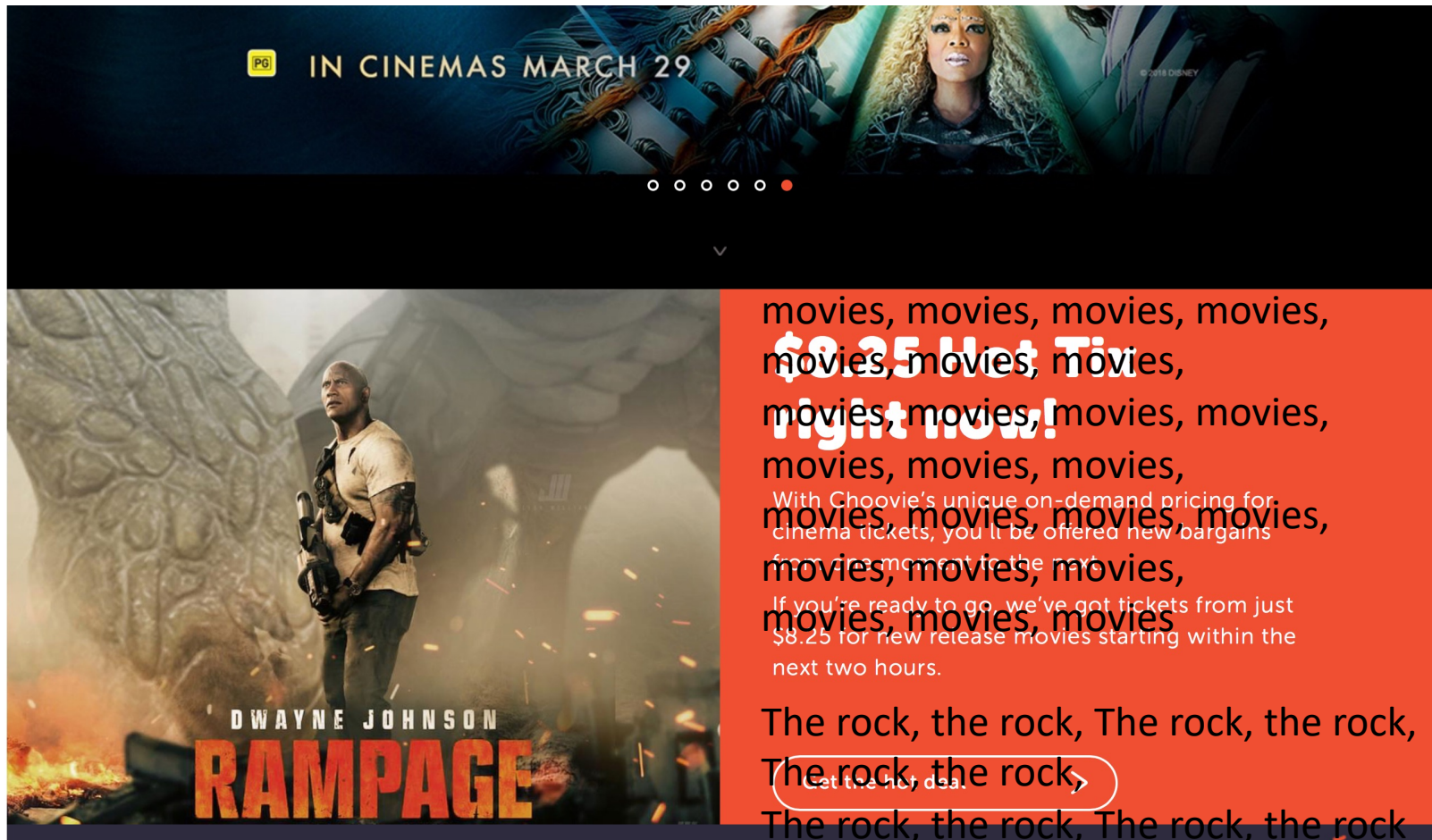
**\$8.25 Hot Tix
right now!**

With Choovie's unique on-demand pricing for cinema tickets, you'll be offered new bargains from one moment to the next.

If you're ready to go, we've got tickets from just \$8.25 for new release movies starting within the next two hours.

Get the hot deal >

Now highly ranked, but would you buy tickets from it?



