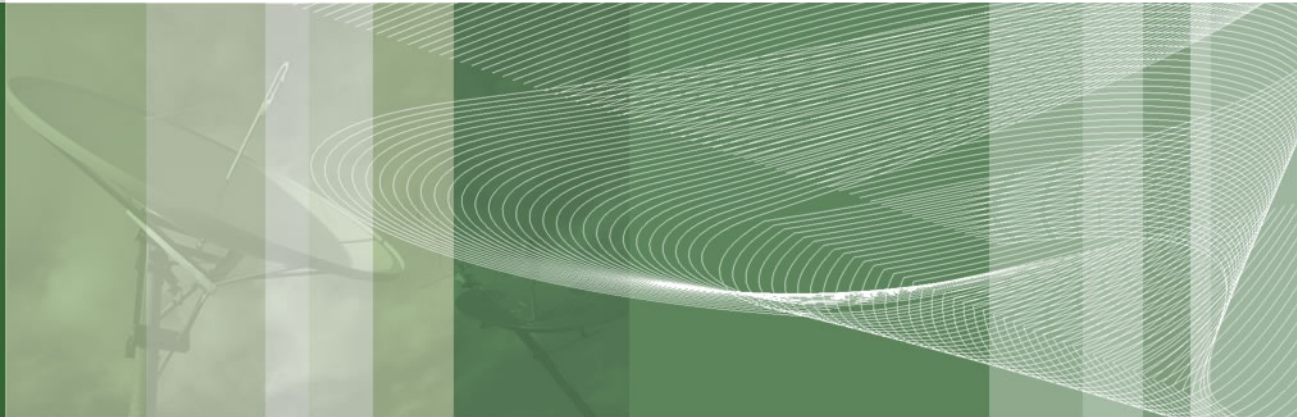


POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione



Search Computing

Web information retrieval

Stefano Ceri

Original material authored by Marco Tagliasacchi

- Search engine evolution
- Modeling the Web
- Web Crawling
- Link Analysis
- Page Rank
- HITS

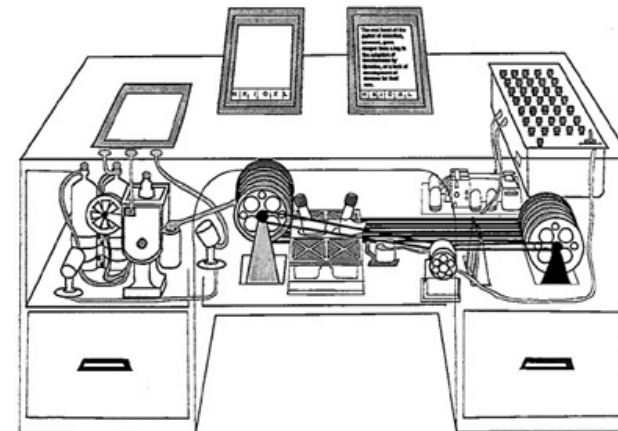
- Web search has become a standard (and often preferred) source of information finding
 - "... 92% of Internet users say the Internet is a good place to go for getting everyday information..."
2004 Pew Internet Survey
 - 7.2 billion Web searches/month
 - (3.9 billion by Google) far exceed the world population
 - By the end of 2008, 50% of applications include a search facility as a primary interface for end users

- The Web in 2006 has:
 - 109 million distinct web sites
 - 29.7 billion web pages
 - ~5 pages for every man, woman, and child on the planet
- 161 Exabyte (10^{18} TB) of information was created or replicated worldwide in 2006.
 - IDC estimates 6X growth by 2010 to 988 Exabyte (a zetabyte) / year
 - That's more than in the previous 5,000 years.
- The largest source of data are **USERS**
 - YouTube Videos
 - 1.7 billion served / month
 - 1 million streams / day
 - = 75 billion e-mails

Web search engines history

5

- The concept of hypertext and a memory extension really came to life in July of 1945, Vannevar Bush's "As we May Think - The Atlantic Monthly"
 - "The summation of human experience is being expanded at a prodigious rate. **Our ineptitude in getting at the record is largely caused by the artificiality of the systems of indexing.** ... Having found one item, moreover, one has to emerge from the system and re-enter on a new path. **The human mind does not work this way. It operates by association.**"
 - He then proposed the idea of a virtually limitless, fast, reliable, extensible, associative memory storage and retrieval system. He named this device a memex.



Web search engines history

- **Gerard Salton (1960s - 1990s)** developed the SMART informational retrieval system. Salton's Magic Automatic Retriever of Text included important concepts like the vector space model, Inverse Document Frequency (IDF), Term Frequency (TF), term discrimination values, and relevancy feedback mechanisms.
- **Ted Nelson** created Project Xanadu in 1960 and coined the term hypertext in 1963. His goal with Project Xanadu was to create a computer network with a simple user interface that solved many social problems.
- The first search engine created was **Archie**, created in 1990 by Alan Emtage, a student at McGill University in Montreal. Archie helped combined a data gatherer with a regular expression matcher for retrieving file names matching a user query. Essentially Archie was a database of web filenames which it would match with the users queries.

▪

Welcome to **archie.icm.edu.pl**

Archie Query Form



Search for:

Database: ☐ Worldwide Anonymous FTP ☐ Polish Web Index
Search Type: ☐ Sub String ☐ Exact ☐ Regular Expression
Case: ☐ Insensitive ☐ Sensitive

Do you want to look up strings only (no sites returned):

☐ NO ☐ YES

Output Format For Web Index Search: ☐ Keywords Only

☐ Excerpts Only

☐ Links Only

Tim Berners-Lee & the WWW

8

- While an independent contractor at CERN from June to December 1980, **Berners-Lee** proposed a project based on the concept of hypertext, to facilitate sharing and updating information among researchers. With help from **Robert Cailliau** he built a prototype system named Enquire.
- He returned in 1984 as a fellow. In 1989, CERN was the largest Internet node in Europe, and Berners-Lee saw an opportunity to join hypertext with the Internet. In his words, *"I just had to take the hypertext idea and connect it to the TCP and DNS ideas and — ta-da! — the World Wide Web"*.
- The first Web site built was at <http://info.cern.ch/> and was first put online on August 6, 1991. It provided an explanation about what the World Wide Web was, how one could own a browser and how to set up a Web server. It was also the world's first Web directory, since Berners-Lee maintained a list of other Web sites apart from his own.
- In 1994, Berners-Lee founded the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology.

Web search engines history

9

- First generation (1994-1997, Excite, Lycos, etc.):
 - use only textual data on page (word frequency)

- Second generation (1998, Google):
 - use Web specific properties
 - Link analysis
 - Click-through data (what results people click on)
 - Anchor text (how people refer to linked pages)

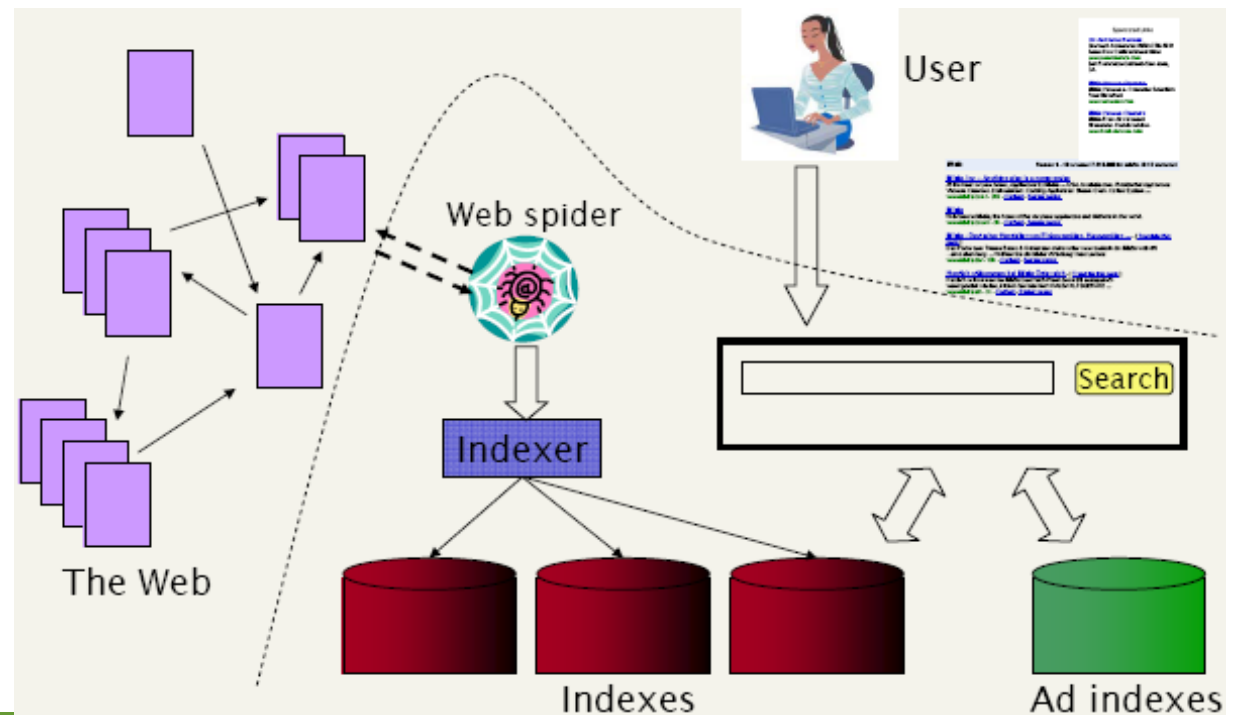
- Third generation
 - Semantic analysis
 - Focus on the user
 - Context-dependent
 - User interaction
 - multimedia data

<http://www.webhostingbuzz.com/blog/2011/06/13/search-engines-evolution/>

Web search engine architecture

10

- Web Search engines aren't just black boxes
 - Crawling document collections
 - Building indexes
 - Ranking
 - Serving search results
 - User interface and design
 - Infrastructure



- Search engine evolution
- Modeling the Web
- Web Crawling
- Link Analysis
- Page Rank
- HITS
- Extensions to Link Analysis

- The network of web pages and links define a directed graph
 - web pages define the **nodes** $1, \dots, n$
 - hyperlinks define the **edges**
- The graph is described by an $n \times n$ **adjacency matrix** **E**
 - $E_{i,j} = 1$ if there is a link from i to j
 - $E_{i,j} = 0$ otherwise
- Definitions:
 - **indegree** of node i : number of edges coming into i
 - **outdegree** of node i : number of edges coming out from i
 - A directed graph is **weakly connected** if any node can be reached from any other node by traversing edges either in their indicated direction or in the opposite direction

- **Random networks:**
 - start from a collection of n nodes
 - each of the $n(n-1)$ potential edges materialized with a fixed probability
- Random graph models are not suitable for the Web
- **Scale-free network** model more suitable
- In the large, the Web graph can be modeled in terms of degree distribution which follows a **power law**

$$\Pr(\text{out-degree} = k) \propto 1 / k^{a_{out}}$$

$$\Pr(\text{in-degree} = k) \propto 1 / k^{a_{in}}$$



i.e. the probability that the out-degree (in-degree) is equal to k is inversely proportional to k^a , for some model parameter a

- Web graph grows by adding nodes
- **Preferential attachment**: a new node is linked to existing nodes with higher probability to existing nodes that already have a large degree
- This model predicts an exponent equal to 3. Exponents from Web measurements range between 2.1 and 2.5

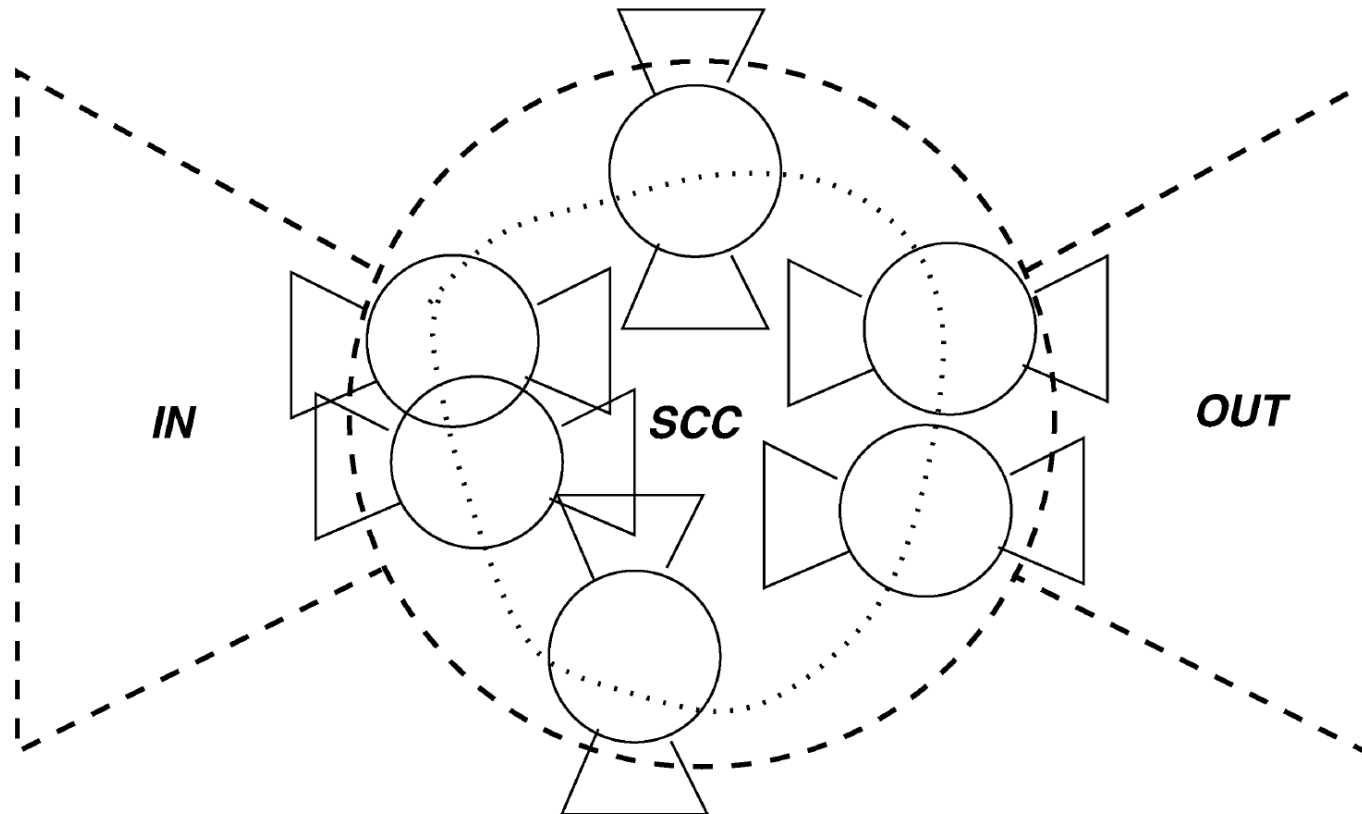
- A consequence of the scale invariance is that the structure of the Web is “fractal”

- A **thematically unified cluster** (TUC) is a cluster of Webpages that share a common trait:
 - By content
 - By domain
 - By geographic location
 - Etc.

- Thematically unified clusters display the same characteristics as the Web at large

Web self-similarity

“Bowtie” structure

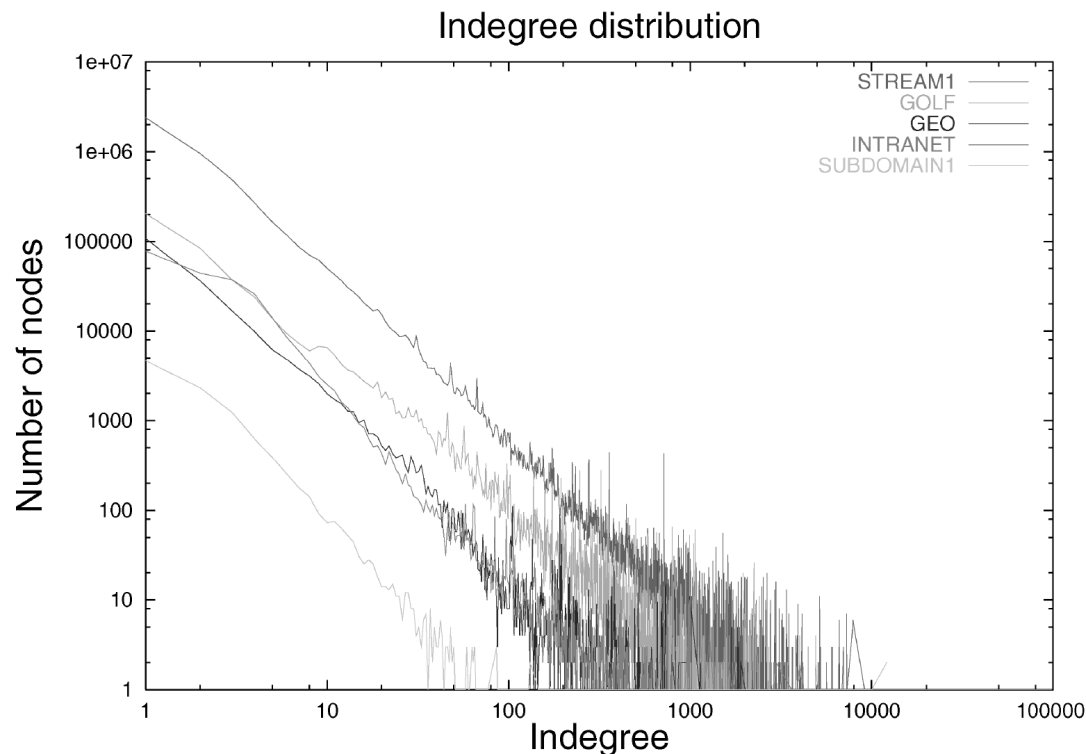


SCC: largest strongly connected component

OUT: components reached by the pages in SCC

IN: components that can reach pages in SCC

- Various parameters describing graph structure computed on TUCs are similar to each other and to those computed on the Web graph at large
 - In-degree distribution
 - Out-degree distribution
 - Connected component size



**Example of in-degree
distribution for different
TUCs**

Web self-similarity

- Web self-similarity allows simplification of data service design problems
 - Advanced Web search (**Link analysis**)
 - Data-driven approach for Web site generation
 - Web specific query languages
 - Community extraction
 - Taxonomy construction

- Search engine evolution
- Modeling the Web
- Web Crawling
- Link Analysis
- Page Rank
- HITS

- Web crawling is the process by which we gather pages from the Web
- Goal: quickly and efficiently gather as many useful Web pages as possible, together with the link structure that interconnects them

Web crawling requirements

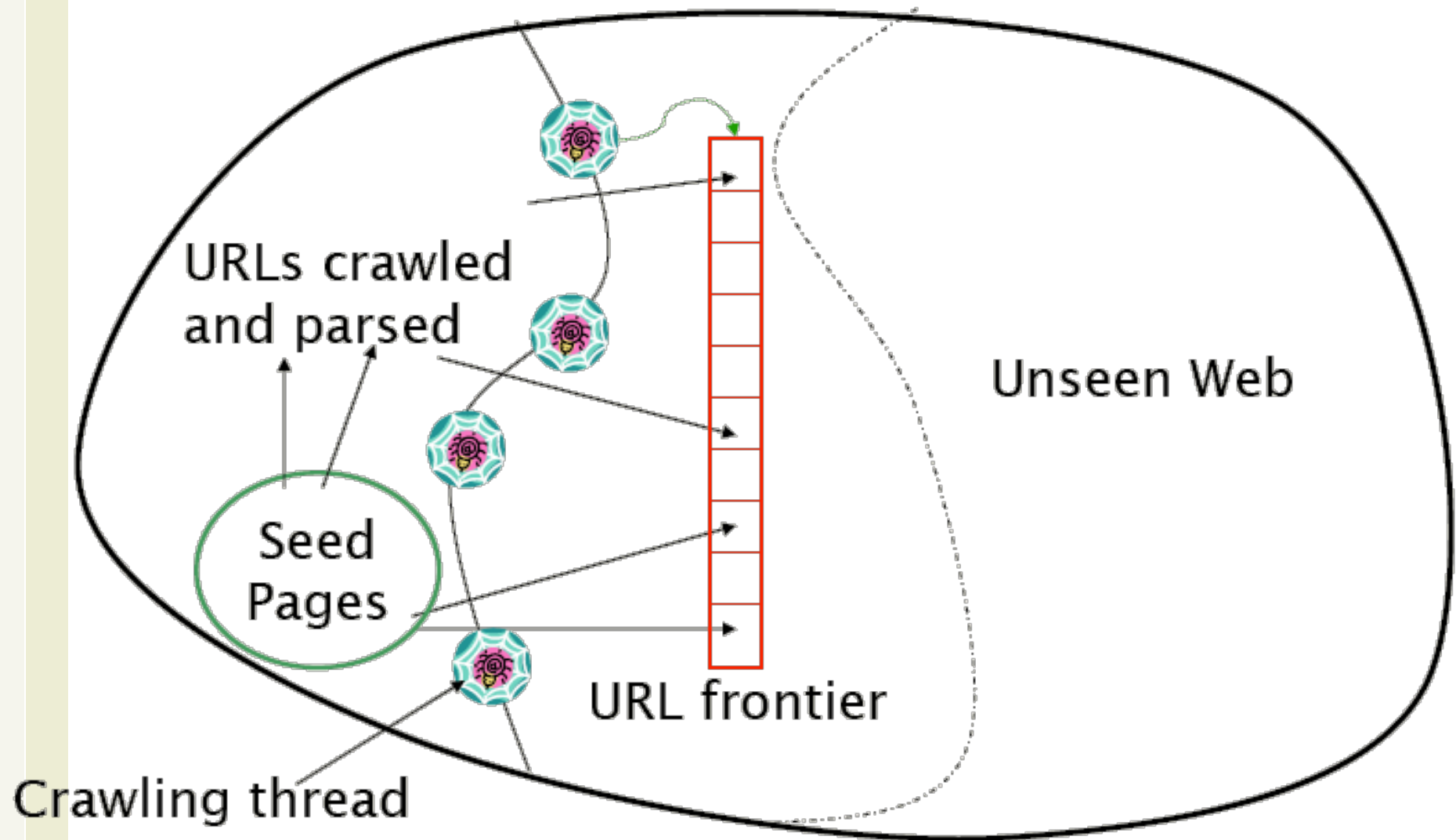
21

- **Robustness**: MUST avoid spider traps (fetching an infinite number of pages in a particular domain)
- **Politeness**: MUST respect Web servers policies, regulating the rate at which crawlers can visit them
- **Distributed processing**: SHOULD have the ability to execute in a distributed fashion across multiple machines
- **Scalability**: SHOULD scale up the crawling rate by adding extra machines and bandwidth
- **Performance and efficiency**: SHOULD make efficient use of system resources (processors, storage, network bandwidth)
- **Quality**: SHOULD fetch “useful” pages first
- **Freshness**: SHOULD operate in continuous mode

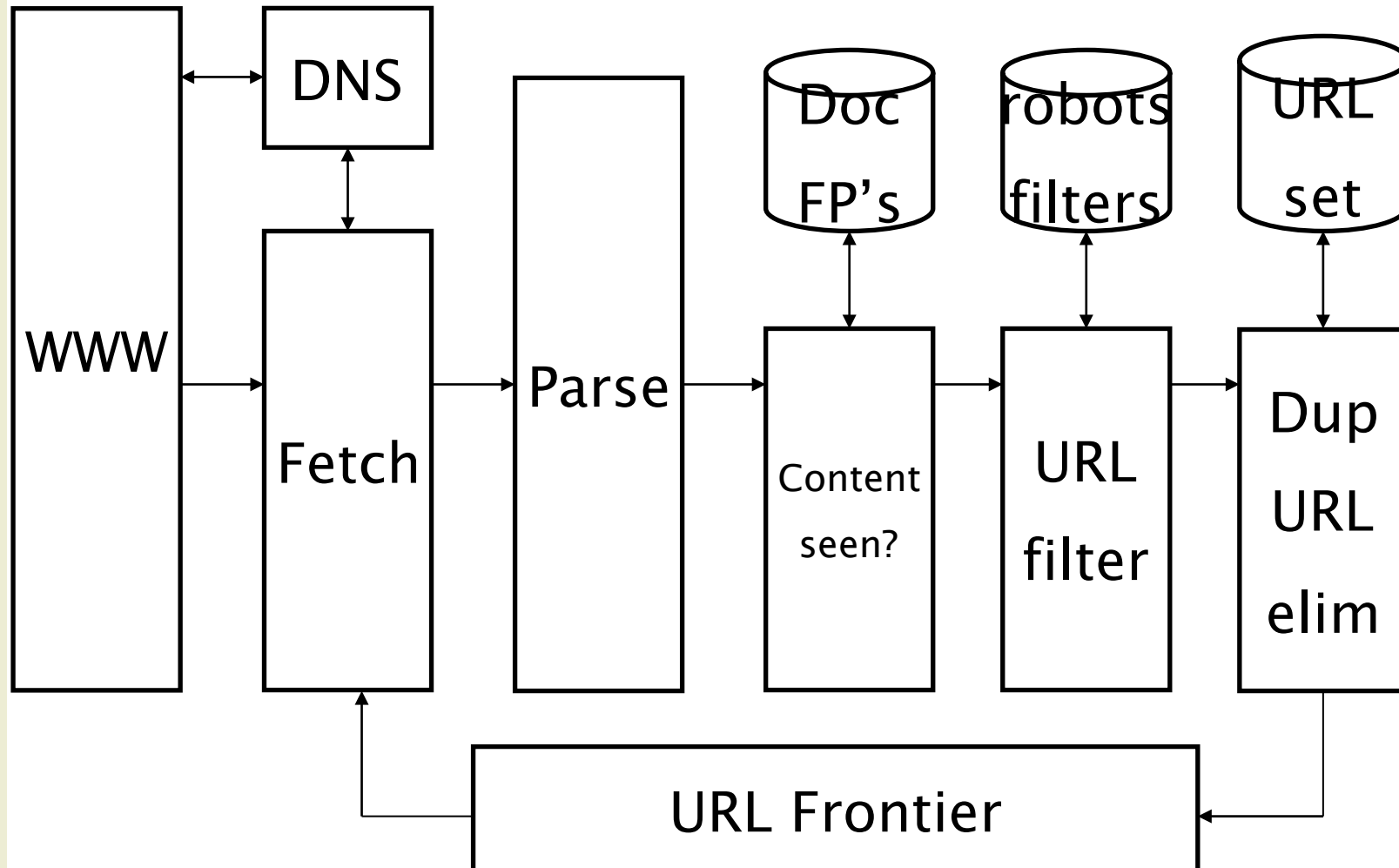
Web crawling process

22

1. Initialize URL frontier with a **seed set** of known URLs
2. Select a URL from the URL frontier
3. Connect the server and **fetch** the page pointed by the URL
4. Parse the retrieved HTML document
 1. **extract text**
 2. **detect hyperlinks**
5. Discard URLs that
 - can't be analyzed (e.g. .exe, .jpg, etc.)
 - have already been visited
6. Add filtered URLs to a URL frontier
7. Goto step 2



- Single node architecture
- Processing distributed to several threads

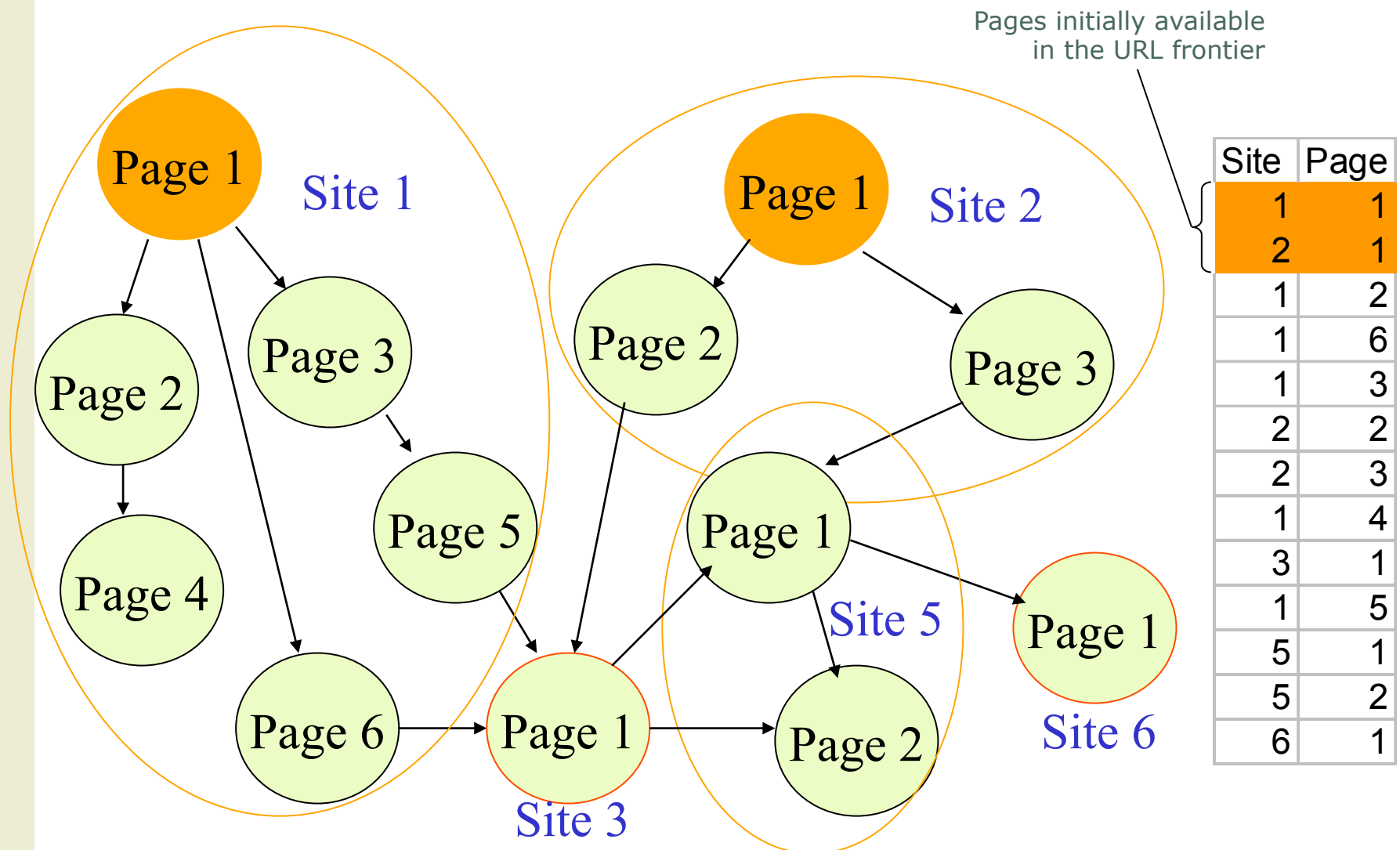


- **URL frontier**: containing URLs yet to be fetched in the current crawl
- **DNS resolution**: determines the Web server from which to obtain the URL
- **Fetch**: retrieves the Web page at a URL
- **Parsing**: extracts the text and the set of links from the fetched web pages
- **URL filtering**: determines if the URL can be crawled (checking the robots.txt constraints)
- **Duplicate elimination**: determines whether an extracted link is already in the URL frontier or has recently been fetched (e.g. by using document fingerprints – Doc FP's)

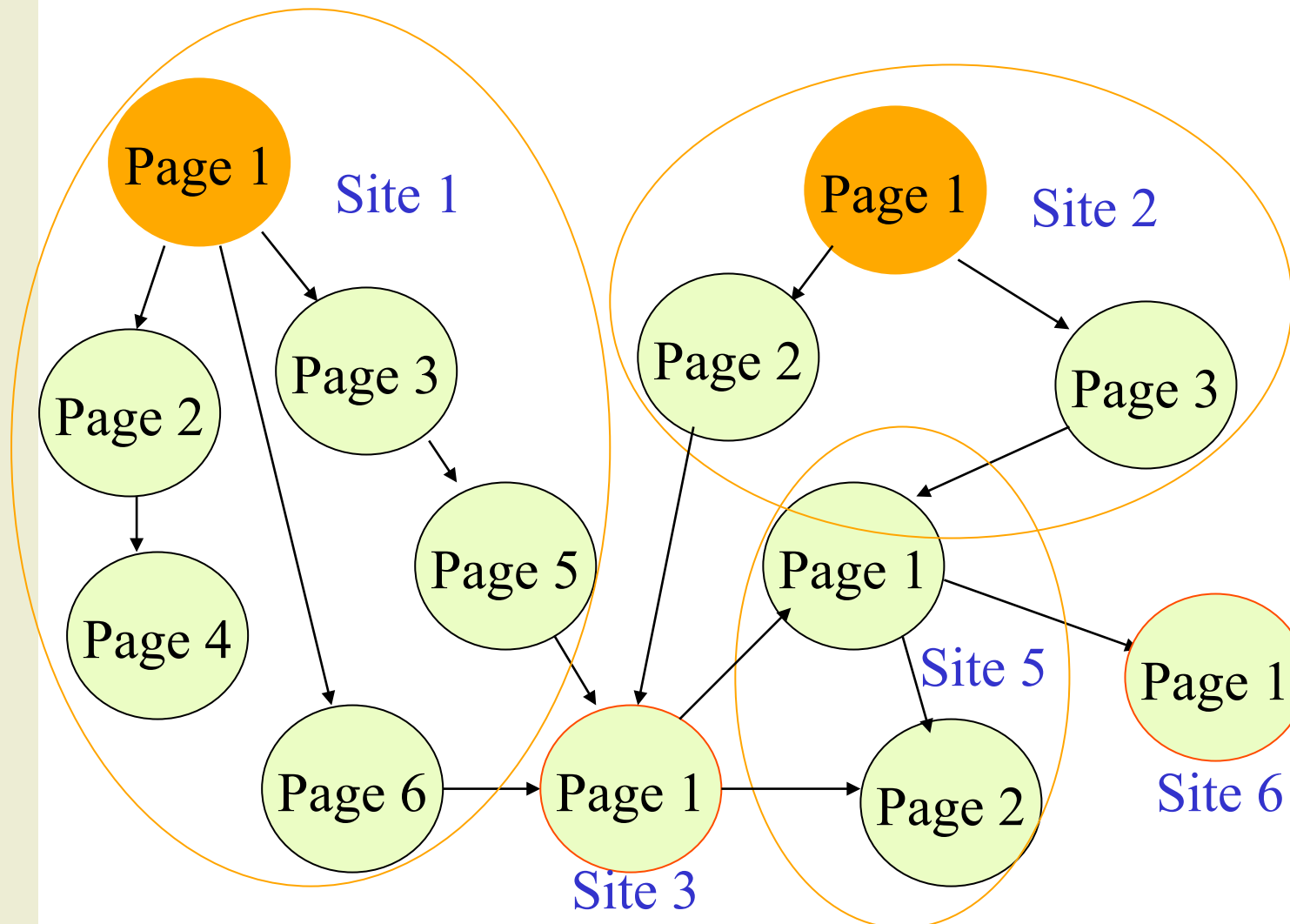
■ URL frontier

- Pages are **added** to the URL frontier according to the following strategies
- **Breath first** strategy: given a Web page in the URL frontier, add all pages linked by the current page. Coverage is wide but superficial
- **Depth first** strategy: given a Web page in the URL frontier, follow the first link in the current page until the first page without links

- URL frontier: Breath first strategy



- URL frontier: Depth first strategy



Site	Page
1	1
1	2
1	4
1	6
1	3
1	5
3	1
5	1
5	2
6	1
2	1
2	2
2	3

- URL frontier

- URLs are removed from the URL frontier and assigned to available crawling threads
- The frontier returns URLs in some order. The goal is to ensure:
 - implicit politeness
 - only one connection be open at a time to any host
 - a waiting time of a few seconds occurs between successive requests to a host
 - high priority pages (e.g. those that change frequently) are crawled preferentially

■ DNS resolution

- Each Web server has a unique IP address
- DNS resolution: translates URL to an IP address
- DNS resolution is a **bottleneck** in Web crawling
 - DNS servers are distributed
 - DNS resolution might entail multiple requests and round trips across the internet
- Introducing **DNS caching** helps

- URL filtering

- Many hosts place certain portions of their Web site off-limits to crawling
- Place **robot.txt** file in the root of the URL hierarchy
- robot.txt is compliant to the **Robot Exclusion Protocol**
- Expresses the request that specified Web crawlers (robots) ignore specified files or directories in their search
- For Web sites with multiple sub-domains, each subdomain must have its own robot.txt file
- The protocol is purely advisory. It relies on the cooperation of the crawler. Marking an area of a site out of bounds with robots.txt does not guarantee privacy

- URL filtering: Robot Exclusion Protocol examples
 - (to be inserted into the robot.txt text file present in the directory corresponding to the root URL):
 - allows all robots to visit all files
`User-agent: *`
`Disallow:`
 - blocks all robots from visiting any file
`User-agent: *`
`Disallow: /`
 - blocks all robots to visit files in the `cgi-bin` and `tmp` dir
`User-agent: *`
`Disallow: /cgi-bin/`
`Disallow: /tmp/`

- URL filtering: Robot Exclusion Protocol examples:

- no robot should visit any URL starting with `/yoursite/temp/`, except for the robot called `searchengine`

```
User-agent: *  
Disallow: /yoursite/temp  
User-agent: searchengine  
Disallow:
```

- the proposed Extended Robot Exclusion Protocol adds new directives

```
User-agent: * Disallow: /downloads/  
Request-rate: 1/5 # maximum one page every 5 seconds  
Visit-time: 0600-0845 # visit between 06:00 and 08:45
```

Web crawling architecture

- URL filtering: Robot Exclusion Protocol usage statistics
 - Approximately 22% of the web sites use robot.txt

Domain	%
com	23.69
Org	20.18
net	21.85
edu	25.72
gov	42.98
info	26.51
Total	22.41

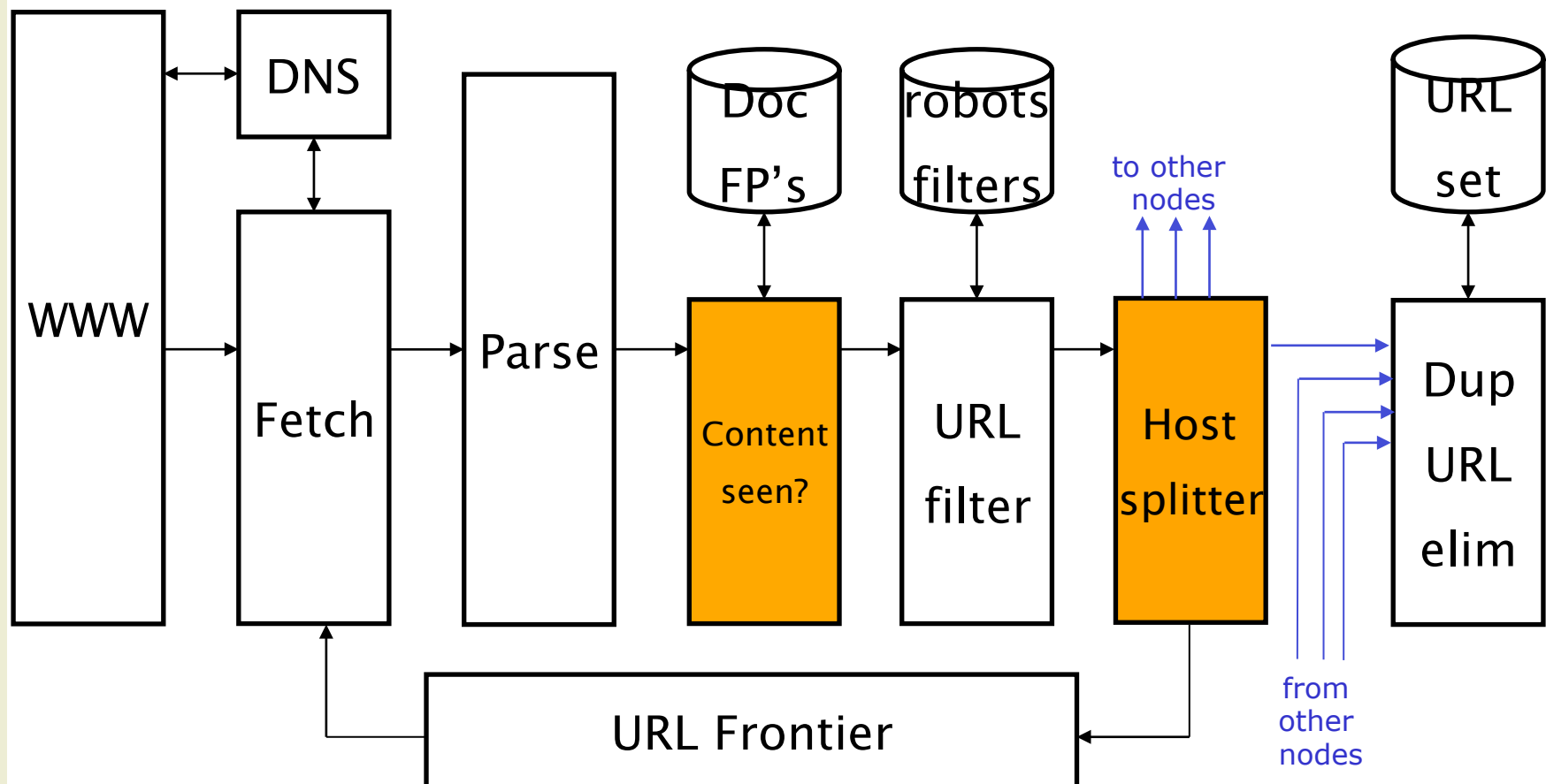
- Of them, approximately 20% contain syntax errors!

■ Duplicate elimination

- **Duplicated content** detection: the same Web page can be duplicated at different hosts (e.g. because of mirroring). Check whether the current content has already been seen at another URL
 - Document fingerprint (e.g. checksum)
- **Duplicated URL** detection: a URL is not added to the URL frontier if
 - it is already in the frontier
 - already crawled (for a non-continuous crawl)

- Crawling is performed by one to potentially hundreds of threads
- Threads may be run in a single process or be partitioned amongst multiple processes running at different nodes in a distributed system
- Distribution of crawling is essential for scaling

- Distributed architecture
- Each process (multi-thread) executed at a single node



- Each crawling process maintains its **URL frontier**
- **Host splitter** despatches surviving URLs after URL filtering to a given crawler node to ensure node balancing
- **Partitioning** the hosts being crawled amongst the crawler nodes can be done by
 - hash function
 - geographic proximity
 - ...
- In order to eliminate duplicated URLs, the local URL set repository is complemented with information **shared** by the other distributed crawling nodes

- The **content seen** module in the distributed architecture is complicated by the following factors:
 - Content fingerprints cannot be partitioned based on host name.
 - The same content can appear on different hosts.
 - The set of fingerprints must be partitioned across the nodes
 - This results in a locality mismatch: most “content seen” tests result in a RPC
 - Little locality in the streams of document fingerprints. Thus caching does not help
 - Documents change over time. It is necessary to save document fingerprint in the URL frontier, along with the URL itself

- Search engine evolution
- Modeling the Web
- Web Crawling
- Link Analysis
- Page Rank
- HITS

The World Wide Web seen as a (directed) graph:

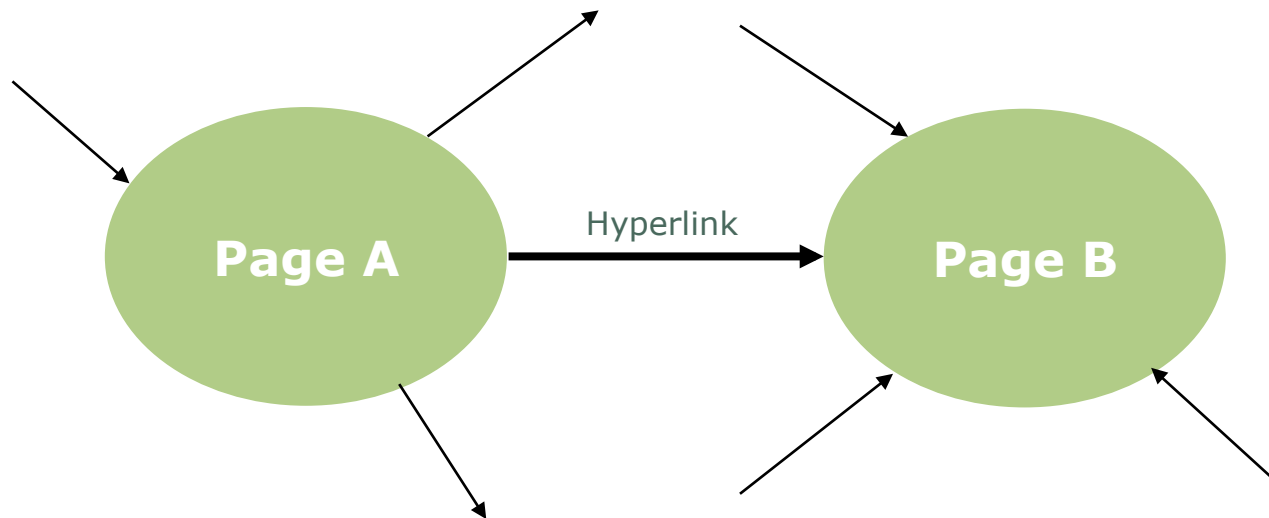
Vertices: Web pages

Edges: hyperlinks

Goal: use of hyperlinks to rank Web search results

Same for other interlinked environments:

- Dictionaries
- Encyclopedias
- Scientific publications
- Social networks



Intuitions

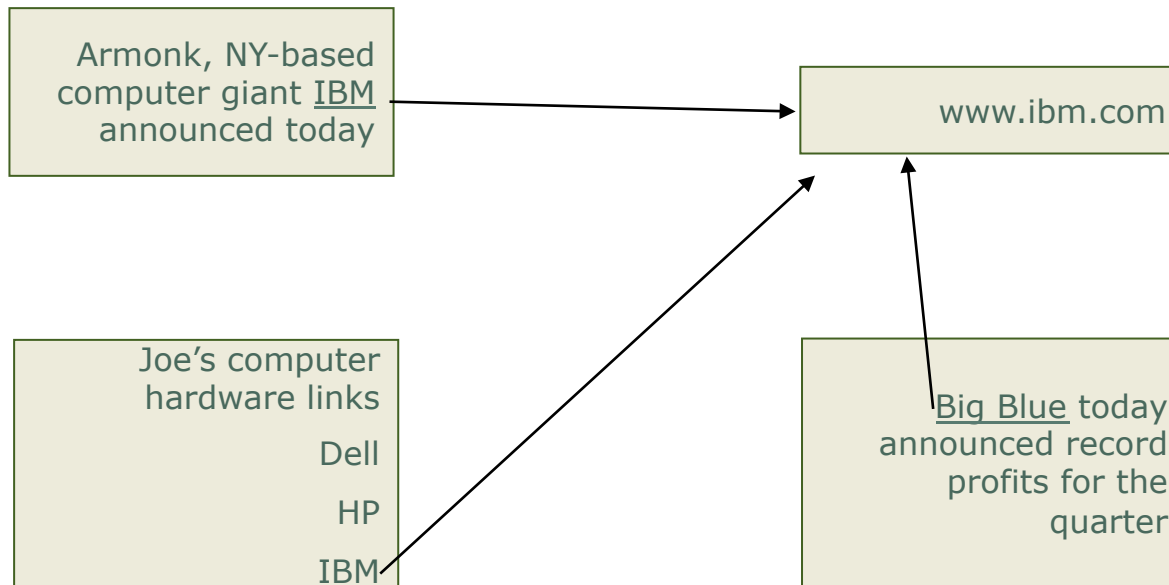
- The **anchor text** might describe the target page B
 - Anchor text indexing
- The hyperlink from A to B connotes a conferral of **authority** on page B, by the creator of page A
 - Link based ranking