Set the text colour same as the background so human can't see

PG IN CINEMAS MARCH 29

DWAYNE JOHNSON
RAMPAGE

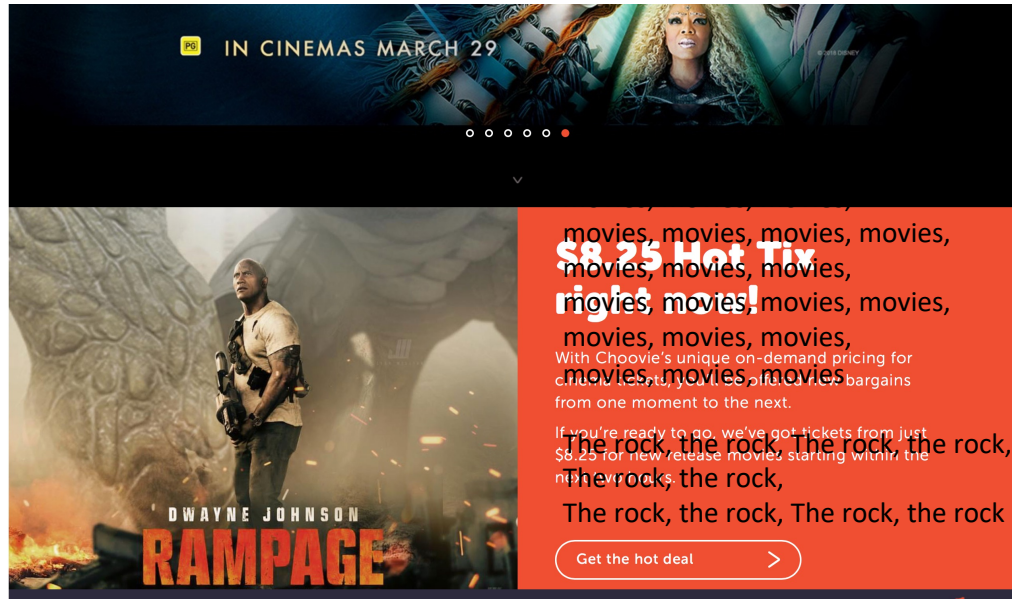
**\$8.25 Hot Tix
right now!**

With Choovie's unique on-demand pricing for cinema tickets, you'll be offered new bargains from one moment to the next.

If you're ready to go, we've got tickets from just \$8.25 for new release movies starting within the next two hours.

Get the hot deal >

Boosting web ranking like this is called



Term Spam

- Spammers make most search engine useless
- Killed many search engine companies

This guy saved search engine with an algorithm



Larry Page

Chief Executive Officer of Alphabet



Lawrence Edward Page is an American computer scientist and Internet entrepreneur who co-founded Google with Sergey Brin. Page is the chief executive officer of Google's parent company, Alphabet Inc. [Wikipedia](#)

Born: 26 March 1973 (age 45 years), [East Lansing, Michigan, United States](#)

Height: 1.81 m

Net worth: 48.3 billion USD (2018) [Forbes](#)

Spouse: [Lucinda Southworth](#) (m. 2007)

Education: [East Lansing High School](#) (1987–1991), [MORE](#)

Called PageRank

Lessons from term spam

- Given a search query (which consists of query terms), the relevance of a website should not purely or mainly be based on the 'matching' terms on the website, because the owner of the website can easily manipulate the terms on his/her website to fool the search engine.
- It is a lot harder for a fraudster to manipulate the websites which he/she does not own.

Key observations inspired PageRank

- Google uses PageRank among other things
- **PageRank idea**: Can we rank a website without using the terms at all?
- **Yes**, based on two observations:
- Observation 1: non-evil website owners often **place links** on their websites to **pages that they think are good and useful**.
- Observation 2: random web surfer indicates which pages a surfer/user is **likely to be visiting**, and **this purely depends on the links (not the terms)**
- Ob1+Ob2 => **random web surfers is more likely to visit useful pages.**

PageRank

- PageRank is a function that assigns a real number to each page that has been crawled and whose links have been discovered.
 - $F(\text{web_1}) = 0.01$, $F(\text{web_2}) = 0.0002$, ...
- **Intention**: The higher the PageRank the more important the page.
- There is no fixed algorithm for assigning the PageRank.
- Variation of basic idea can lead to different rankings.