Issues of conventional inverted index search:

- Several instances where page B does not provide a description of itself
 - <http://www.ibm.com> page does not contain “computer”
- Gap between how a page presents itself and how web users would describe it (e.g. Big Blue → IBM)
- Many pages embed text in graphics and images, making HTML parsing inefficient

Solution:

- Include anchor text terms in **inverted indexing**
- **Weight anchor text** terms based on frequency (to penalize words such as “click” or “here”)



- Can sometimes have unexpected **side effects**, e.g.
 - searching for big blue on some search engines returns the home page of IBM
 - anchor text such as “evil empire” used to lead to unexpected results. Exploited in orchestrated campaigns against specific Web sites
- **Anchor text** and **body text** are jointly used to compute the relevance of a Web page with respect to a given query. Can use index anchor text with less weight to limit the aforementioned side-effects

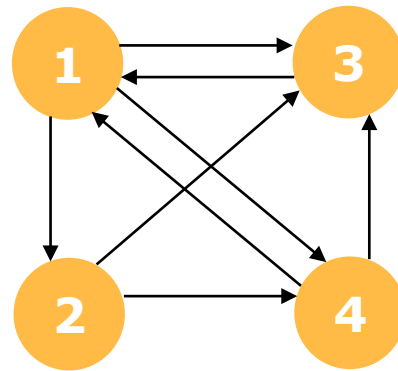
- Compute the **authoritativeness** exploiting the hyperlink structure of the Web
- The Web can be seen as a network of recommendations, a **social network**
- Web search engines of the second generation identify relevance combining
 - **topic-relatedness** (boolean, vector, probabilistic models)
 - **authoritativeness** (link base ranking)

- Link-based ranking systems rank a **base set** of web pages. Depending on the base set we have:
 - **Query dependent** systems: rank a set of Web pages that have been identified as topic-related
 - based on both topic relatedness and authoritativeness
 - must be computed online at query time
 - best known algorithm: **HITS**
 - **Query independent systems**: rank the entire Web
 - only based on authoritativeness
 - can be computed offline
 - At query time, it must be merged in some way with a query dependent ranking based on topic-relatedness
 - Best known algorithm: Google **PageRank**

- Search engine evolution
- Modeling the Web
- Web Crawling
- Link Analysis
- Page Rank
- HITS

- Any Web page is assigned a **non-negative score**
- A page score is derived from links made to that page from other web pages
- The links to a given page are called **backlinks**
- Web democracy: pages vote for the importance of other pages by linking to them

A simple approach: count the **number of backlinks**



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

- Issue: ignores the fact that a link to page j from an important page should boost page j 's importance score more than a link from an unimportant one
 - For example, a link to your homepage from www.yahoo.com should boost your page's score more than a link from www.foobar.com

- A simple solution:
 - Compute score of page j as the sum of scores of all pages linking to page j
 - e.g. $x_1 = x_3 + x_4$
 - Does not work: a single page gains influence by linking to lots of other pages
- A better solution:
 - If page j contains n_j links, one of which to page k , we will boost page k 's score by x_j/n_j instead than by x_j
 - Each page gets **one vote**, which is **divided** up among its outgoing links

$$x_k = \sum_{j \in L_k} \frac{x_j}{n_j}$$

L_k is the set of page k 's backlinks

n_j is the number of page j 's outgoing links

PageRank Example

52

$$x_1 = \frac{x_3}{1} + \frac{x_4}{2}$$

$$x_2 = \frac{x_1}{3}$$

$$x_3 = \frac{x_1}{3} + \frac{x_2}{2} + \frac{x_4}{2}$$

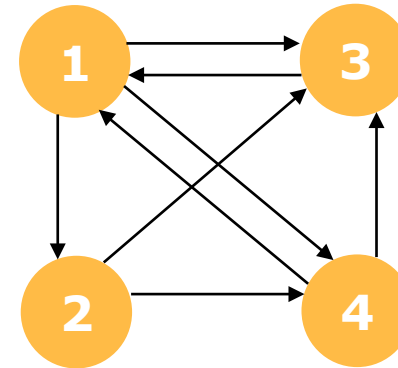
$$x_4 = \frac{x_1}{3} + \frac{x_2}{2}$$



$$\mathbf{A}\mathbf{x} = \mathbf{x}$$

where

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



- The score can be obtained by solving the system

$$\mathbf{Ax} = \mathbf{x}$$

$\mathbf{A}_{ij} = 0$ if there is no link between page j and i

$\mathbf{A}_{ij} = \frac{1}{n_j}$ otherwise, with n_j the number of outgoing links of page j

i.e. finding one eigenvector corresponding to the eigenvalue 1

The matrix \mathbf{A} is the transpose of the (weighted) adjacency matrix of the Web graph

- The score can be obtained by the power iteration method

$$\mathbf{x} = \lim_{k \rightarrow \infty} \mathbf{A}^k \mathbf{x}_0$$

where \mathbf{x}_0 is some initial column vector with non-zero entries

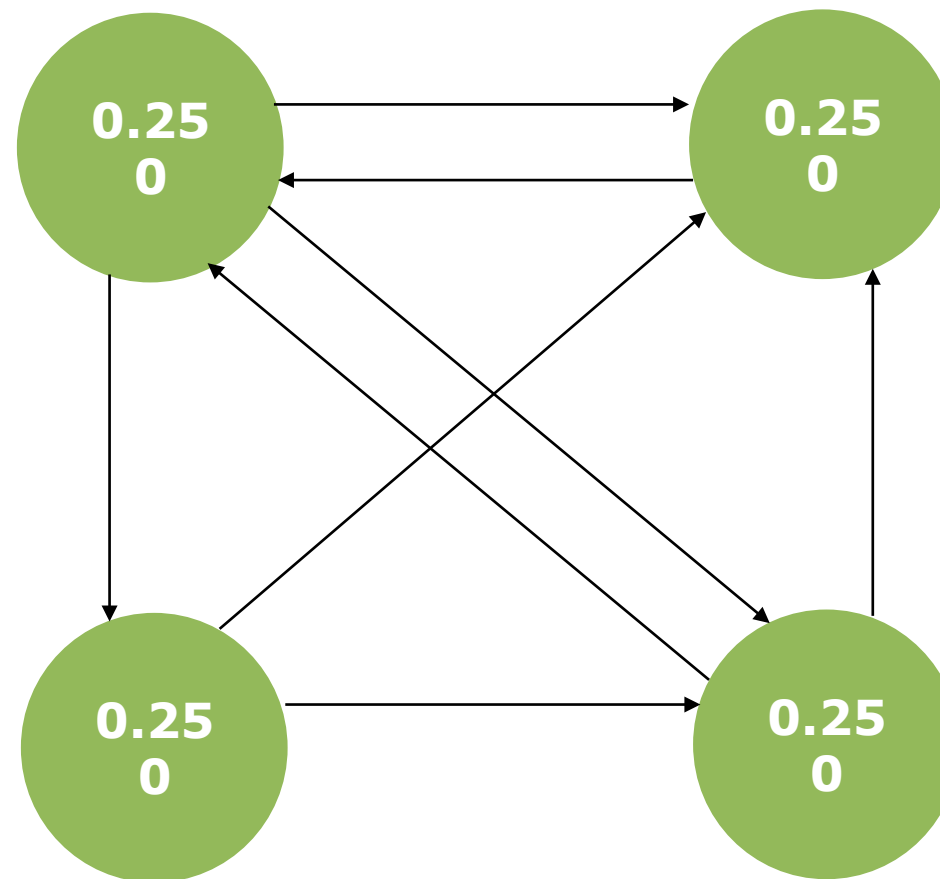
- Convergence is guaranteed for **strongly connected** graphs (i.e. if you can get from any page to any other page in a finite number of steps)

Random surfer interpretation

- The surfer **randomly moves** from one page to next
 - If the surfer is currently at a page with r outgoing links, he randomly chooses one of these links
- The component x_j of the normalized score vector \mathbf{x} is
 - the **fraction of time** that the surfer spends, in the long run, on page j of the web or, equivalently
 - the **probability** that a surfer following a **random walk** has arrived on page i at some distant point in the future
- More important pages tend to be linked to by many other pages and so the surfer hits those most often.

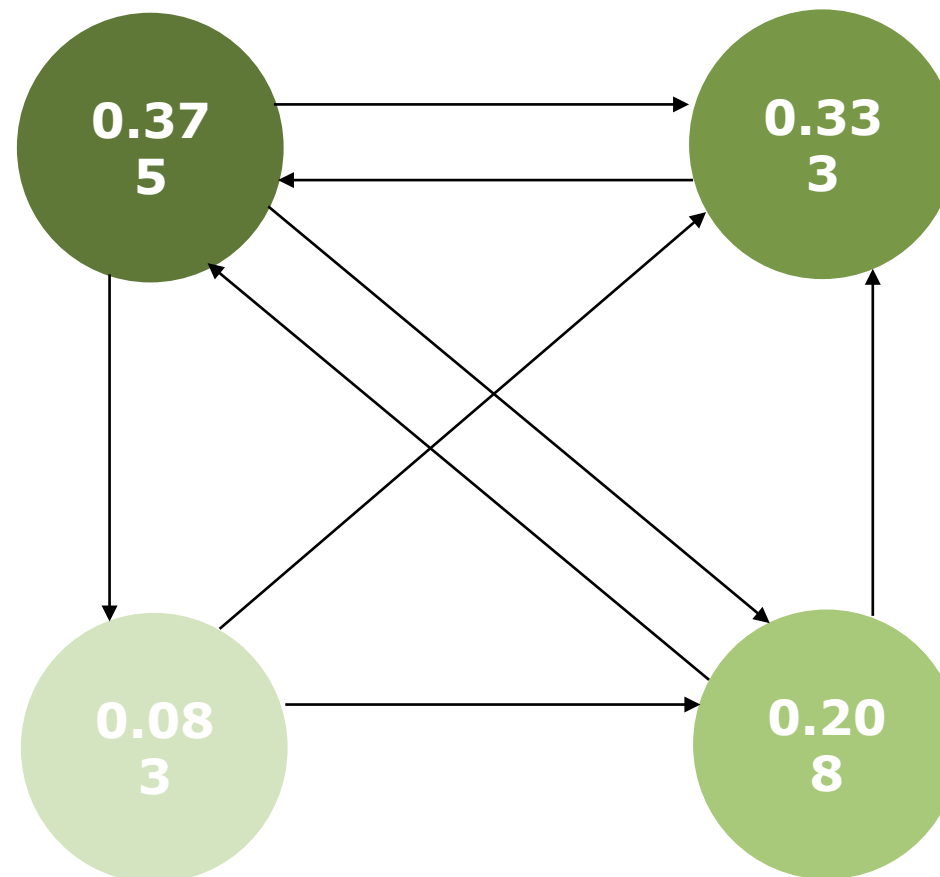
- Power iterations

$k = 0$



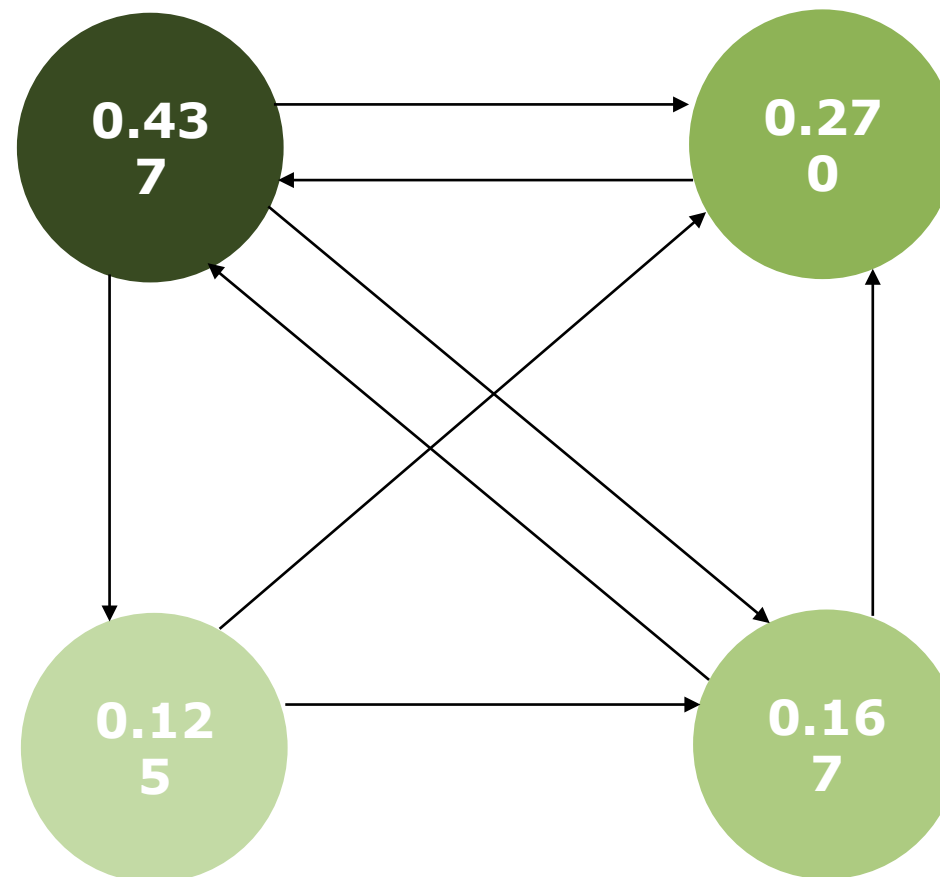
- Power iterations

$k = 1$



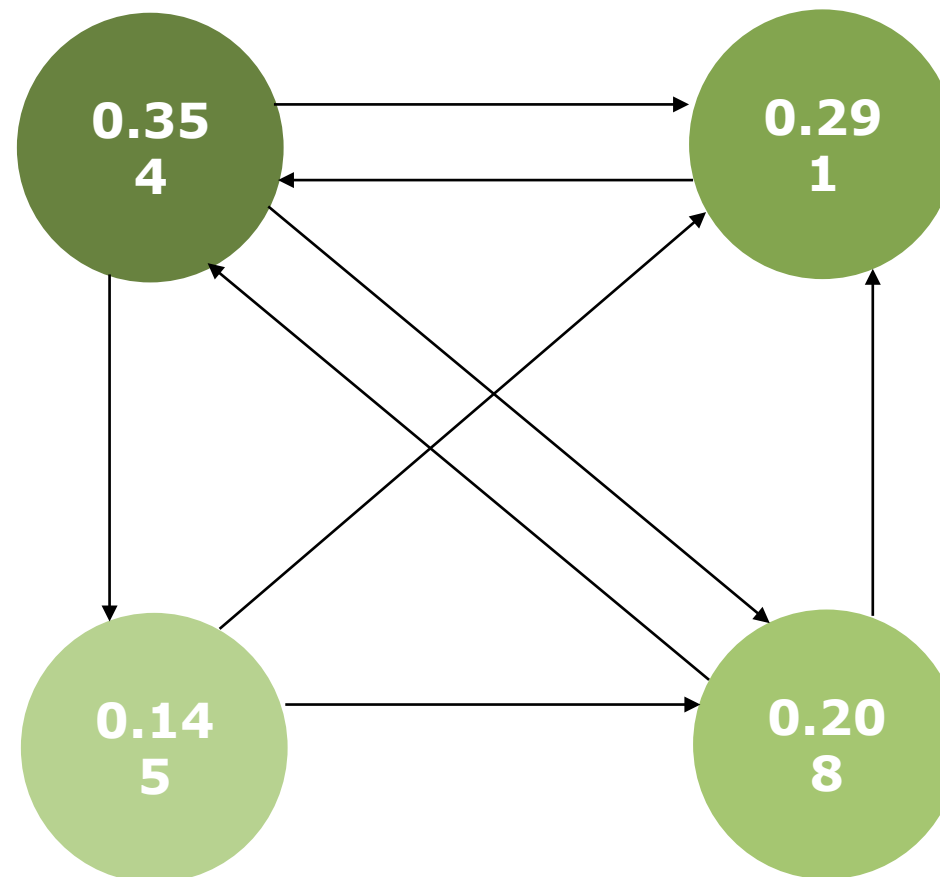
- Power iterations

$k = 2$



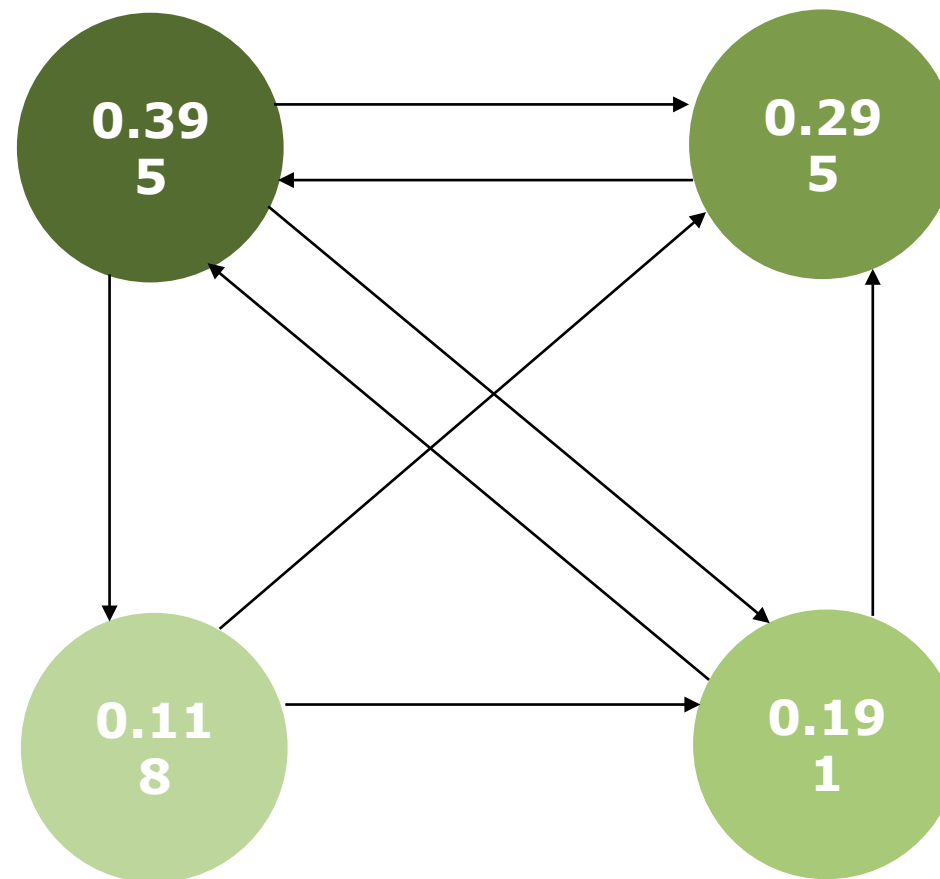
- Power iterations

$k = 3$



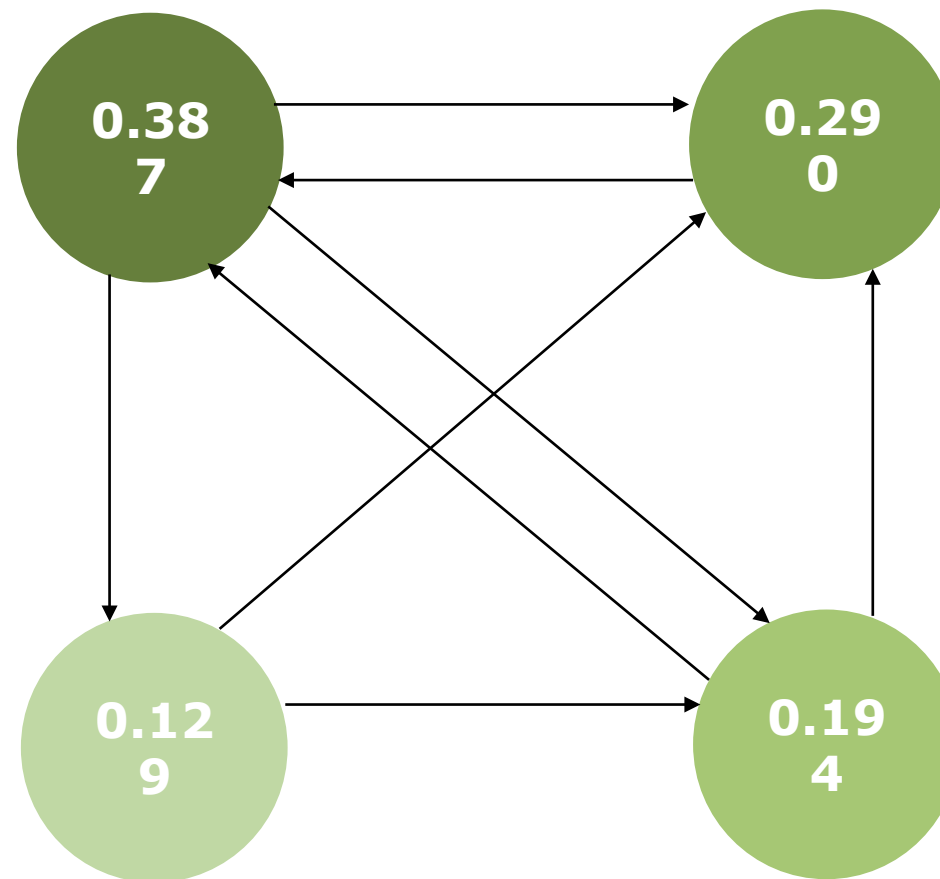
- Power iterations

$k = 4$

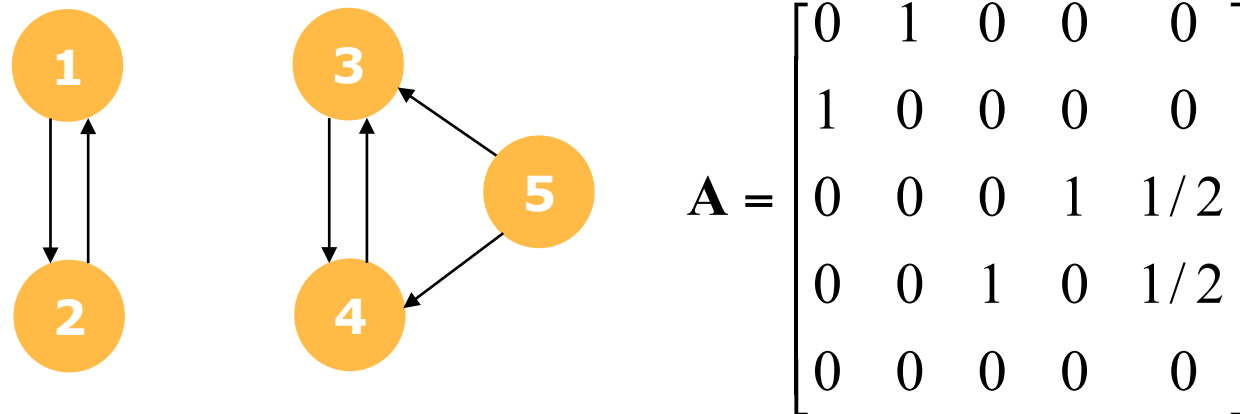


- Upon convergence

$k \rightarrow \infty$



- Power iterations may not always converge to a unique ranking when the graph is not strongly connected
- Example:



- Equation $\mathbf{x} = \mathbf{A}\mathbf{x}$ is satisfied by both vectors:

$$\mathbf{x}^{(1)} = [1/2 \quad 1/2 \quad 0 \quad 0 \quad 0]^T$$

$$\mathbf{x}^{(2)} = [0 \quad 0 \quad 1/2 \quad 1/2 \quad 0]^T$$

and by any linear combination of $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$

- To fix this, at each step the random surfer can randomly jump to any page of the Web with some probability m ($1 - m$: **damping factor**)
- We can generate unique importance scores by modifying the matrix \mathbf{A} as follows

$$\mathbf{M} = (1 - m)\mathbf{A} + m\mathbf{S}$$

$$\text{where } \mathbf{S} \in \mathbb{R}^{n \times n}, S_{i,j} = 1/n \\ 0 \leq m < 1$$

- The **PageRank** score is computed as

$$\mathbf{x} = \lim_{k \rightarrow \infty} \mathbf{M}^k \mathbf{x}_0$$

Enhanced Random surfer interpretation

- The surfer **randomly moves** from one page to next
 - If the surfer is currently at a page with r outgoing links,
 - with probability $1-m$ he randomly chooses one of these links
 - with probability m he jumps to any randomly selected page on the web
- Equivalent to adding implicit links from each page to any other page
- The resulting graph is strongly connected

- For example, setting $m = 0.15$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1/2 \\ 0 & 0 & 1 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \mathbf{M} = \begin{bmatrix} 0.03 & 0.88 & 0.03 & 0.03 & 0.03 \\ 0.88 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.88 & 0.455 \\ 0.03 & 0.03 & 0.88 & 0.03 & 0.455 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \end{bmatrix}$$

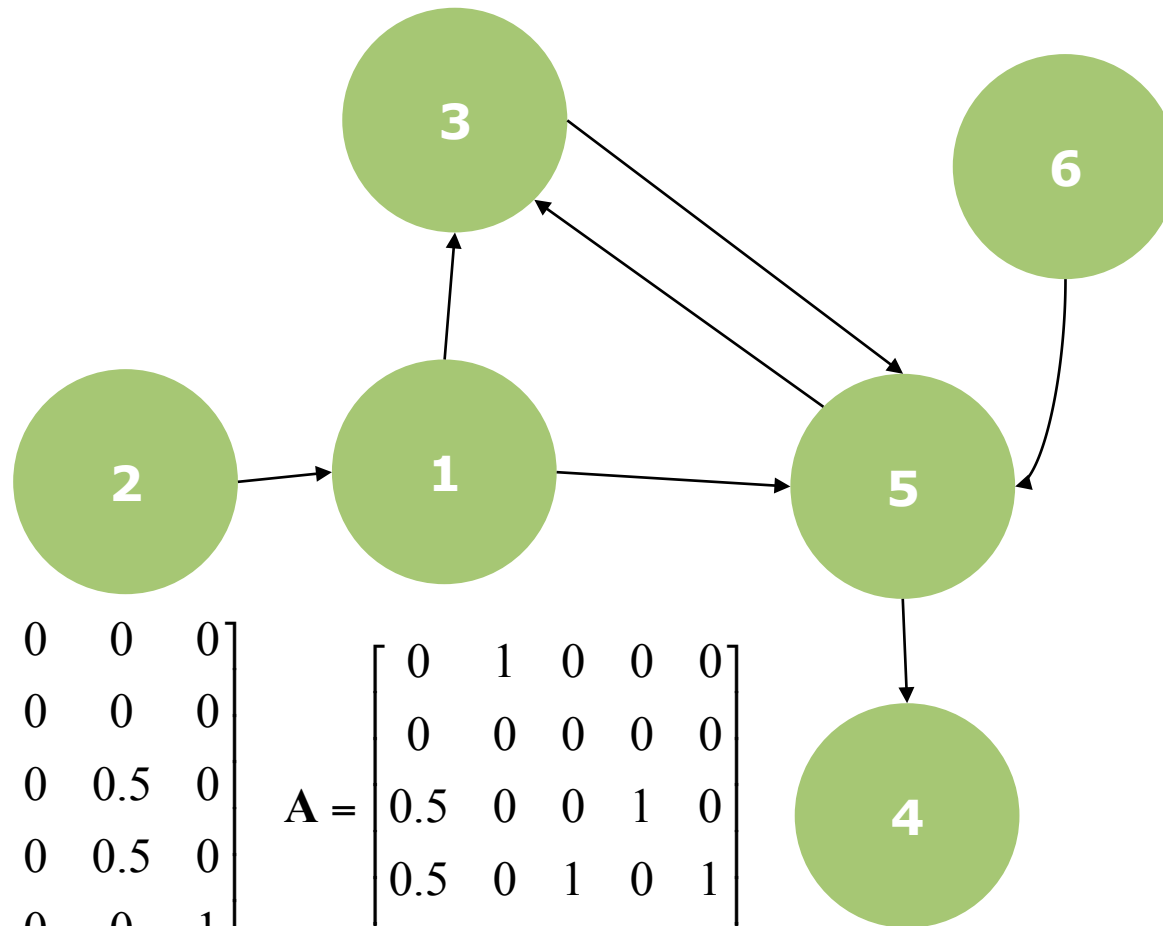
- The unique PageRank score is given by

$$\mathbf{x} = [0.2 \quad 0.2 \quad 0.285 \quad 0.285 \quad 0.03]^T$$

- **dangling node**: a node without outgoing links
- Common examples of webpages with no links to other pages:
 - image & music files
 - pdf files
 - pages whose links haven't been crawled
 - protected pages
- some estimates say more than 50%, others say between 60% and 80%

- A web with **dangling nodes** produces a matrix **A** which contains one or more columns of all zeros. Matrix **A** does not have an eigenvalue equal to 1
- Solution 1:
 - **Remove** dangling nodes
 - build matrix **M**, with $m > 0$
 - Compute the importance scores for the remaining nodes
 - **Reinsert** dangling nodes, computing their importance scores based on incoming links only
- Solution 2
 - **replace the all-zero columns** of matrix **A** corresponding to dangling nodes with $1/n$ term
 - equivalent to **browsing to a new page** at random when the user reaches a page with no outgoing links
 - build matrix **M**, with $m > 0$
 - Compute the importance scores

- Remove dangling nodes

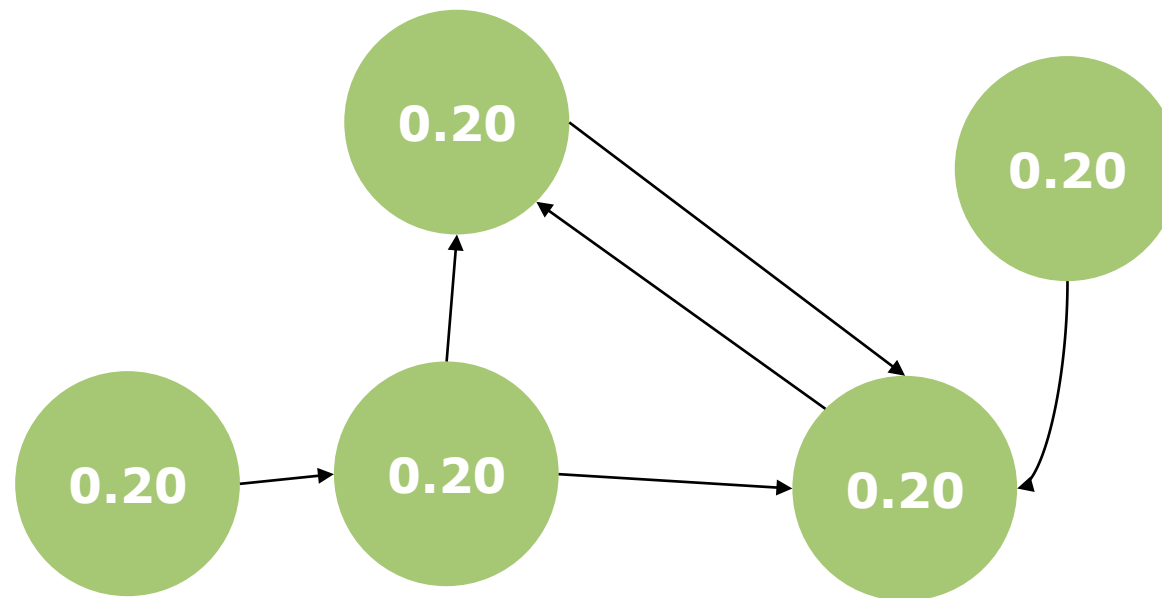


$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0.5 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 1 & 0 & 0 \\ 0.5 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

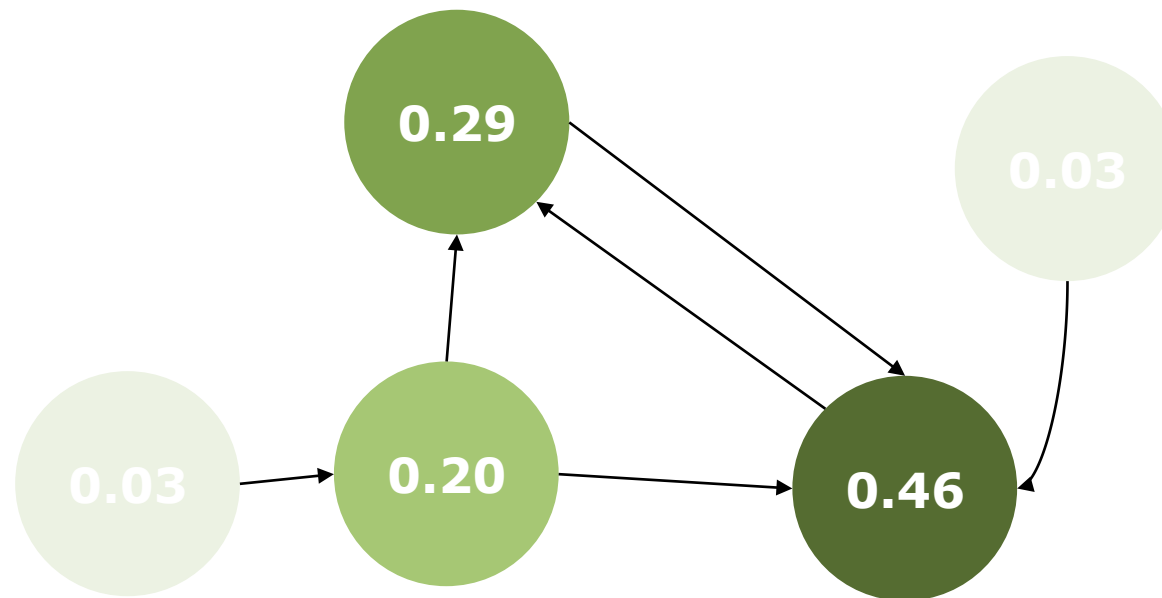
- Power iterations

$k = 0$



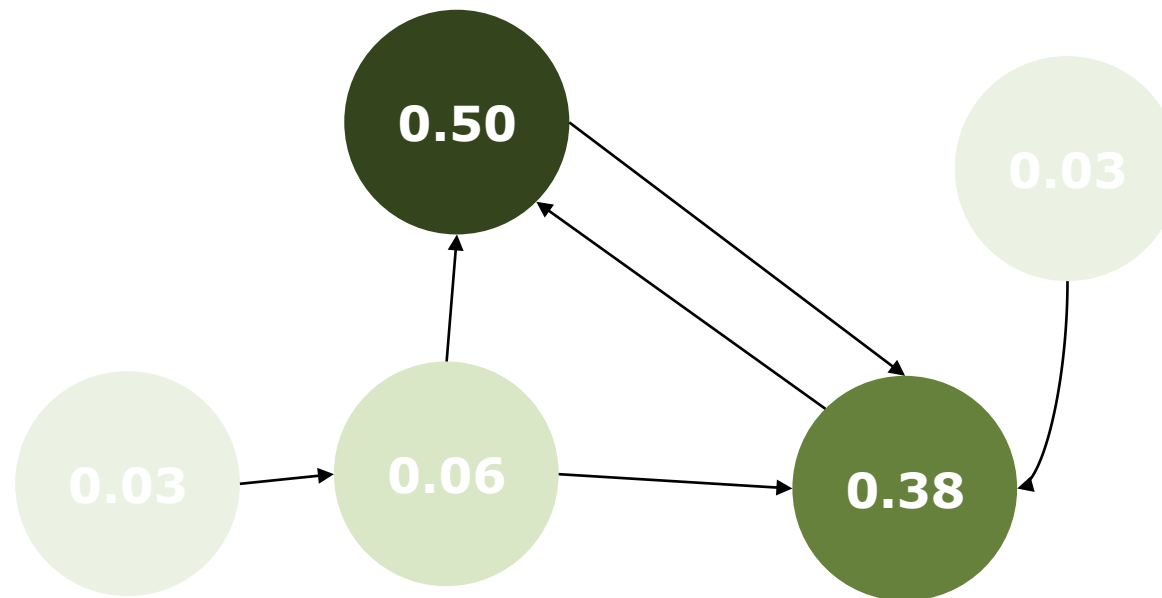
- Power iterations (with damping factor $m = 0.15$)

$k = 1$



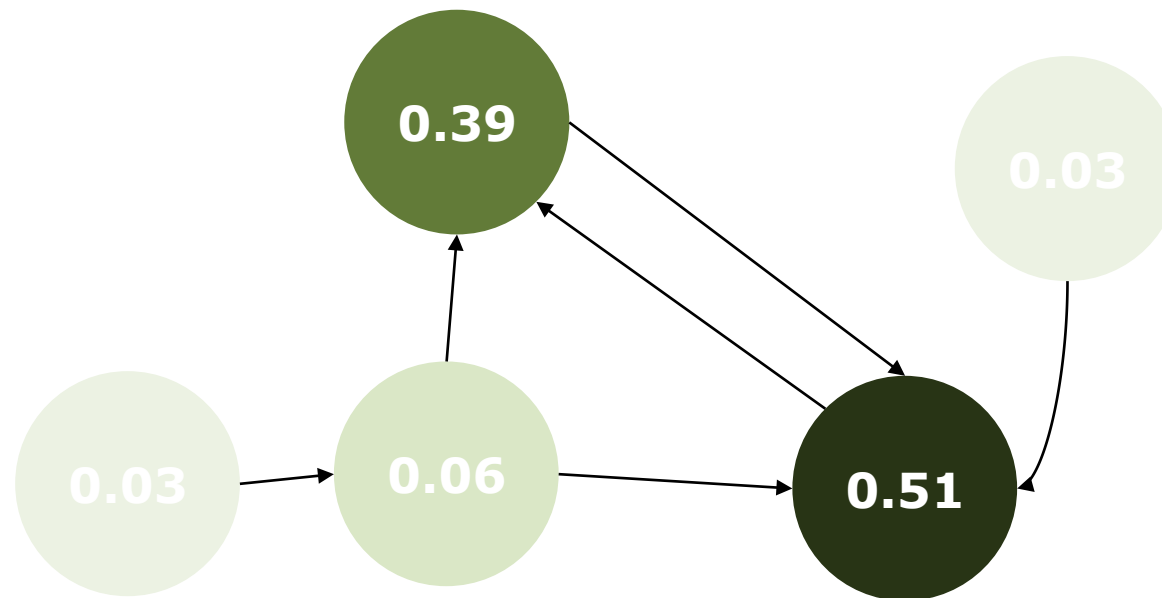
- Power iterations (with damping factor $m = 0.15$)

$k = 2$



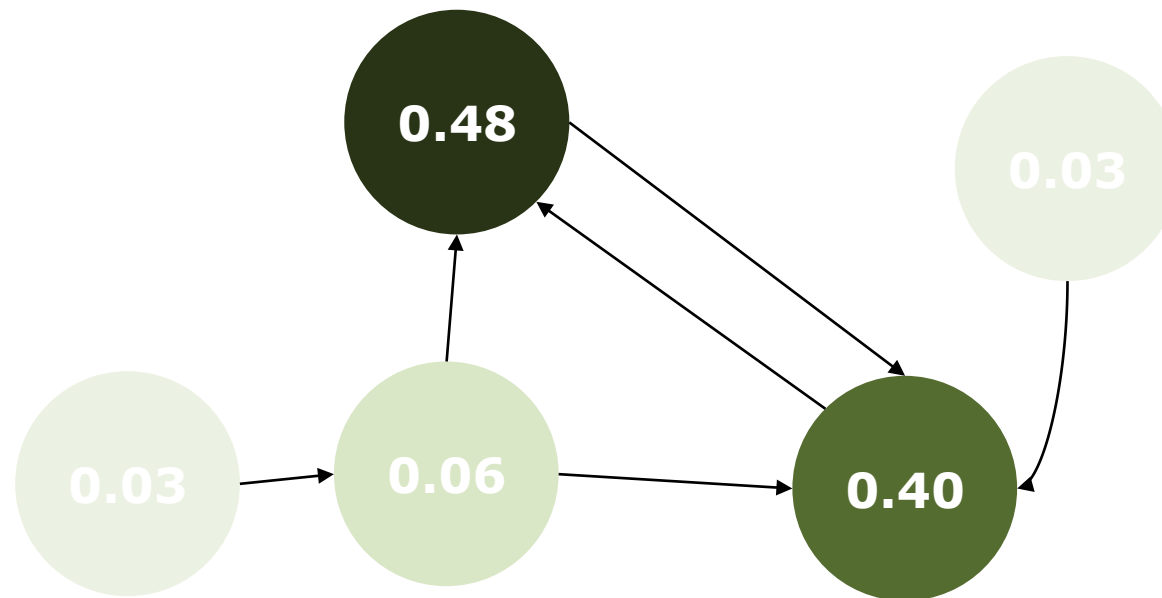
- Power iterations (with damping factor $m = 0.15$)

$k = 3$



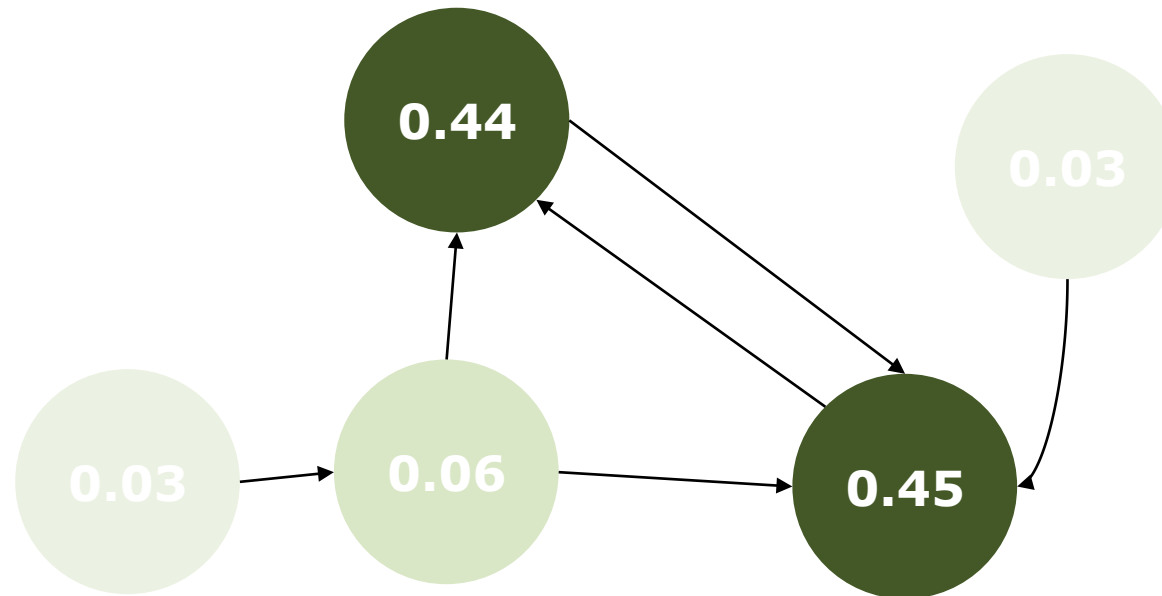
- Power iterations (with damping factor $m = 0.15$)

$k = 4$

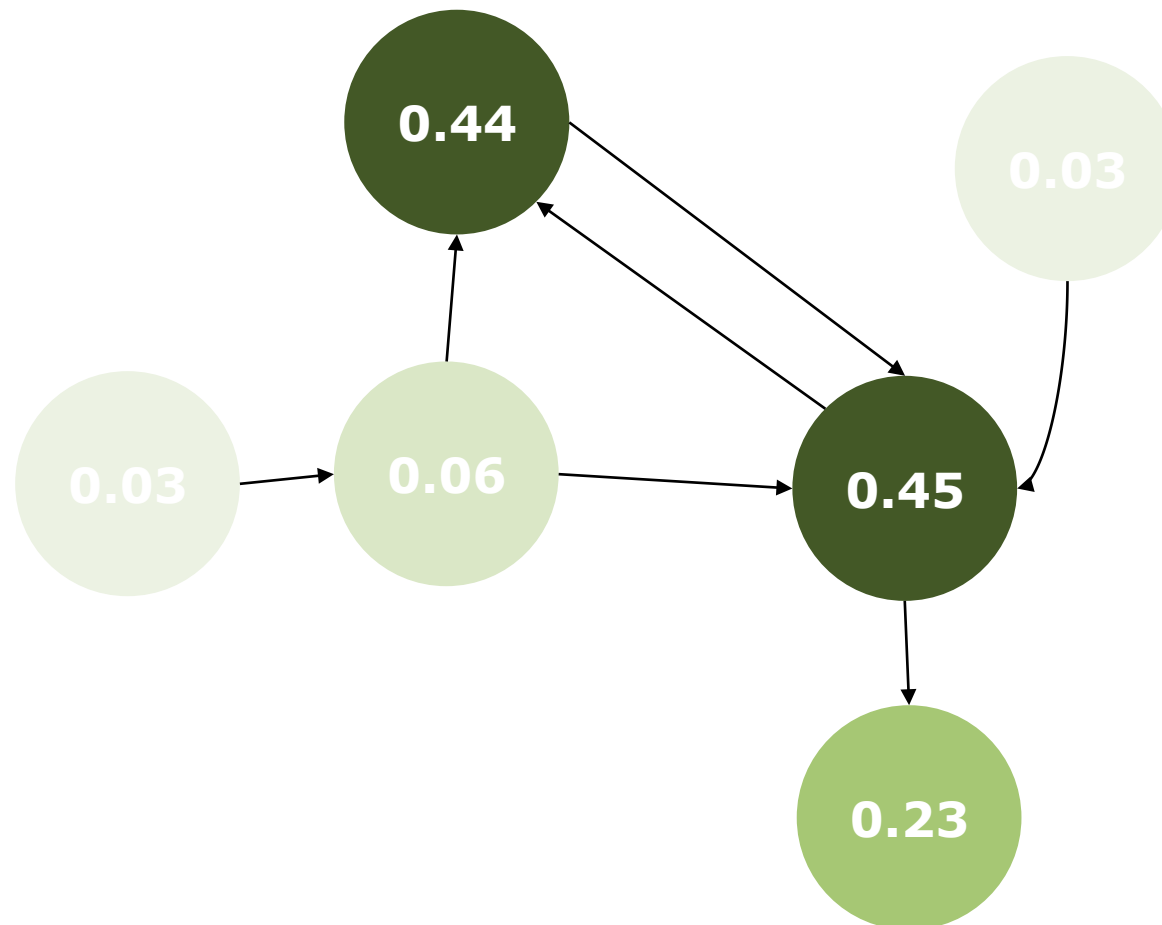


- Power iterations (with damping factor $m = 0.15$)

$k \rightarrow \infty$



- Add dangling node ($x_4 = x_5 / 2$)