PageRank – Basic Idea

- Think of the Web as directed graph
 - Pages are nodes
 - There is an arc from page p_i to p_j if there is at least one link from p_i to p_j

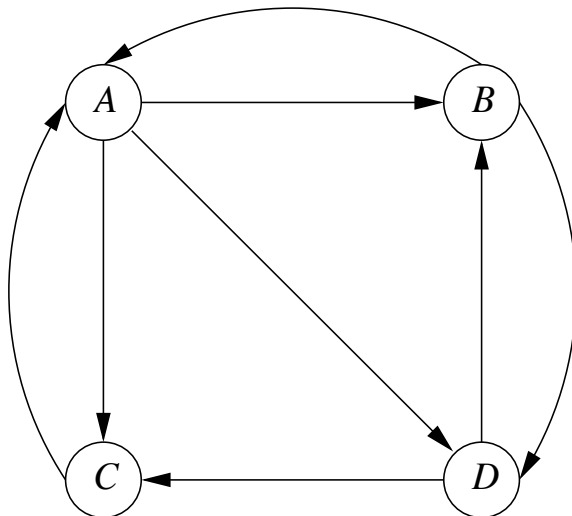
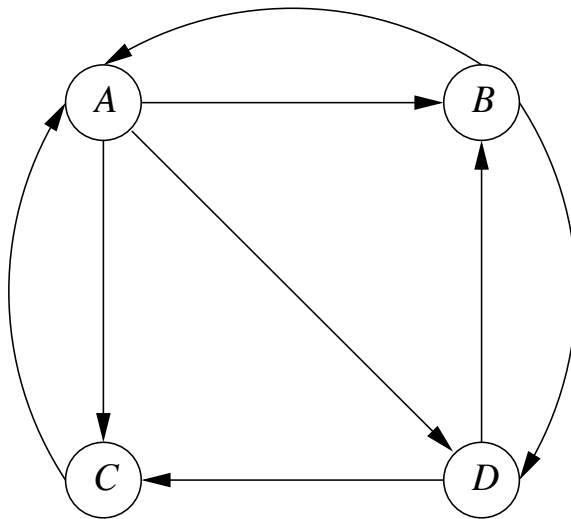


Figure 5.1 in Rajaramam/Ullman

Random Surfer

- PageRank of a page is a random surfer on the Web.
- Random Surfer moves at each step from his current page to a (**uniformly**) randomly chosen page that it links to.
- We can associate with a Web graph its transition probability matrix



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

Transition Matrix

- The transition matrix M of the Web describes what happens to a random surfer after one step.
- For n pages, this matrix $M=(m_{ij})$ has dimension $n \times n$.
- We have $m_{ij}=1/k$ if there is a link from j to i and the total number of outgoing links from j is k .
- If there is no link from j to i , we have $m_{ij}=0$.

Random Surfer

- The probability distribution of the location of a random surfer can be described by a vector of length n , where the j th component is the probability that the surfer is at page j .
- The probability is the (idealized) PageRank function.
- Suppose that we start a random surfer at any of the n pages
- The initial vector $v_0 = (1/n, \dots, 1/n)$ has $1/n$ on each of the components.
- If the current distribution is v , then $x = Mv$ will be the distribution of the next step.
 - The distribution will be Mv_0 after one step.
 - After two steps the distribution will be $M(Mv_0)$ and so on.
- Behavior is an example of a Markov process.

Limiting Distribution

- If the graph is strongly connected (possible to get from any node to any other node) then the surfer approaches the **limiting distribution** $v=Mv$.
 - the **limiting distribution** is also known as **stationary distribution**, **equilibrium distribution**
- The limiting v is an eigenvector of M , i.e. $v=\lambda Mv$ for some constant eigenvalue λ . **Why?**
 - As M is stochastic (columns adding up to 1), v is the principal eigenvector with eigenvalue 1.
 - The principal eigenvector tells us where the surfer is most likely after a long time (intuition of PageRank).
- M is extremely sparse: a tiny fraction of the entries are non-zero
 - Computing the principal eigenvector by starting with an initial vector v and multiplying by M some number of times until it shows little change at each round (in practice 50-75 iterations are enough for the Web) is a lot faster than doing

How to compute the limiting (stationary) v ?