- We still need an efficient way of solving $\mathbf{x} = \mathbf{M}\mathbf{x}$

- Power method

- Start with a typical \mathbf{x}_0 $\|\mathbf{x}_0\|_1 = 1 \quad x_{0,i} > 0$

- Generate the sequence $\mathbf{x}_k = \mathbf{M}\mathbf{x}_{k-1} \quad \mathbf{x}_k = \mathbf{M}^k \mathbf{x}_0$
(so)

- It is possible to prove that any positive stochastic matrix \mathbf{M} has a unique vector \mathbf{q} with positive components such that $\mathbf{M}\mathbf{q} = \mathbf{q}$, with $\|\mathbf{q}\|_1 = 1$

- The vector can be computed as $\mathbf{q} = \lim_{k \rightarrow \infty} \mathbf{M}^k \mathbf{x}_0$

- As a practical matter, note that the $n \times n$ positive matrix \mathbf{M} has **no non-zero elements**
- The multiplication $\mathbf{M}\mathbf{v}$, for $\mathbf{v} \in \mathbb{R}^n$ typically takes $O(n^2)$, which is huge if $n = 8.000.000.000$

- The equation

$$\mathbf{x} = \mathbf{M}\mathbf{x} = (1 - m)\mathbf{A}\mathbf{x} + m\mathbf{S}\mathbf{x}$$

can be cast as

$$\mathbf{x} = (1 - m)\mathbf{A}\mathbf{x} + m\mathbf{s}$$

where $\mathbf{s} \in \mathbb{R}^n, s_i = 1/n$

Since $\|\mathbf{x}\|_1 = 1$ if $\mathbf{S}\mathbf{x} = \mathbf{s}$

- In practice we will use this iteration (valid if $\|\mathbf{x}_{k-1}\|_1 = 1$)

$$\mathbf{x}_k = (1 - m)\mathbf{A}\mathbf{x}_{k-1} + m\mathbf{s}$$

- This is far more efficient, since the matrix \mathbf{A} is **sparse**

- Google displays the PageRank of each page in the Google toolbar



- PageRank scores are converted into a logarithmic-like scale and assume values between 0 and 10

- PageRank scores are **continuously** updated by Google, but they are **exported** and made available to the Google toolbar periodically, typically every few months.
- PageRank can be used to raise the **weight** of important pages:

$$weight(t, d) = TFIDF(t, d)PR(d)$$

where t is an indexed term and d is an indexed page

$TFIDF$ = term-frequency*inverse document frequency

$PR(d)$ = PageRank score of document d

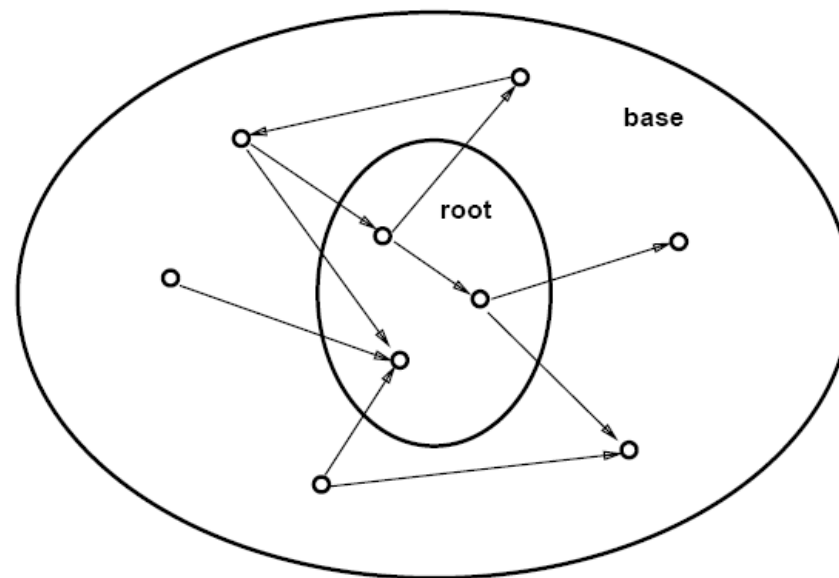
- Obviously, the actual use of PageRank by Google is confidential

- Search engine evolution
- Modeling the Web
- Web Crawling
- Link Analysis
- Page Rank
- HITS

HITS: Hypertext Induced Topic Search

- **Query dependent.** Link analysis carried out over query induced graph
- Two kind of important pages:
 - **hubs** are pages that point to good authorities
 - **authorities** are pages that point to good hubs
- Two scores (authority and hub score) for each page
 - Hub's value is in the **links which exit the node**, as it is used in order to select the pages containing relevant information
 - Authority's value is in the **links which enter the node**, as it is used to describe contents
 - Good hubs point to good authorities. Good authorities are pointed by good hubs
- Iterative score computation

1. Send query q to text-based IR system to obtain the **root set** of pages, i.e. using a boolean or a **vector space model**
2. Expand the base set by radius one to obtain an expanded graph (**base set**)
 - add pages with **outgoing** links to pages in the root set
 - add pages with **incoming** links from pages in the root set



3. Run power iteration

- Let \mathbf{E} denote the base set graph incidence matrix
- $\mathbf{E}(u, v) = 1$ if there is a link from u to v
- while \mathbf{a} and \mathbf{h} change “significantly”, do

$$\begin{aligned} \mathbf{a}_k &= \mathbf{E}^T \mathbf{h}_{k-1} & \mathbf{a}_k &= \mathbf{a}_k / \|\mathbf{a}_k\|_1 \\ \mathbf{h}_k &= \mathbf{E} \mathbf{a}_k & \mathbf{h}_k &= \mathbf{h}_k / \|\mathbf{h}_k\|_1 \end{aligned}$$

where

- \mathbf{h} is the hub score
- \mathbf{a} is the authority score

4. Report top-ranking authorities and hubs

- Note: the entire process called **topic distillation**

- Intuition behind power iteration equations:

$$\mathbf{a}_k^0 = \mathbf{E}^T \mathbf{h}_{k-1} \quad \mathbf{a}_k = \mathbf{a}_k^0 / \|\mathbf{a}_k^0\|_1 \quad (1)$$

$$\mathbf{h}_k^0 = \mathbf{E} \mathbf{a}_k \quad \mathbf{h}_k = \mathbf{h}_k^0 / \|\mathbf{h}_k^0\|_1 \quad (2)$$

- Top hub** pages increase the **authority score** of pages they point to – equation (1).
 - top hubs “vote for” top authorities
- Top authority** pages increase the **hub score** of pages they point to – equation (2)
 - top authorities “vote for” top hubs
- Upon convergence, vectors **a** and **h** can be sorted to report top-rank authority and top-rank hub pages

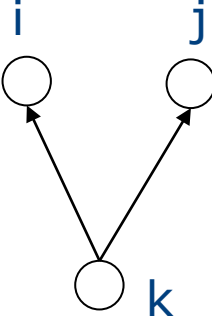
- The computation of hub and authority scores can be cast as the problem of **seeking an eigenvector** associated to the largest eigenvalue of a square matrix
- In fact, power iteration equations can be rewritten obtaining separate recursive equations for each score

$$\mathbf{h}_k^0 = \mathbf{E}\mathbf{E}^T \mathbf{h}_{k-1} \quad \mathbf{h}_k = \mathbf{h}_k^0 / \|\mathbf{h}_k^0\|_1$$

$$\mathbf{a}_k^0 = \mathbf{E}^T \mathbf{E} \mathbf{a}_{k-1} \quad \mathbf{a}_k = \mathbf{a}_k^0 / \|\mathbf{a}_k^0\|_1$$

- Hub score converges to an **eigenvector** corresponding to largest eigenvalue of $\mathbf{E}\mathbf{E}^T$
- Authority score converges to an **eigenvector** corresponding to largest eigenvalue of $\mathbf{E}^T \mathbf{E}$

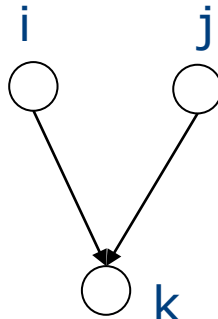
- $\mathbf{E}^T \mathbf{E}$ is the **authority matrix**:

$$\mathbf{A} = \mathbf{E}^T \mathbf{E} = \left(\begin{array}{c} \vdots \\ \mathbf{A}_{ij} \\ \vdots \end{array} \right) = \underbrace{\left(\begin{array}{c} \vdots \\ \mathbf{E}_{ik}^T \\ \vdots \end{array} \right)}_{\mathbf{E}^T} \underbrace{\left(\begin{array}{c} \vdots \\ \mathbf{E}_{kj} \\ \vdots \end{array} \right)}_{\mathbf{E}}$$


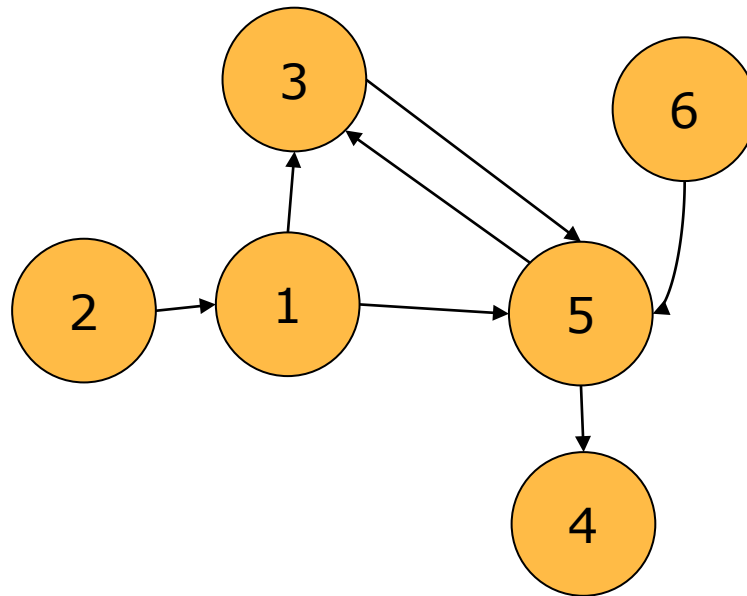
The diagram shows a node labeled k at the bottom with two arrows pointing upwards to nodes labeled i and j . This illustrates that node k is a co-citation for both nodes i and j .

- $\mathbf{A}_{ij} = \sum_{k=1}^n \mathbf{E}_{ik}^T \mathbf{E}_{kj} = \sum_{k=1}^n \mathbf{E}_{ki} \mathbf{E}_{kj}$ is the number of co-citations, i.e. the number of nodes that point to both i and j

- $\mathbf{E}\mathbf{E}^T$ is the **hub matrix**:

$$\mathbf{H} = \mathbf{E}\mathbf{E}^T = \begin{matrix} & \text{j} \\ \text{i} & \left[\begin{array}{c} \text{---} \mathbf{H}_{ij} \text{---} \\ \text{---} \end{array} \right] \end{matrix} = \underbrace{\left[\begin{array}{c} \text{---} \mathbf{E}_{ik} \text{---} \\ \text{---} \end{array} \right]}_{\mathbf{E}} \underbrace{\left[\begin{array}{c} \text{---} \mathbf{E}_{kj}^T \text{---} \\ \text{---} \end{array} \right]}_{\mathbf{E}^T}$$


- $\mathbf{H}_{ij} = \sum_{k=1}^n \mathbf{E}_{ik} \mathbf{E}_{kj}^T = \sum_{k=1}^n \mathbf{E}_{ik} \mathbf{E}_{jk}$ is the number of co-reference, i.e. the number of nodes that pointed by both i and j

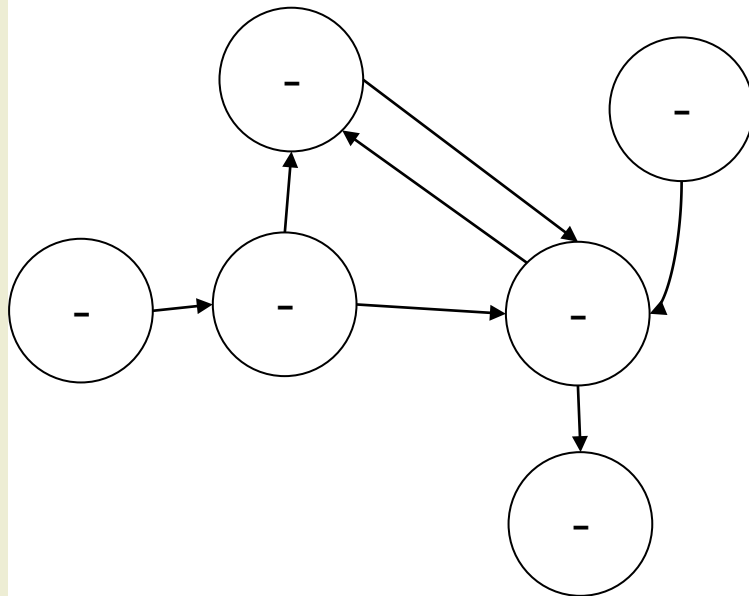


$$\mathbf{E}\mathbf{E}^T = \begin{bmatrix} 2 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

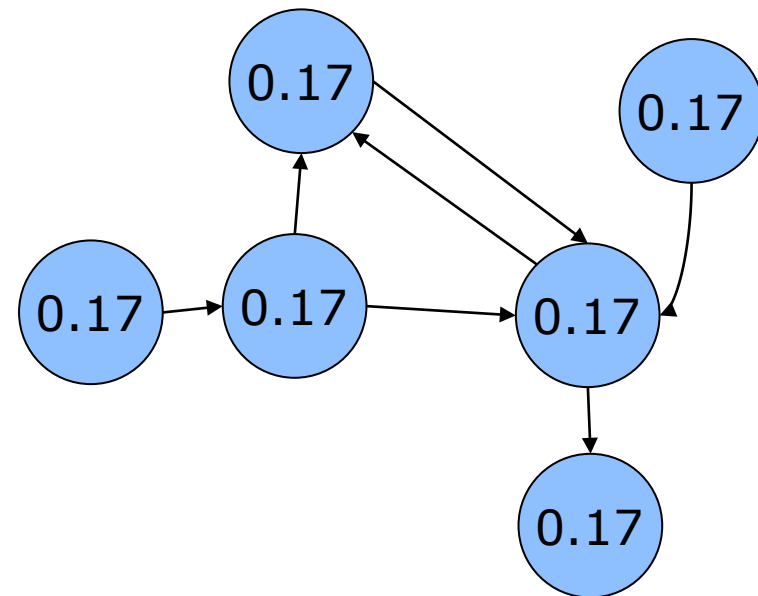
$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{E}^T\mathbf{E} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Power iterations



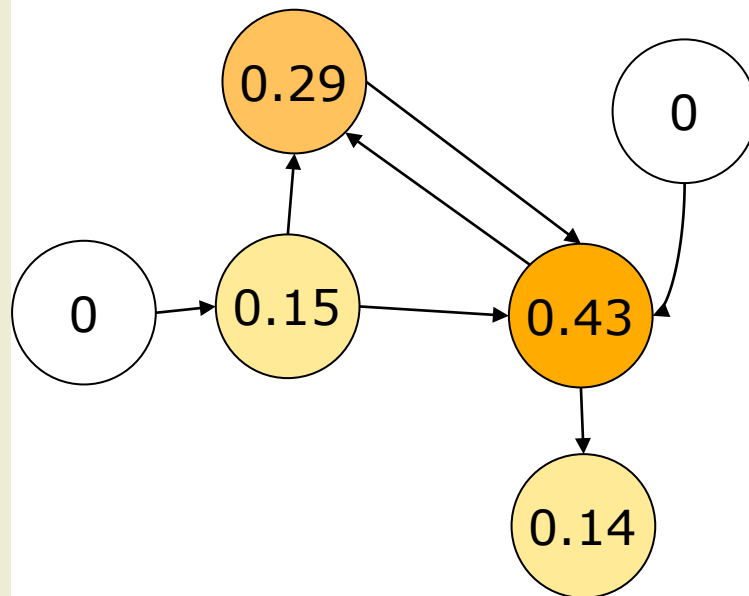
authority scores



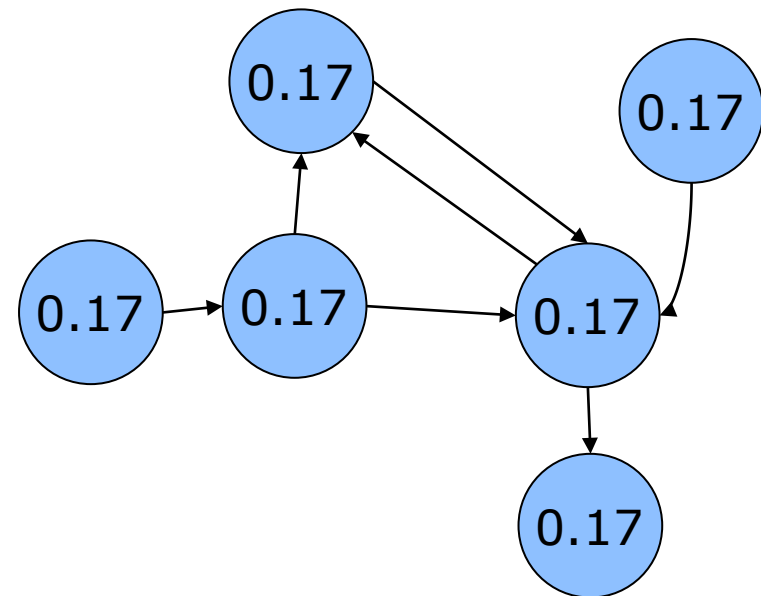
hub scores

 \mathbf{h}_0

- Power iterations



authority scores

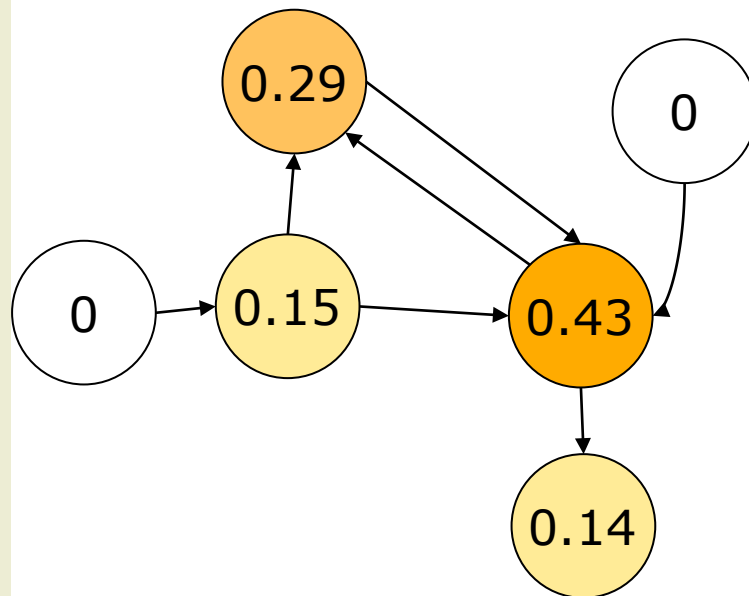


hub scores

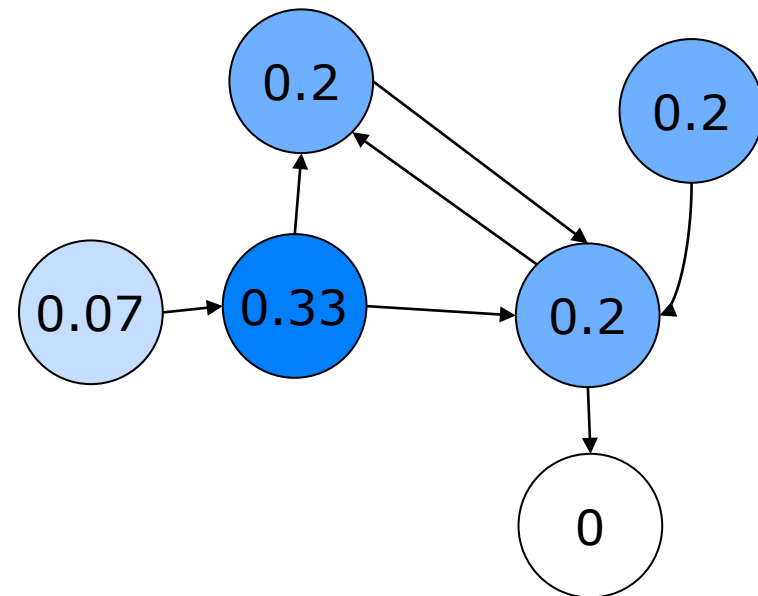
$$\mathbf{a}_1 = \frac{\mathbf{E}^T \mathbf{h}_0}{\|\mathbf{E}^T \mathbf{h}_0\|_1}$$

$$\mathbf{h}_0$$

- Power iterations



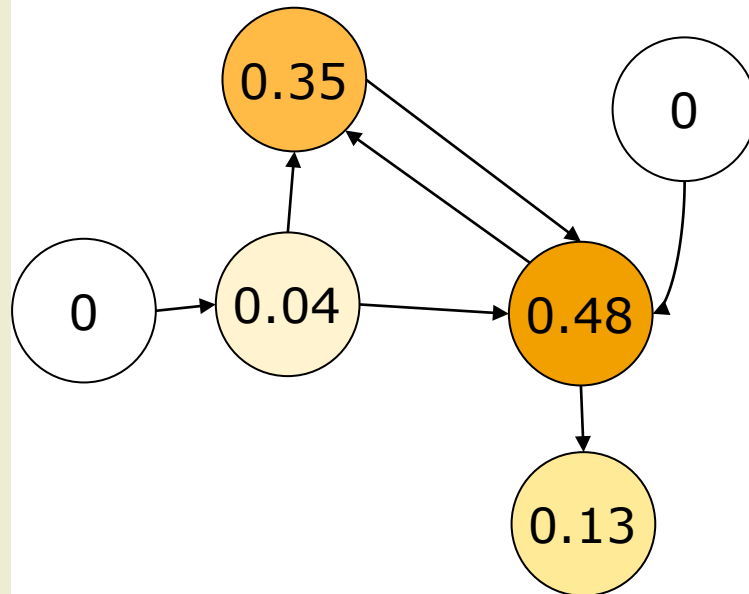
authority scores

 \mathbf{a}_1 

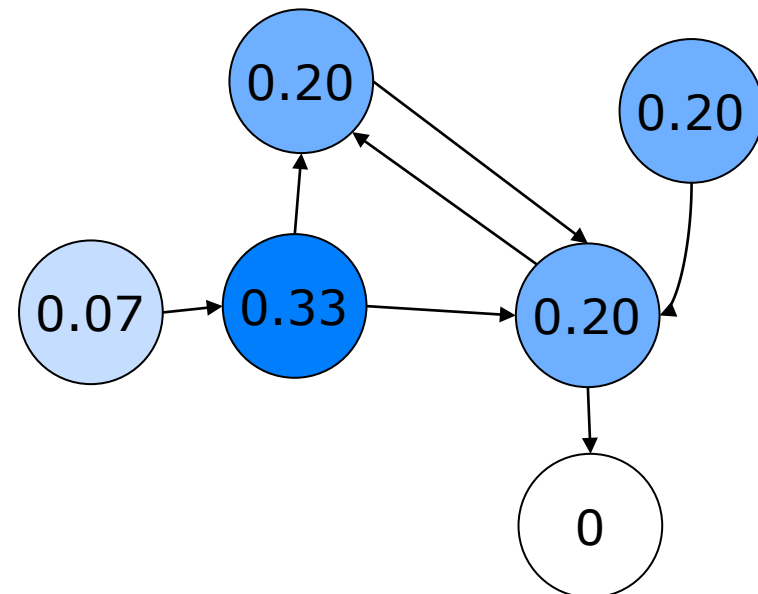
hub scores

$$\mathbf{h}_1 = \frac{\mathbf{E}\mathbf{a}_1}{\|\mathbf{E}\mathbf{a}_1\|_1}$$

- Power iterations



authority scores

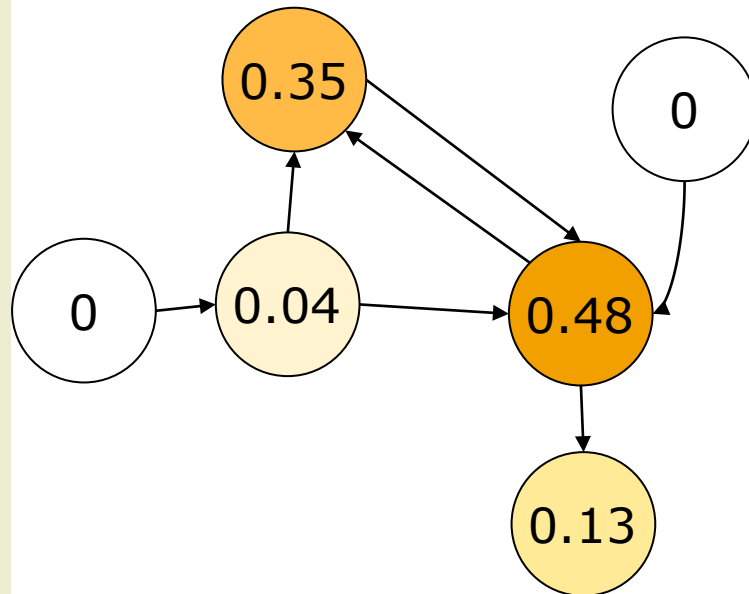


hub scores

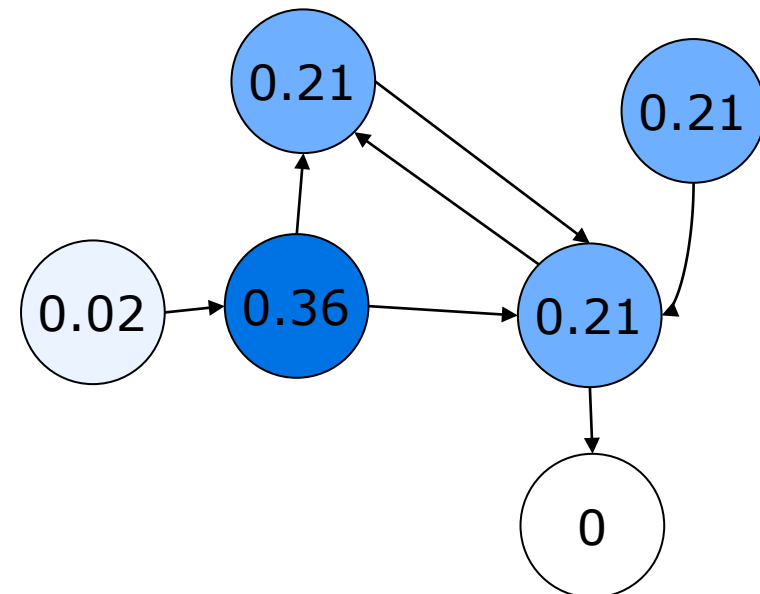
$$\mathbf{a}_2 = \frac{\mathbf{E}^T \mathbf{h}_1}{\|\mathbf{E}^T \mathbf{h}_1\|_1}$$

 \mathbf{h}_1

- Power iterations



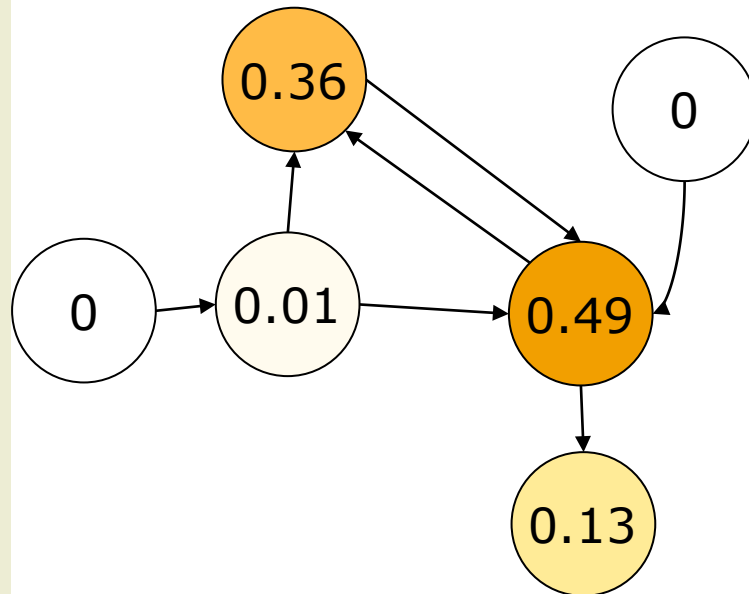
authority scores

 \mathbf{a}_2 

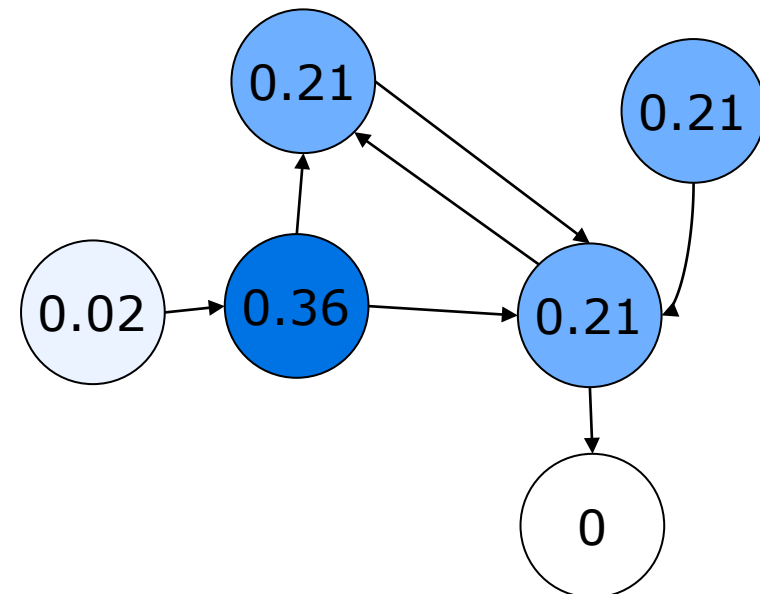
hub scores

$$\mathbf{h}_2 = \frac{\mathbf{E}\mathbf{a}_2}{\|\mathbf{E}\mathbf{a}_2\|_1}$$

- Power iterations



authority scores

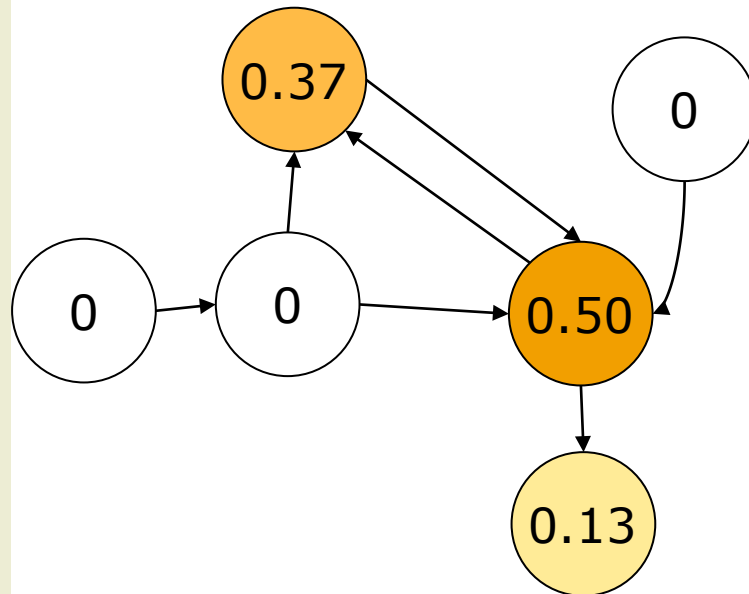


hub scores

$$\mathbf{a}_3 = \frac{\mathbf{E}^T \mathbf{h}_2}{\|\mathbf{E}^T \mathbf{h}_2\|_1}$$

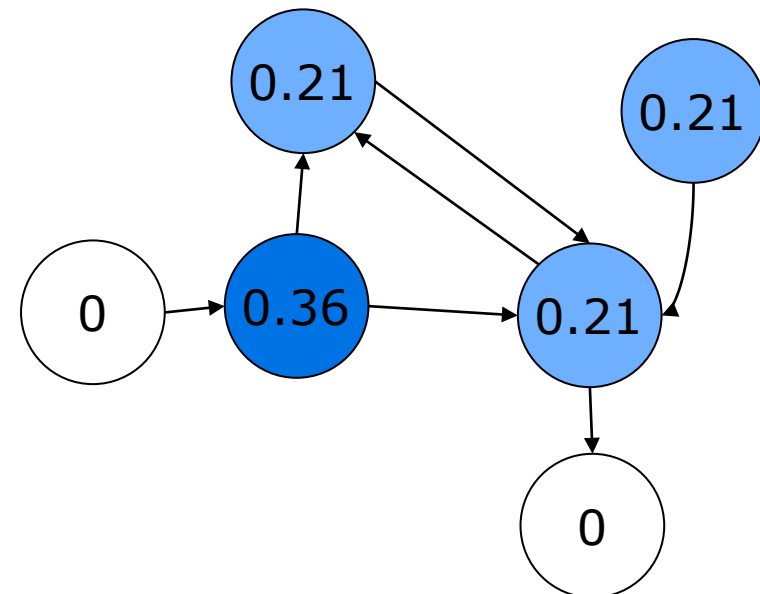
 \mathbf{h}_2

- Upon convergence



authority scores

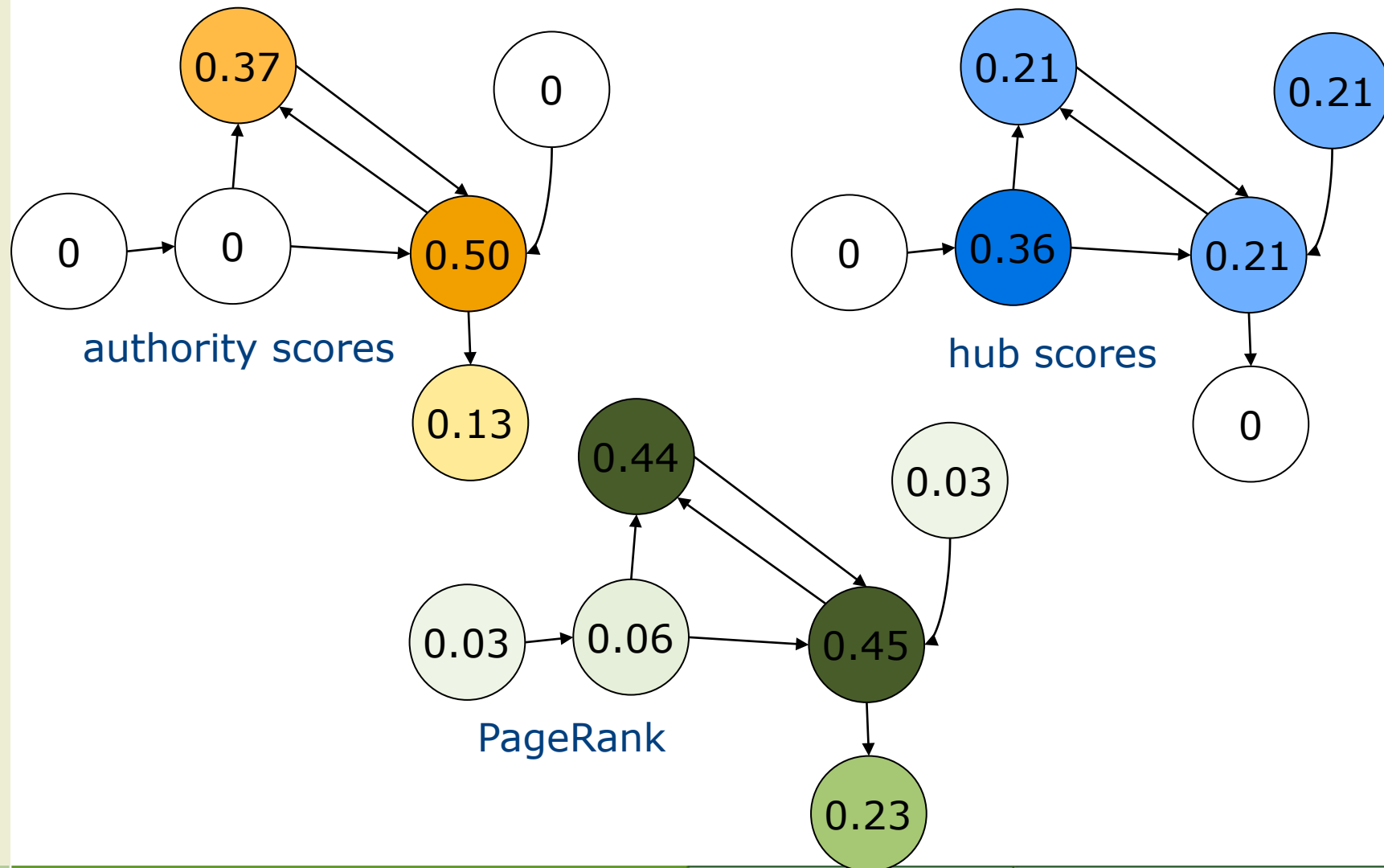
\mathbf{a}_{∞}



hub scores

\mathbf{h}_{∞}

- Compare with PageRank



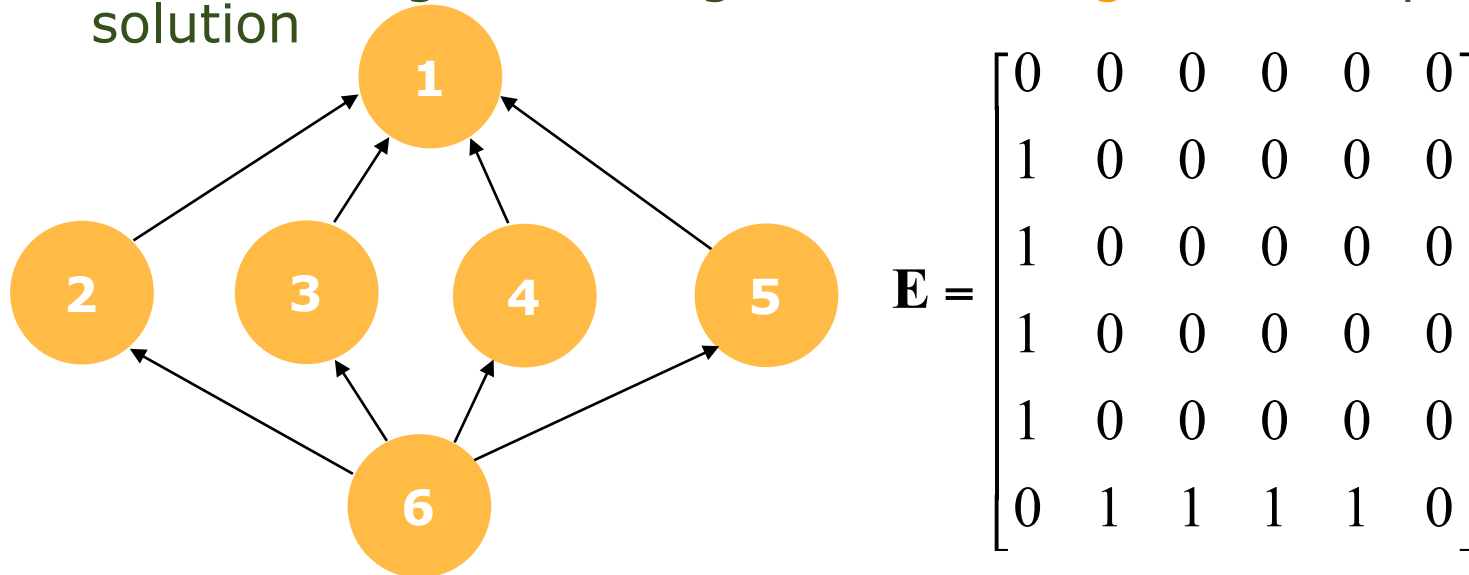
Comparison of HITS with PageRank

- HITS computes **two scores** (hub and authority) for each page
- PageRank computes **one score** for each page
- PageRank scores are **similar** (but not identical) to authority scores computed by HITS
 - In PageRank, page i confers authority to page j if there is a link between i and j (cfr. matrix \mathbf{A})
 - In HITS, page i confers authority to page j if there is a page that links to both of them (cfr. matrix $\mathbf{E}^T\mathbf{E}$)
- PageRank does not identify top hub pages

- Issues:
 - non-unique ranking
 - if the graph is not strongly connected, as it is the case for the Web graph, HITS might not converge to a unique solution
 - disconnected webs
 - upon convergence, HITS assigns non-negative scores only to the largest connected component
 - sensitivity to local topology
 - subtle changes in the local graph topology might significantly alter HITS scores
 - nepotistic links
 - a group of sites can endorse each other by adding cross links in order to artificially increase the HITS scores
 - topic drift/contamination
 - the expansion step might add pages to the base set whose topic might differ from the original root set

- Issue:

- The HITS algorithm might **not converge** to a unique solution



- Equation $\mathbf{a} = \mathbf{A}\mathbf{a}$ is satisfied by both vectors:

$$\mathbf{a}^{(1)} = [1 \ 0 \ 0 \ 0 \ 0 \ 0]^T$$

$$\mathbf{a}^{(2)} = [0 \ 0.25 \ 0.25 \ 0.25 \ 0.25 \ 0]^T$$

and by any linear combination of them

- Same argument holds for hub scores

- Solution:
 - similarly to PageRank, we might add a constant to the entries of \mathbf{E} such that \mathbf{A} and \mathbf{H} are **strictly positive** matrices, thus the largest eigenvalue is a simple root
- Example

$$\mathbf{E} = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 1.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 1.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 1.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 1.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 1.1 & 1.1 & 1.1 & 1.1 & 0.1 \end{bmatrix}$$

produces **unique** hub and authority scores

$$\mathbf{a} = [0.30 \quad 0.16 \quad 0.16 \quad 0.16 \quad 0.16 \quad 0.04]^T$$

$$\mathbf{h} = [0.04 \quad 0.16 \quad 0.16 \quad 0.16 \quad 0.16 \quad 0.30]^T$$

- Issue:
 - If the query is ambiguous (e.g. “java” or “jaguar”), the expanded set contains a few, almost **disconnected communities**
 - Each community have a dense subgraph
 - Highest order eigenvectors found by HITS reveal hubs and authorities in **largest community**
- Solution:
 - Find **higher order eigenvectors**

- Find the k principal eigenvectors of $\mathbf{M} = \mathbf{E}\mathbf{E}^T$ or $\mathbf{M} = \mathbf{E}^T\mathbf{E}$ while \mathbf{X} does not converge

$$\mathbf{X} \leftarrow \mathbf{M}\mathbf{X} \quad \mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \dots]^T \in \mathbb{R}^{n \times k}$$

for $i = 1, 2, \dots, k$

for $j = 1, 2, \dots, i-1$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - (\mathbf{x}_i \cdot \mathbf{x}_j) \mathbf{x}_j$$

end

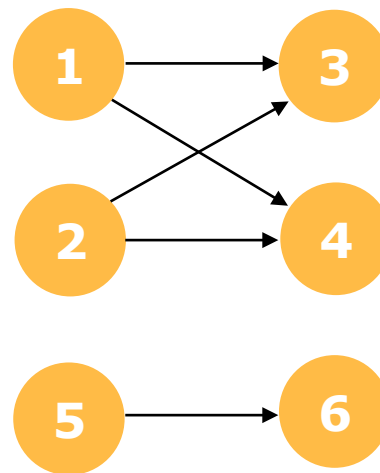
normalize \mathbf{x}_i to unit norm

end

end

Each node has k hub and authority scores (can be used for graph clustering)

- Example with two disconnected webs



$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- The hub and authority scores are **unique**, (since the largest eigenvalue is a single root)
- But they favour **largest connected component**

$$\mathbf{a} = [0 \quad 0 \quad 0.5 \quad 0.5 \quad 0 \quad 0]^T$$

$$\mathbf{h} = [0.5 \quad 0.5 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

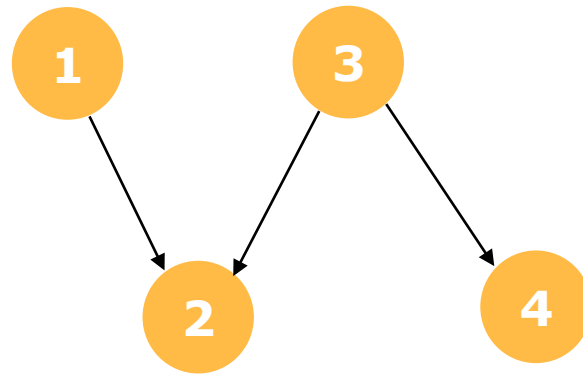
- Finding the eigenvectors associated with the **two largest** eigenvalues

$$\lambda_1 \quad \mathbf{a}^1 = [0 \quad 0 \quad 0.5 \quad 0.5 \quad 0 \quad 0]^T$$
$$\mathbf{h}^1 = [0.5 \quad 0.5 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

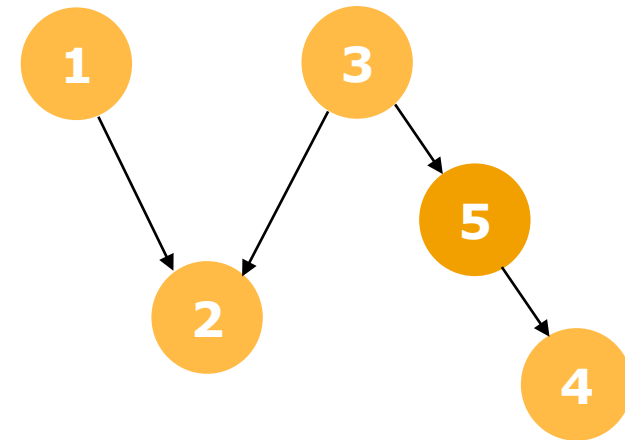
$$\lambda_2 \quad \mathbf{a}^2 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]^T$$
$$\mathbf{h}^2 = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]^T$$

- We get the correct authority and hub scores for the two disconnected components

- Issue:
 - HITS is **sensitive** to local topology



$$\mathbf{a} = [0 \quad 0.61 \quad 0 \quad 0.39]^T$$



$$\mathbf{a} = [0 \quad 0.5 \quad 0 \quad 0 \quad 0.5]^T$$

Node's 4 authority score vanishes to zero compared to those of nodes 2 and 5

- Solutions:
 - **SALSA** -Stochastic Algorithm for Link Structure Analysis
 - Stochastic HITS (**random surfer** behavior)