- Method 1: $v = Mv$ is essentially a linear equation system, which may suggest to use Gaussian Elimination whose complexity is cubic to the number of equations.
 - There can be billions of nodes
- Method 2: $v_1 = M^*v_0, v_2 = M^*v_1, \dots, v_n = M^*v_{(n-1)}$.
 - Even that iteration is quadratic at each round, but we can speed it up by taking advantage of the fact that the matrix M is very sparse; there are on average about ten links per page, i.e., ten nonzero entries per column.

Example

Suppose we apply the process to

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

Then we the following sequence of approximations

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \begin{bmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix} \begin{bmatrix} 15/48 \\ 11/48 \\ 11/48 \\ 11/48 \end{bmatrix} \begin{bmatrix} 11/32 \\ 7/32 \\ 7/32 \\ 7/32 \end{bmatrix} \cdots \begin{bmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{bmatrix}$$

Structure of the Web in real world

- Strongly connected component (SCC).
- In-component: pages that could reach the SCC, but that can not be reached from SCC
- Out-component: pages reachable from SCC, but that can't reach SCC.
- Trendrils: Pages reachable from the in(out)-component but not able to reach the in(out)-component.
- Tubes: pages reachable from in-component and able to reach out-component, but can't reach and not reachable from SCC.
- Isolated components: Can't reach and not reachable from SCC and in/out-components.

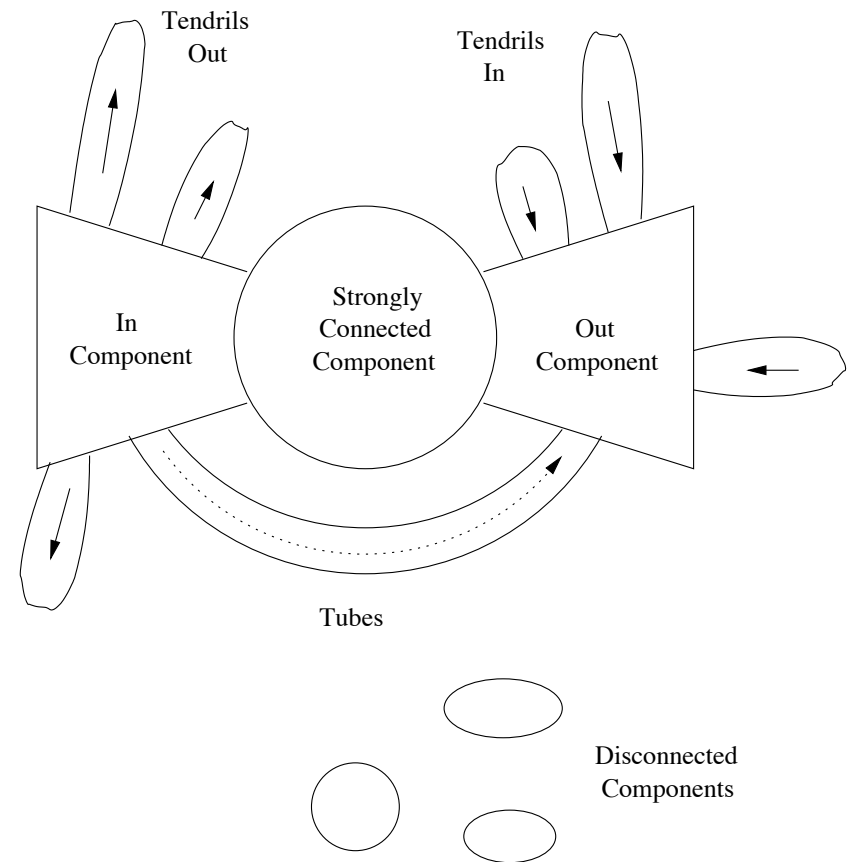


Figure 5.2 in Rajaraman/Ullman

Markov-Process

- Several structures violate the assumption needed for the Markov process iteration
- For example: When a random surfer enters the out-component, it can never leave.
- As a result, PageRank is usually modified to deal with these circumstances.