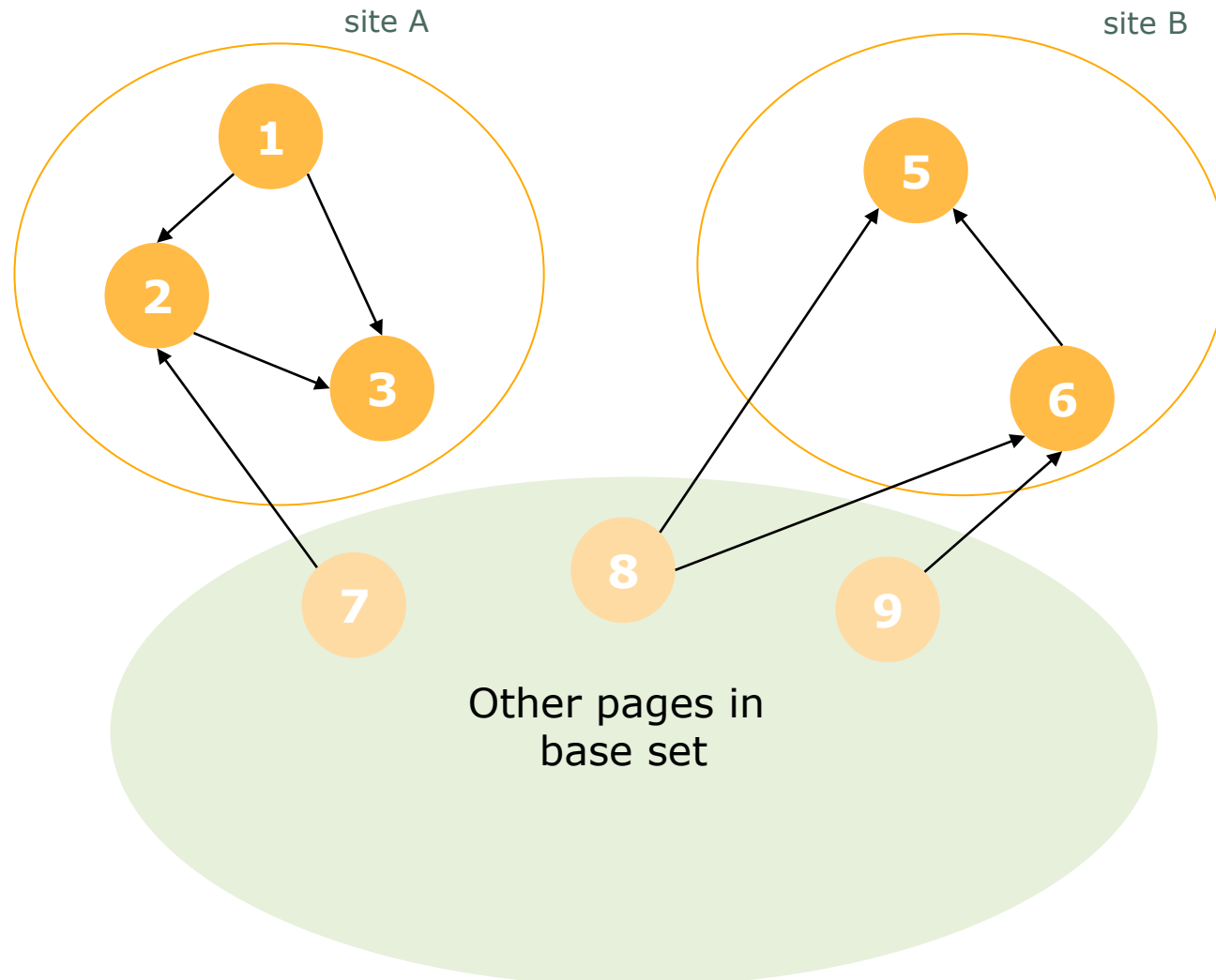
Stochastic HITS:

- At every timestep,
 - with probability d , the surfer jumps to a node in the base set uniformly **at random**
 - with probability $1-d$
 - If it is an odd timestep, the surfer takes a **random outlink** from the current node
 - If it is an even timestep, the surfer goes backward on a **random inlink** leading to the current node
- Stochastic HITS is more **stable** (as d increases) in face of small changes in hyperlink graph

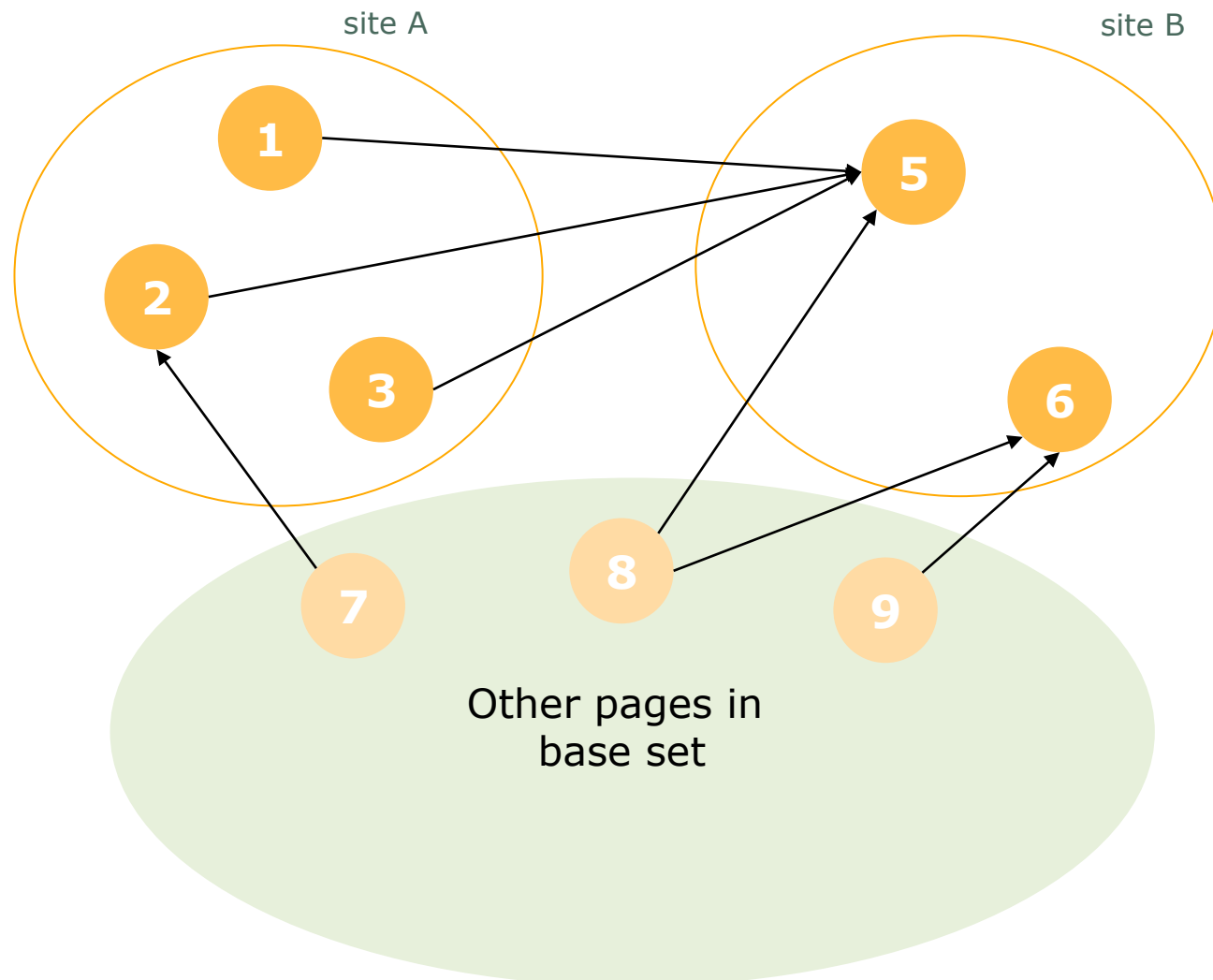
- HITS discards links connecting pages on the same host
 - These links, authored by the same people, do not confer authority and could be regarded as nepotistic
- To artificially boost authority and hub scores, webmasters could add several **nepotistic** links connecting pages of two sites (or multiple sites, when they are managed by the same organization)
- Nepotistic links should not be taken into account (or, at least, deemphasized) when computing hub/authority scores, since they are added with the purpose of **maliciously manipulating** the ranking computed by HITS

- Solution:
 - A **site**, not a page, should be unit of voting power
 - If k pages on the same host link to a target page, the corresponding edges are assigned a **weight** $1/k$
- Matrix **E** is no more zero-one, but contains zeros and positive real numbers
- However, the properties of **A** and **H** remain the same

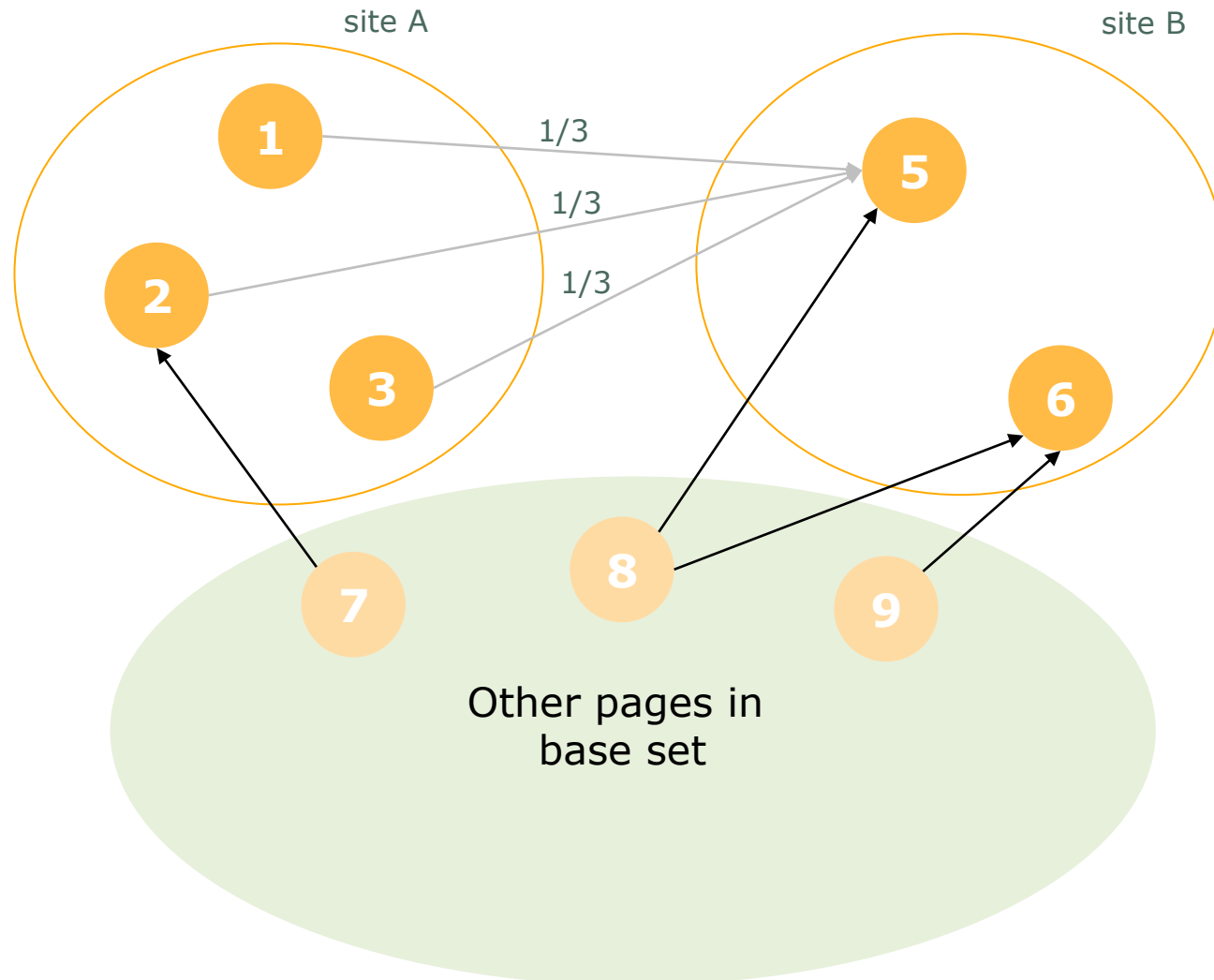
- HITS ignores links within pages in the same site



- To boost score of page 5, add several links from A to 5



- To fix this, add weights on the links



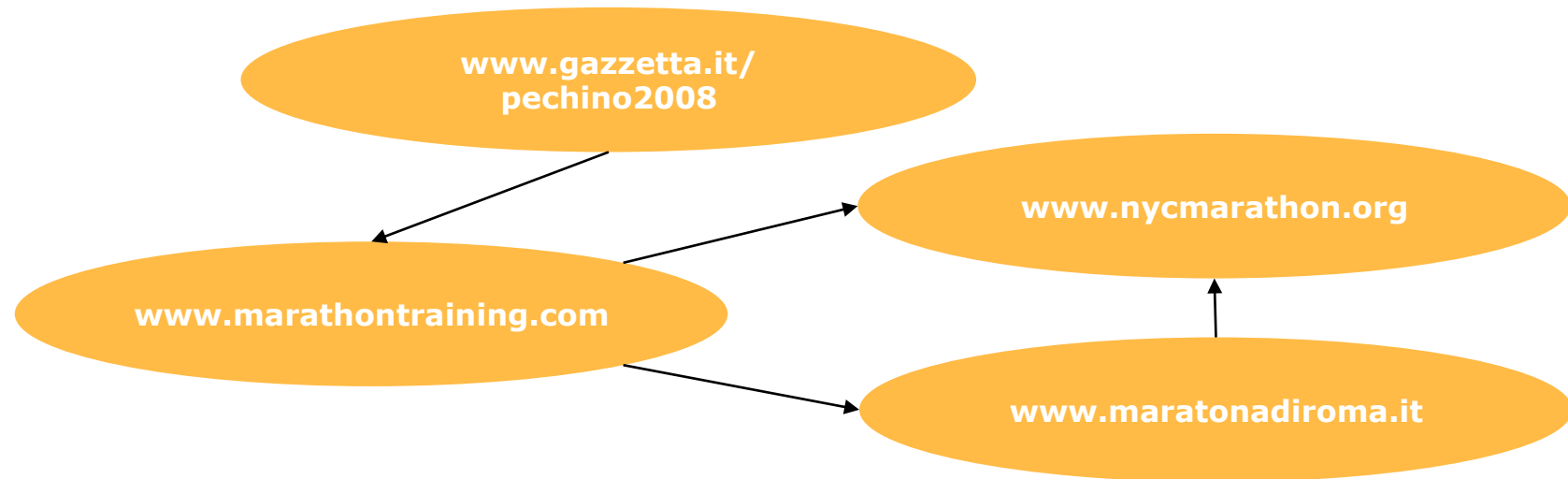
- The **expansion step** in HITS is meant to increase recall
- Given a query,
 - authorities might be missing from the root set, but hubs are present
 - example: the query “computer” might not include IBM web site
 - authorities are included, but hubs (needed to confer authority) are missing
- The expansion step relies on the **locality of content** on the Web
 - Given a page about a topic, following an outgoing link is more likely to find another page on the same topic than random sampling a page from the web

- Locality of content works in a **very short radius**
- Within a small number of links, the probability that all nodes have the same topic as the starting root set vanishes rapidly
- Even at radius one, severe contamination of the root set may occur
 - **topic generalization**: query “marathon” finds good hubs/authorities for “sports”
 - **topic drift**: due to pages containing many topics

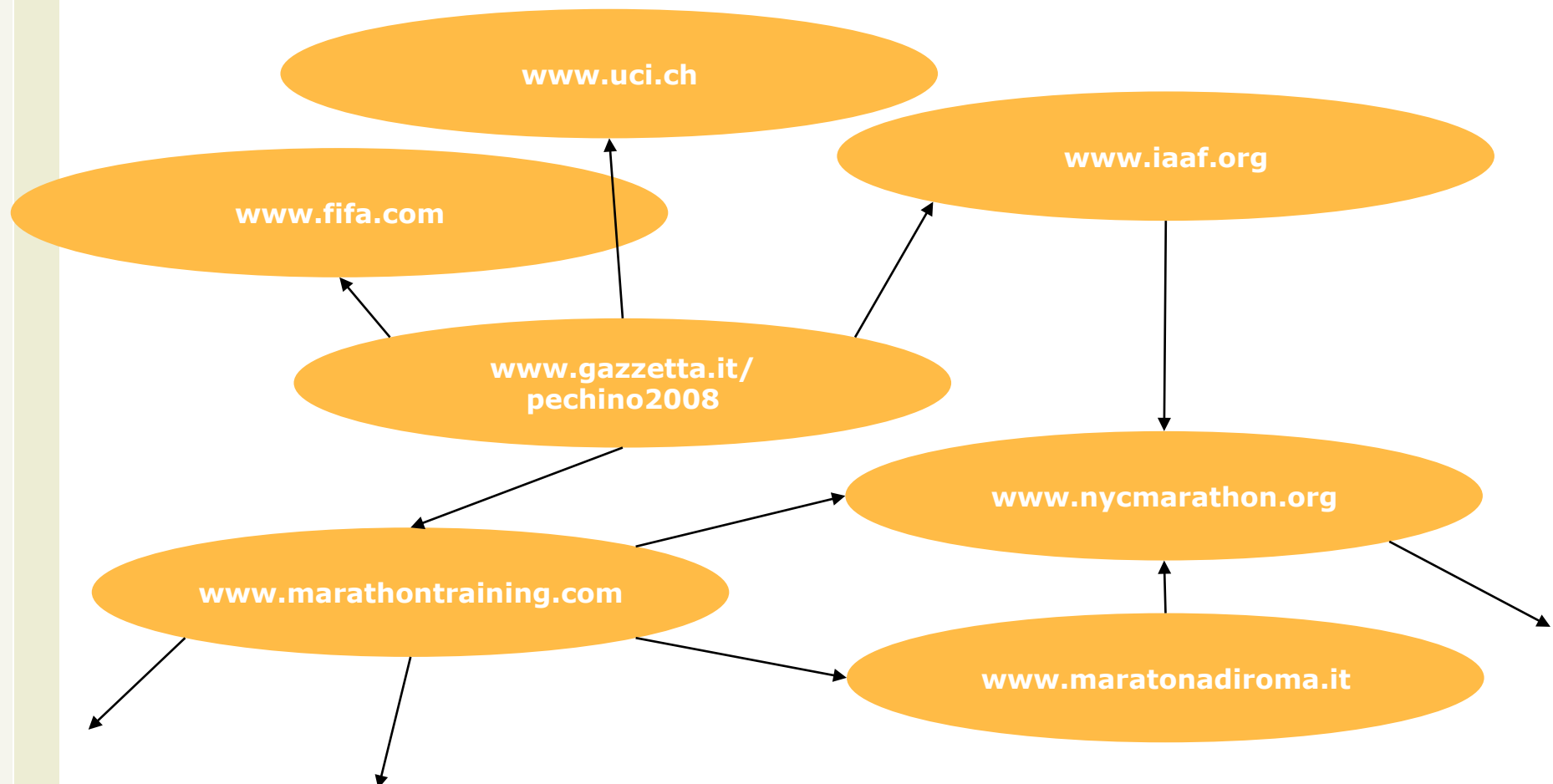
Solution:

- avoid indiscriminate expansion based only on links
- in the expansion step, remove pages that are too **dissimilar** (in terms of content) from the root set
 1. Compute the **term vectors** of the document in the root set (using TFIDF weighting)
 2. Compute the centroid of these vectors
 3. For any page to be added in the expansion step, compute **cosine similarity** between its vector and the centroid of the root set
 4. If the cosine is too large, discard the page

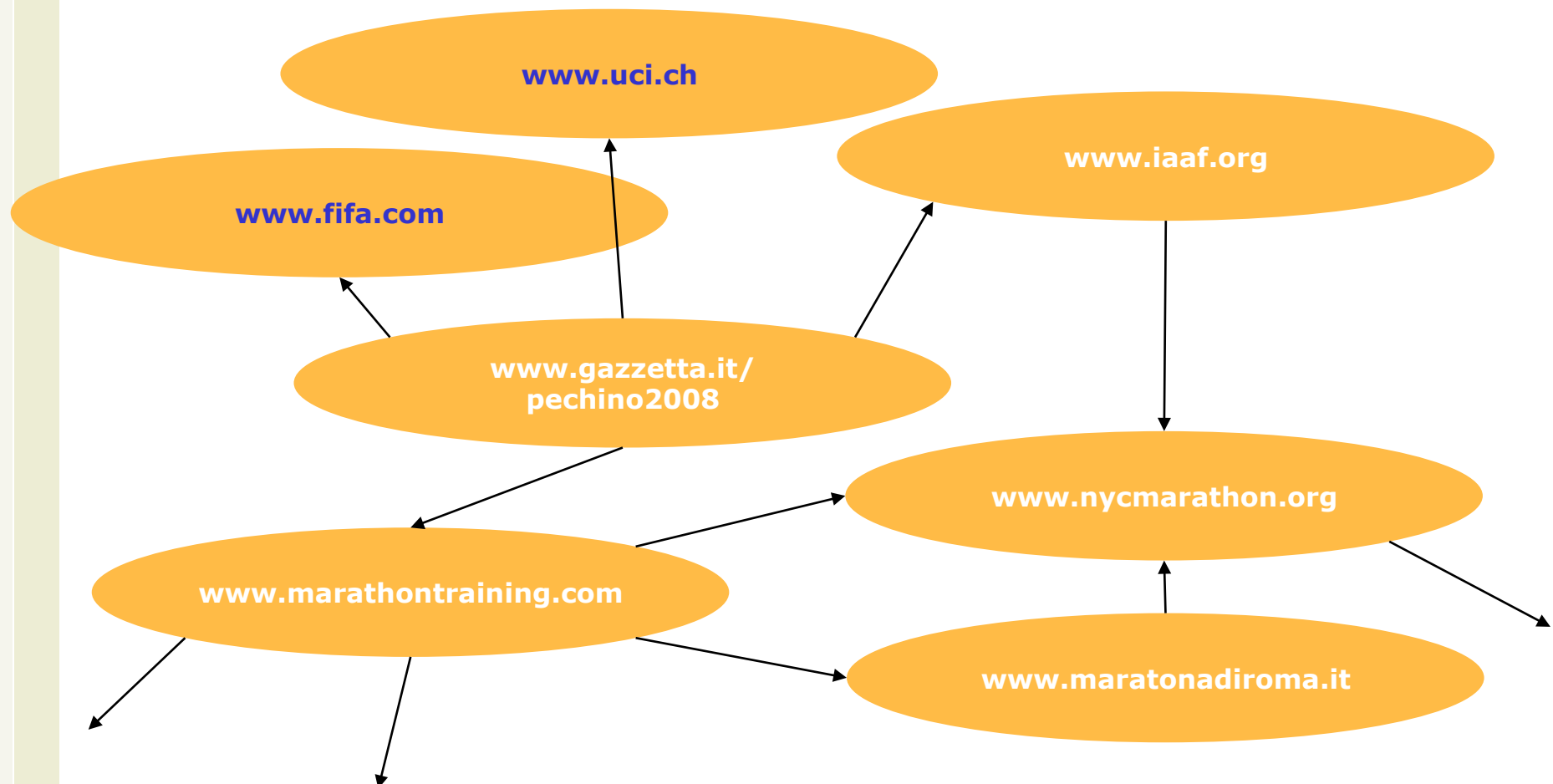
- Root set of the query “marathon”