Two problems to be avoided:

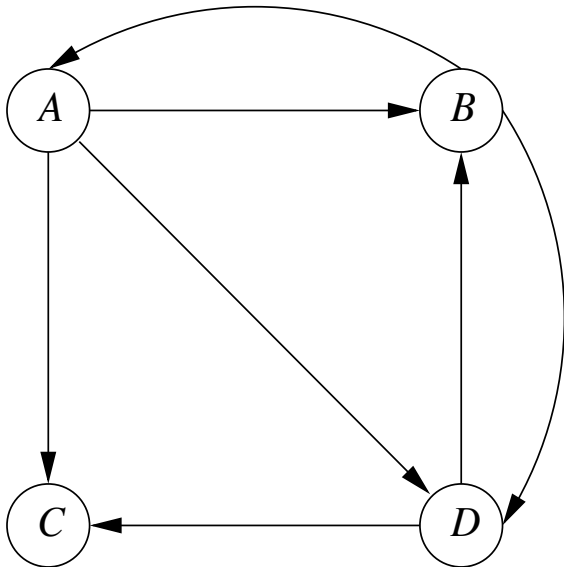
- Dead ends: pages that don't have a link out
- Spider traps: group of pages that all have outlinks, but never link to any other page.

Avoiding Dead Ends

- Page with no link out is a dead end.
- Transition matrix is no longer stochastic as some of the columns will sum to 0 (rather than 1).
- A matrix whose column sums are less than 1 is called substochastic.
- If we compute $M^i v$ for increasing powers of i of a substochastic matrix M , then some (or all) components of the vector go to 0.

Example

In our example we remove the link from C to A and C becomes a dead end.



Corresponding transition matrix

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

Figure 5.3 in Rajaraman/Ullman

Example

- Repeatedly multiplying the initial vector $(1/4, 1/4, 1/4, 1/4)$ by M gives the sequence

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

Dealing with dead-ends

Two ways of dealing with dead ends:

1. Drop dead ends from the graph and their incoming arcs.
Might create new dead end which are dropped recursively.
Recursive deletion of dead ends leads to a graph consisting of SCC, in-component and parts of isolated components.
2. Modify process of random surfer by “taxation”. This also allows to deal with spider traps.

PageRank after Recursive Deletion

- After recursive deletion of dead ends, we can solve the remaining graph G .
- Afterwards, we restore the graph, but keep the PageRank value for the nodes of G .
- Nodes not in G , but with predecessors all in G have their PageRank computed by summing, over all predecessors p , the PageRank p divided by the number of successors of p in the full graph.
- Process is iterated until PageRank values for all nodes of the original graph have been obtained.

Example

Input graph:

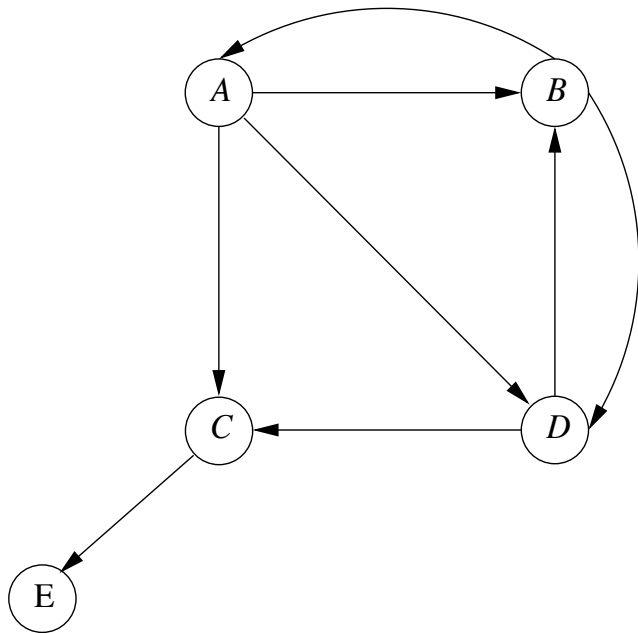


Figure 5.4 in Rajaraman/Ullman

After recursive dead end removal:

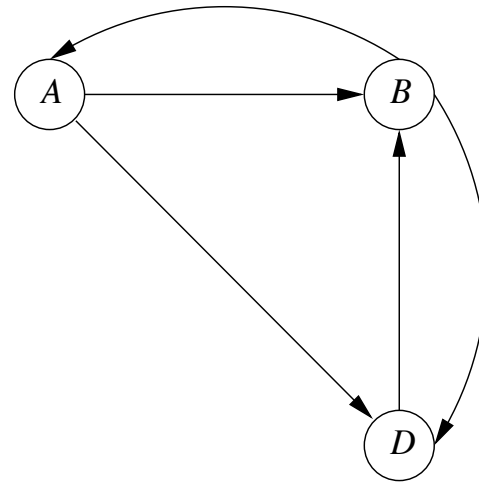


Figure 5.5 in Rajaraman/Ullman

Example

Matrix of graph given in Fig 5.5 (consisting of nodes A, B, D) is

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

For the sequence of vectors, we get

$$\begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \begin{bmatrix} 1/6 \\ 3/6 \\ 2/6 \end{bmatrix} \begin{bmatrix} 3/12 \\ 5/12 \\ 4/12 \end{bmatrix} \begin{bmatrix} 5/24 \\ 11/24 \\ 8/24 \end{bmatrix} \cdots \begin{bmatrix} 2/9 \\ 4/9 \\ 3/9 \end{bmatrix}$$

PageRank of A is 2/9, of B is 4/9, of D is 3/9.

Example

- C was the last node to be deleted and we have the PageRank for all of its predecessors (A, D)
- A has three successors and D has 2 successors
- Implies that A contributes $1/3$ of its PageRank and D contributes $1/2$ of its PageRank to C
- PageRank of C becomes

$$\frac{1}{3} \times \frac{2}{9} + \frac{1}{2} \times \frac{3}{9} = 13/54.$$

Example

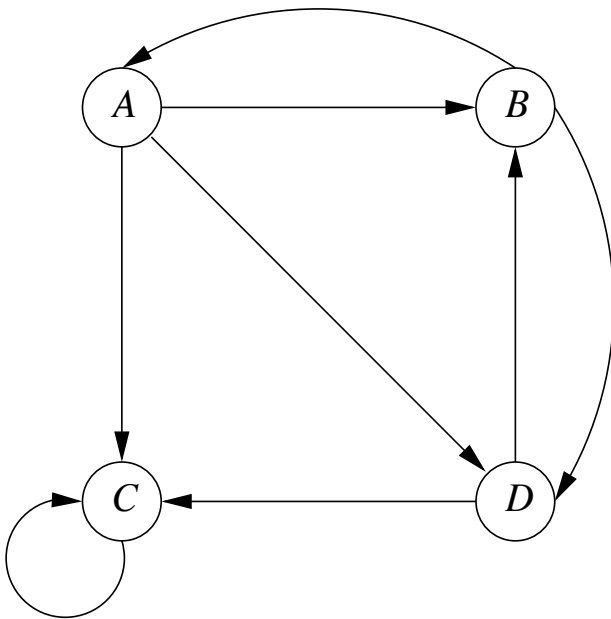
- Remains to compute PageRank of E (single predecessor node C)
- C has only one successor and PageRank of E is the same as for C: $13/54$.
- Note that the sum of PageRanks now exceeds 1 as we have $(2/9, 4/9, 13/54, 3/9, 13/54)$ for the distribution vector.
- No longer a probability distribution. However the numbers represent a decent estimate on the importance of pages.

Spider Traps and Taxation

- A spider trap is a set of nodes with no dead ends but no arcs out.
- These structures cause the PageRank calculation to place all PageRank in the spider trap.

Example

- Graph with node C has spider trap.



Corresponding transition matrix

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

Figure 5.6 in Rajaraman/Ullman

Example

Repeatedly multiplying the initial vector $(1/4, 1/4, 1/4, 1/4)$ by M gives the sequence

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

Intuition: Random Surfer gets trapped at node C.

Dealing with Spider Traps

- To deal with spider traps, we modify the PageRank calculation by allowing each random surfer with a small probability to teleport to a random page.
- This allows to leave spider traps.
- The step to calculate the new vector \mathbf{v}' from \mathbf{v} is now:

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

- \mathbf{e} is a vector of all 1s.
- Parameter β usually in the range 0.8 to 0.9 determines the trade-off between standard PageRank computation and teleporting to a random page.

Example

- Setting $\beta=0.8$ the equation for the graph in Figure 5.6 becomes

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

- For the first few iterations, we get

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

- Spider trap C still has more than $\frac{1}{2}$ of the PageRank, but each of the nodes gets some PageRank now.

How to use PageRank in a Search Engine?

- Each search engine has a secret formula that decides in which order to show pages as a response to a search query.
- Google is said to use over 250 different properties of pages to form such a linear order of pages.

How to use PageRank in a Search Engine?

- In order to be considered for the ranking at all, the page has to have at least one of the search terms in the query.
- Normally, all terms have to be present for a page to be in the top ten.
- Among the qualified pages, a score is computed for each of them and the PageRank is an important component.
- Other components include the presence of search terms in prominent places (e.g. headers of links to page itself).

Efficient Computation of PageRank

- To compute PageRank, we have to carry out 50-75 large matrix-vector multiplications to get a good approximation.
- We can use map-reduce to do this (see Chapter2).

We have to deal with two issues:

- Transition matrix M is very sparse. It's more effective to represent the matrix by its non-zero elements instead of storing the whole matrix.
- We want to reduce the amount of data that must be passed from the Map to the Reduce tasks.

Representing Transition Matrices

- The average Web page has about 10 out-links.
- Assume that we analyze graphs of ten billion pages (nodes).
- Transition matrix is very sparse.

Representation for sparse matrix:

- List locations of nonzero entries and their values.
- For each column the nonzero values are the same
- We list nonzero values by column and for each column the outdegree of its corresponding page.

Example

Transition matrix:

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

Resulting representation:

Source	Degree	Destinations
<i>A</i>	3	<i>B, C, D</i>
<i>B</i>	2	<i>A, D</i>
<i>C</i>	1	<i>A</i>
<i>D</i>	2	<i>B, C</i>

PageRank Iteration using Map-Reduce

- For one iteration of PageRank, we have to compute

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

- If the number of nodes n is small enough such that \mathbf{v} fits into main memory then result \mathbf{v}' also fits into main memory (standard matrix vector multiplication weighted by β and adding additional terms)
- Given the size of the Web, we assume that \mathbf{v} doesn't fit into the main memory.
- We can break M into vertical strips and \mathbf{v} into horizontal stripes and apply the approach from Chapter 2 to execute Map-Reduce efficiently.

PageRank without the matrix M

A better way: just use the nodes (of the graph), ignore the matrix M.

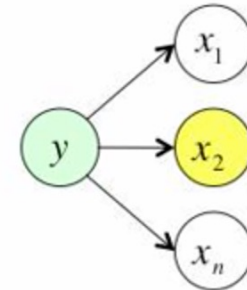
- Info to begin with:
 - We need to know the total number of nodes N.
 - For each node, the in-links and out-links
- For each node x, store its PageRank PR(x), and its out-link number Out(x) (i.e. the number of out-links of x)
- Initialize the PR(x) i.e. $PR(x) = 1/N$.
- Update PR for each x

$$PR(x) = \frac{(1-\beta)}{N} + \sum_{y \rightarrow x} \frac{PR(y)}{Out(y)}$$

PageRank with Map-Reduce (no matrix)

- **Mapper:** $\langle y, \{x_1 \dots x_n\} \rangle$ node + out-links

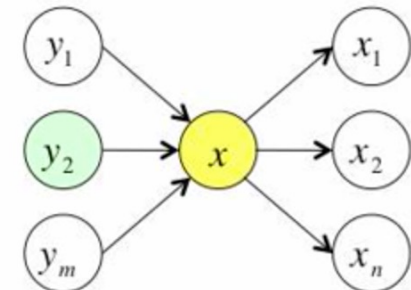
- for $j=1 \dots n$: **emit** $\langle x_j, \frac{PR(y)}{out(y)} \rangle$
- **emit** $\langle y, \{x_1 \dots x_n\} \rangle$



- **Reducer:** $\left\langle x, \left\{ \frac{PR(y_1)}{out(y_1)}, \dots, \frac{PR(y_m)}{out(y_m)}, \{x_1 \dots x_n\} \right\} \right\rangle$ node + ΔPR from in-links

- **compute:** $PR(x) = \frac{1 - \lambda}{N} + \lambda \sum_{y \rightarrow x} \frac{PR(y)}{out(y)}$

- for $j=1 \dots n$: **emit** $\langle x_j, \frac{PR(x)}{out(x)} \rangle$
- **emit** $\langle x, \{x_1 \dots x_n\} \rangle$



<https://www.youtube.com/watch?v=9e3geIYFOF4>