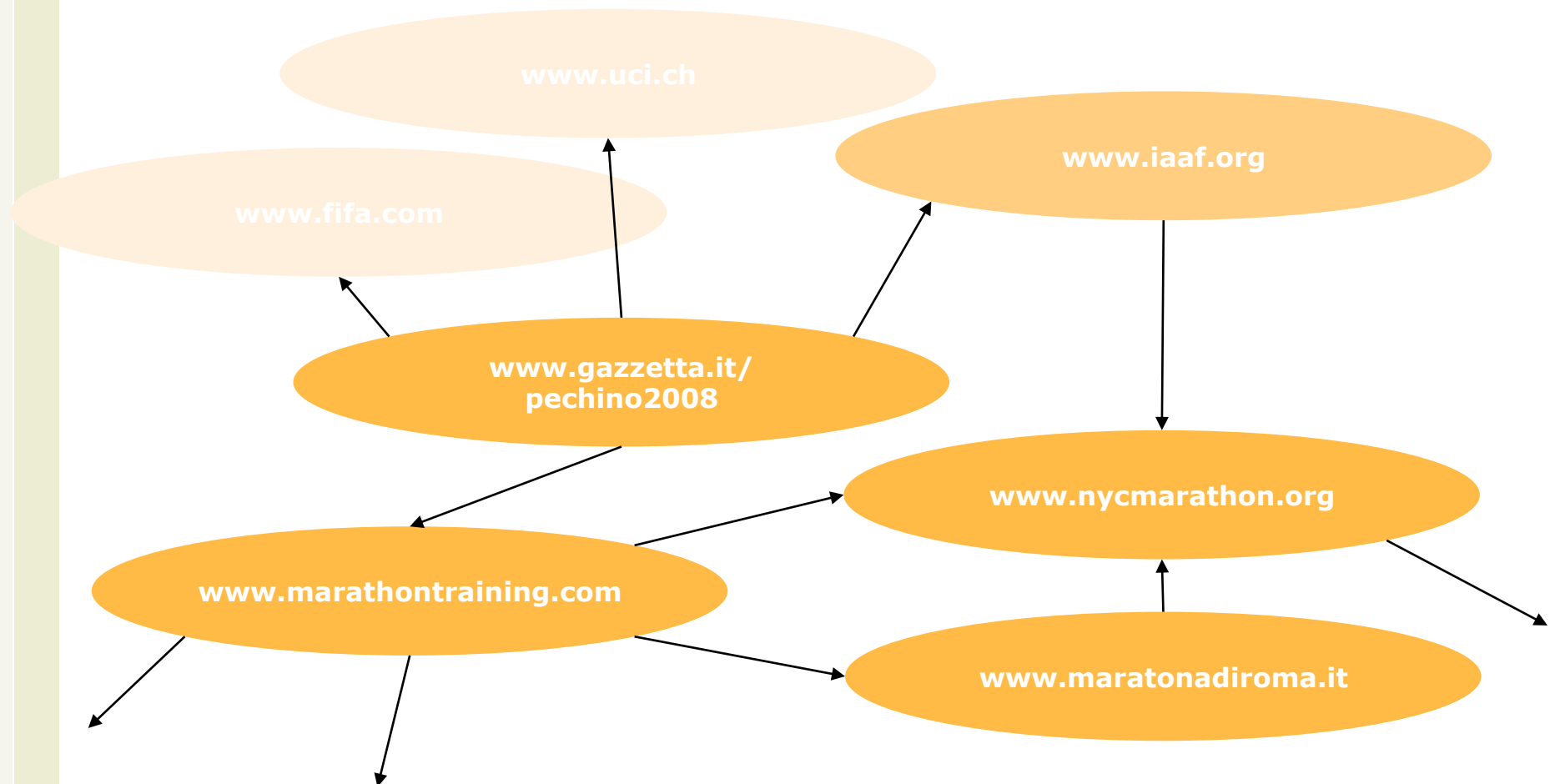
- Expansion step
 - both www.marathontraining.com and www.gazzetta.it would receive a high hub score



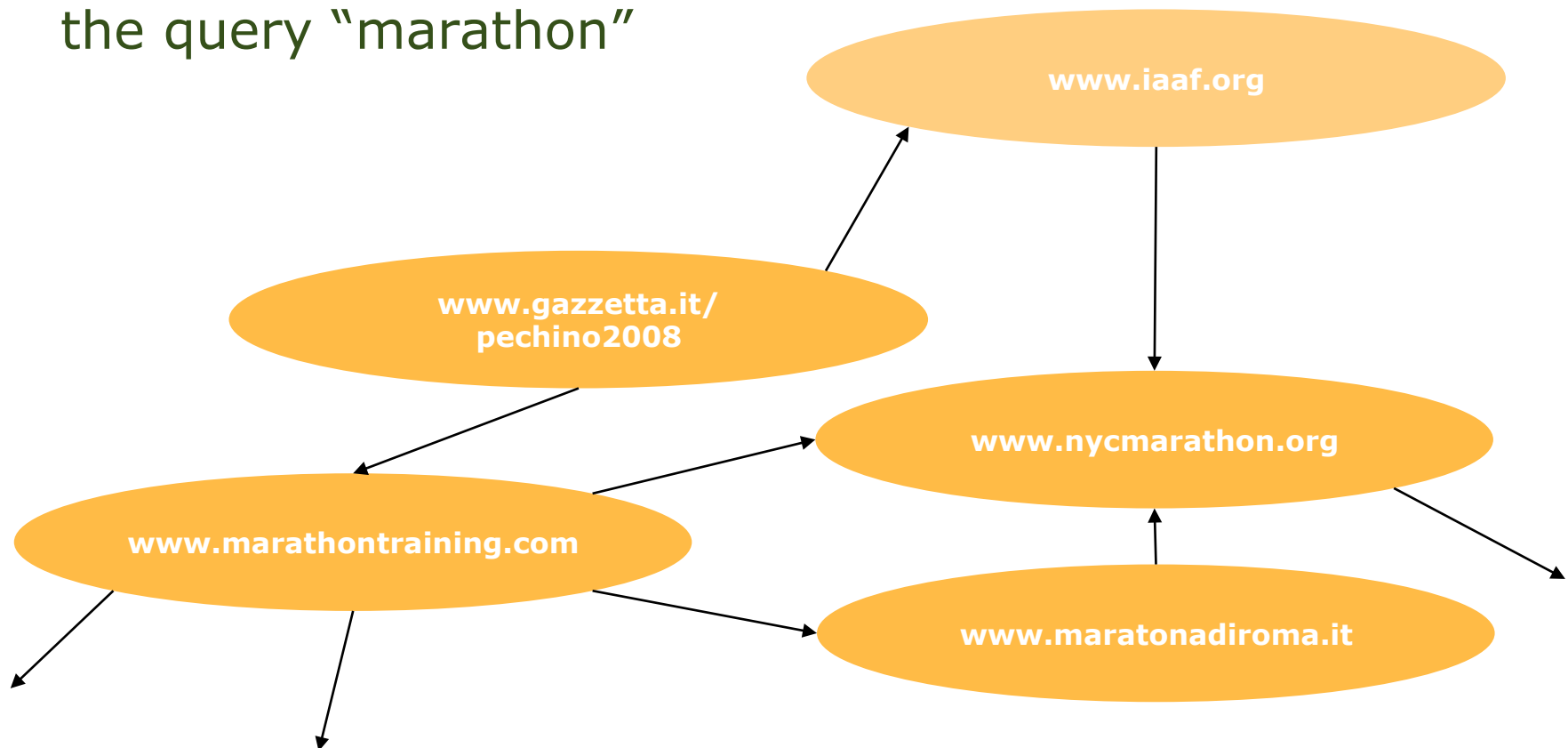
- Notice topic contamination:
 - www.fifa.com (football) and www.uci.ch (cycling)



- Use vector space models to compute cosine similarity between centroid of root set and added pages



- Delete pages with low cosine similarity and recompute scores
- www.marathontraining.com is now likely to receive a higher hub score than www.gazzetta.it with respect to the query "marathon"

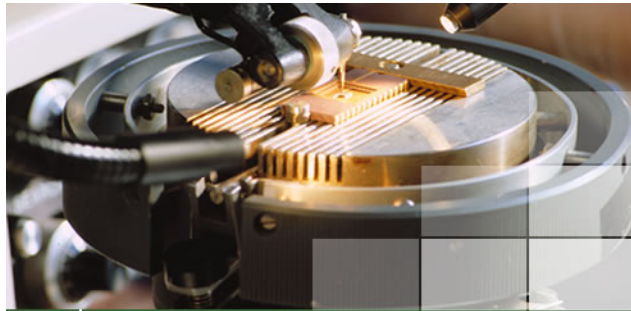


- HITS is query dependent, thus scores can be computed only at **query time**. PageRank is query independent and can be computed **offline**
- HITS computes two scores per page. Returning top ranking hubs is useful starting point for users exploring a topic
- PageRank is more robust to node insertion/deletion

- S. Chakrabarti, "Mining the Web", *Morgan Kaufman*, 2003
- C.D. Manning, P. Raghavan, H. Schutze, "An introduction to information retrieval", *Cambridge University Press*, 2008
- S. Dill, R. Kumar, K. S. Mccurley, S. Rajagopalan, D. Sivakumar, A. Tomkins, "Self-similarity in the web", *ACM Transactions on Internet Technology*, Vol. 2, Issue 3, , pp. 205 – 223, 2002
- J. Kleinberg, "Authoritative sources in a hyperlinked environment", *Journal of the ACM*, Vol. 46, Issue 5, pp. 604-632, 1999
- S. Brin, L. Page "The anatomy of a large-scale hypertextual Web search engine", *Proceedings of the seventh international conference on World Wide Web 7*: pp. 107-117, 1998

- K. Bryan, T. Leise, "The \$25,000,000,000 eigenvector. The linear algebra behind Google", *SIAM Review*, Vol. 48, Issue 3, pp. 569-81. 2006.
- A. Farahat, T. LoFaro, J. C. Miller, G.R. Lesley, A. Ward, "Authority Rankings from HITS, PageRank, and SALSA: Existence, Uniqueness, and Effect of Initialization", *SIAM Journal on Scientific Computing*, Vol. 27, Issue 4, pp. 1181-1201, 2006
- S. Ajay, J. Ekanayake, "Analysis Of The Usage Statistics Of Robots Exclusion Standard," IADIS International Conference WWW/Internet 2006 Murcia, Spain 5-8 October 2006

- Google: A Behind-the-scene Look (2005)
<http://www.uwtv.org/video/player.aspx?dwrid=3898>
- Ricardo Baeza-Yates: An Introduction to Web Retrieval (2011)
http://videolectures.net/essir2011_baeza_yates_retrieval/



 POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Search Computing Recommender Systems

Stefano Ceri

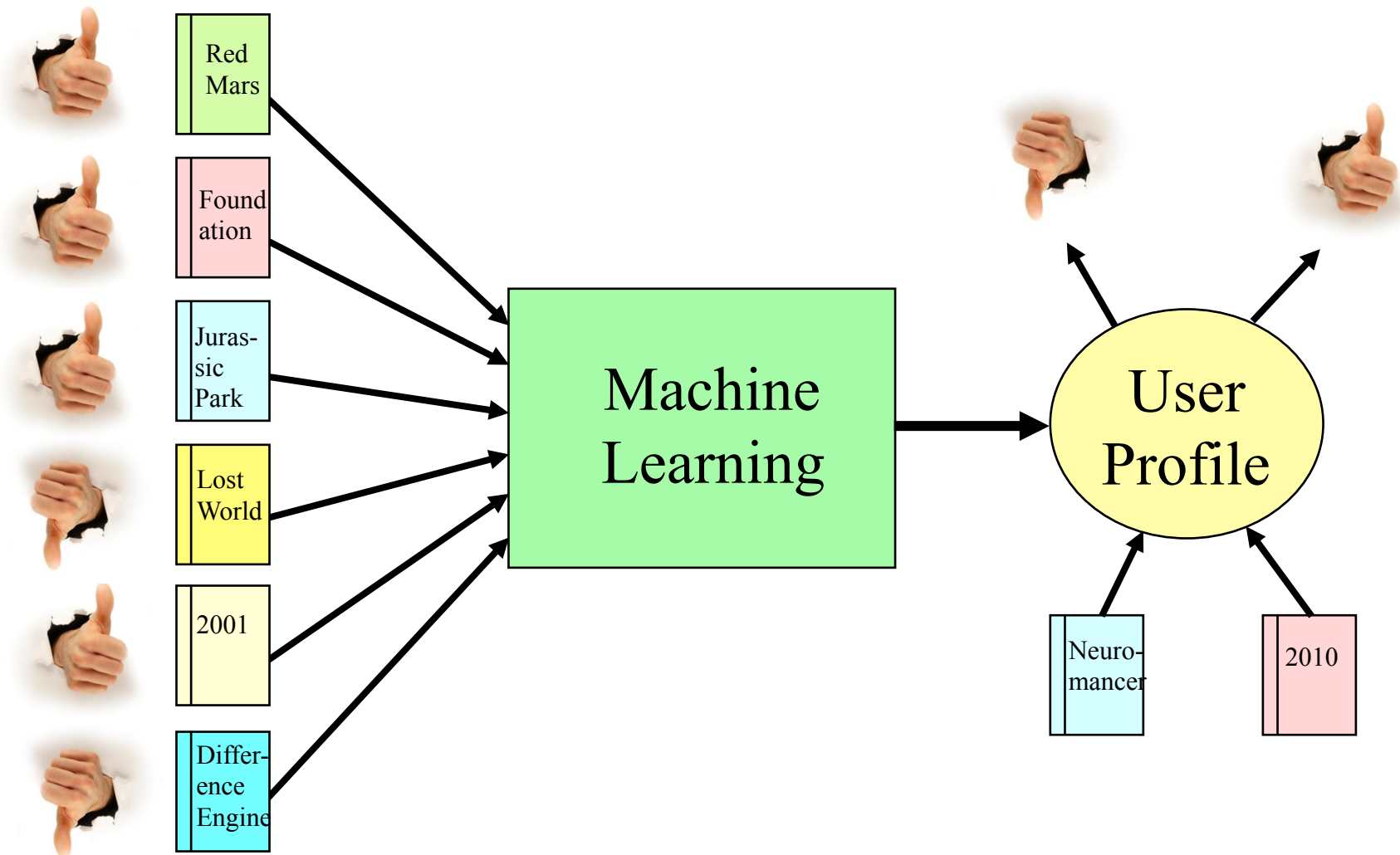
Original material authored by Marco Brambilla

Introduction to Recommender Systems

- Systems for recommending items (e.g. books, movies, CD's, web pages, newsgroup messages) to users based on examples of their preferences.
- Objective:
 - To propose objects fitting the user needs/wishes
 - To sell services (site visits) or goods
- Many search engines and on-line stores provide recommendations (e.g. Amazon, CDNow).
- Recommenders have been shown to substantially increase clicks (and sales).

Recommender Systems - Introduction

Book Recommender



Personalization

- Recommenders are instances of personalization software.
- Personalization concerns adapting to the individual needs, interests, and preferences of each user.
- Includes:
 - Recommending
 - Filtering
 - Predicting (e.g. form or calendar appt. completion)
- From a business perspective, it is viewed as part of Customer Relationship Management (CRM).

Machine Learning and Personalization

- Machine Learning can allow learning a *user model* or *profile* of a particular user based on:
 - Sample interaction
 - Rated examples
 - Similar user profiles
- This model or profile can then be used to:
 - Recommend items
 - Filter information
 - Predict behavior

Types of recommendation systems

1. Search-based recommendations
2. Category-based recommendations
3. Collaborative filtering
4. Clustering
5. Association rules
6. Information filtering
7. Classifiers

1. Search-based recommendations

- The only visitor types a search query
 - « data mining customer »
- The system retrieves all the items that correspond to that query
 - e.g. 6 books
- The system recommends some of these books based on general, non-personalized ranking (sales rank, popularity, etc.)

1.  **Mastering Data Mining: The Art and Science of Customer Relationship Management**
by Michael J. A. Berry, Gordon Linoff
(Paperback)
Average Customer Review: ★★★★★
Usually ships in 24 hours

Our Price: **\$50.00**



Add to cart

Search-based recommendations

- Pros:
 - Simple to implement

- Cons:
 - Not very powerful
 - Which criteria to use to rank recommendations?
 - Is it really « recommendations »?
 - The user only gets what he asked for

2. Category-based recommendations

- Each item belongs to one category or more.
- Explicit / implicit choice:
 - The customer select a category of interest (refine search, opt-in for category-based recommendations, etc.).
 - « [Subjects](#) > [Computers & Internet](#) > [Databases](#) > [Data Storage & Management](#) > [Data Mining](#) »
 - The system selects categories of interest on the behalf of the customer, based on the current item viewed, past purchases, etc.
- Certain items (bestsellers, new items) are eventually recommended

1.



[Developing Bioinformatics Computer Skills](#)

by Cynthia Gibas, Per Jambeck

Usually ships in 24 hours

O'Reilly & Associates

Paperback - 442 pages

1 Ed edition (April 15, 2001)

List Price: ~~\$34.95~~

Our Price: \$24.46

You Save: \$10.49 (30%)

[Click here for more info](#)



LOWER PRICES!

Category-based recommendations

- Pros:
 - Still simple to implement
- Cons:
 - Again: not very powerful, which criteria to use to order recommendations? is it really « recommendations »?
 - Capacity highly depends upon the kind of categories implemented
 - Too specific: not efficient
 - Not specific enough: no relevant recommendations

3. Collaborative filtering

- Collaborative filtering techniques « compare » customers, based on their previous purchases, to make recommendations to « similar » customers
- It's also called « social » filtering
- Follow these steps:
 1. Find customers who are similar (« nearest neighbors ») in term of tastes, preferences, past behaviors
 2. Aggregate weighted preferences of these neighbors
 3. Make recommendations based on these aggregated, weighted preferences (most preferred, unbought items)

Collaborative filtering

- Example: the system needs to make recommendations to customer C

	Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	

- Customer B is very close to C (he has bought all the books C has bought). Book 5 is highly recommended
- Customer D is somewhat close. Book 6 is recommended to a lower extent
- Customers A and E are not similar at all. Weight=0

Collaborative filtering

- Pros:
 - Extremely powerful and efficient
 - Very relevant recommendations
 - (1) The bigger the database, (2) the more the past behaviors, the better the recommendations

- Cons:
 - Difficult to implement, resource and time-consuming
 - What about a new item that has never been purchased?
Cannot be recommended
 - What about a new customer who has never bought anything? Cannot be compared to other customers
→ no items can be recommended

4. Clustering

- Another way to make recommendations based on past purchases of other customers is to cluster customers into categories
- Each cluster will be assigned « typical » preferences, based on preferences of customers who belong to the cluster
- Customers within each cluster will receive recommendations computed at the cluster level

Clustering

	Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	

- Customers B, C and D are « clustered » together. Customers A and E are clustered into another separate group
- « Typical » preferences for **CLUSTER** are:
 - Book 2, very high
 - Book 3, high
 - Books 5 and 6, may be recommended
 - Books 1 and 4, not recommended at all

Clustering

	Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	
Customer F			X		X	

- How does it work?
- Any customer that shall be classified as a member of **CLUSTER** will receive recommendations based on preferences of the group:
 - Book 2 will be highly recommended to Customer F
 - Book 6 will also be recommended to some extent

Clustering

- Problem: customers may belong to more than one cluster; clusters may overlap
- Predictions are then averaged across the clusters, weighted by participation

	Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	
Customer F			X		X	

	Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	
Customer F			X		X	

Clustering

- Pros:
 - Clustering techniques work on aggregated data: faster
 - It can also be applied as a « first step » for shrinking the selection of relevant neighbors in a collaborative filtering algorithm

- Cons:
 - Recommendations (per cluster) are less relevant than collaborative filtering (per individual)

5. Association rules

- **Clustering** works at a group (cluster) level
- **Collaborative filtering** works at the customer level
- **Association rules** work at the item level

Association rules

- Past purchases are transformed into relationships of common purchases

	Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	
Customer F			X		X	

		Also bought...					
		Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customers who bought...	Book 1				1	1	
	Book 2			2		1	1
	Book 3		2			2	
	Book 4	1					
	Book 5	1	1	2			
	Book 6		1				

Association rules

- These association rules are then used to make recommendations
- If a visitor has some interest in Book 5, he will be recommended to buy Book 3 as well
- Recommendations are constrained to some minimum levels of confidence
- What if recommendations can be made using more than one piece of information?
 - Recommendations are aggregated

		Also bought...					
		Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customers who bought...	Book 1				1	1	
	Book 2			2		1	1
	Book 3		2			2	
	Book 4	1					
	Book 5	1	1	2			
	Book 6		1				

Association rules

- Pros:
 - Fast to implement
 - Fast to execute
 - Not much storage space required
 - Not « individual » specific
 - Very successful in broad applications for large populations, such as shelf layout in retail stores

- Cons:
 - Not suitable if knowledge of preferences change rapidly
 - It is tempting to do not apply restrictive confidence rules
→ May lead to literally stupid recommendations

6. Information filtering

- **Association rules** compare items based on past purchases
- **Information filtering** compare items based on their content
- Also called « content-based filtering » or « content-based recommendations »
 - Can exploit syntactical information on objects (features)
 - But also semantic knowledge of objects (concepts/ontologies)

Information filtering

- What is the « content » of an item?
- It can be explicit « attributes » or « characteristics » of the item. For example for a film:
 - Action / adventure
 - Feature Bruce Willis
 - Year 1995
- It can also be « textual content » (title, description, table of content, etc.)
 - Several techniques exist to compute the distance between two textual documents

Information filtering

- How does it work?
 - A textual document is scanned and parsed
 - Word occurrences are counted (may be stemmed)
 - Several words or «tokens» are not taken into account: rarely used or «stop words»
 - Each document is transformed into a normed TFIDF vector, size N (*Term Frequency / Inverted Document Frequency*).
 - The distance between any pair of vector is computed

Information filtering

- An (unrealistic) example: how to compute recommendations between 8 books based only on their title?

- Books selected:
 - Building data mining applications for CRM
 - Accelerating Customer Relationships: Using CRM and Relationship Technologies
 - Mastering Data Mining: The Art and Science of Customer Relationship Management
 - Data Mining Your Website
 - Introduction to marketing
 - Consumer behavior
 - marketing research, a handbook
 - Customer knowledge management

	COUNT							
	building data mining applications for crm	Accelerating Customer Relationships: Using CRM and Relationship Technologies	Mastering Data Mining: The Art and Science of Customer Relationship Management	Data Mining Your Website	Introduction to marketing	consumer behavior	marketing research, a handbook	customer knowledge management
a							1	
accelerating		1						
and		1	1					
application	1							
art			1					
behavior						1		
building	1							
consumer						1		
crm	1	1						
customer		1	1					1
data	1		1	1				
for	1							
handbook							1	
introduction					1			
knowledge								1
management			1					1
marketing					1		1	
mastering			1					
mining	1		1	1				
of			1					
relationship		2	1					
research							1	
science			1					
technology		1						
the			1					
to					1			
using		1						
website				1				
your				1				

	TFIDF Normed Vectors							
	building	data	data mining	marketing	customer knowledge management			
accelerating	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
and	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
application	0.502	0.000	0.000	0.000	0.000	0.000	0.000	0.000
art	0.000	0.000	0.374	0.000	0.000	0.000	0.000	0.000
behavior	0.000	0.000	0.000	0.000	0.000	0.707	0.000	0.000
building	0.502	0.000	0.000	0.000	0.000	0.000	0.000	0.000
consumer	0.000	0.000	0.000	0.000	0.000	0.707	0.000	0.000
crm	0.344	0.296	0.000	0.000	0.000	0.000	0.000	0.000
customer	0.000	0.216	0.187	0.000	0.000	0.000	0.000	0.381
data	0.251	0.000	0.187	0.316	0.000	0.000	0.000	0.000
for	0.502	0.000	0.000	0.000	0.000	0.000	0.000	0.000
handbook	0.000	0.000	0.000	0.000	0.000	0.000	0.537	0.000
introduction	0.000	0.000	0.000	0.000	0.636	0.000	0.000	0.000
knowledge	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.763
management	0.000	0.000	0.256	0.000	0.000	0.000	0.000	0.522
marketing	0.000	0.000	0.000	0.000	0.436	0.000	0.368	0.000
mastering	0.000	0.000	0.374	0.000	0.000	0.000	0.000	0.000
mining	0.000	0.000	0.187	0.316	0.000	0.000	0.000	0.000
of	0.000	0.000	0.374	0.000	0.000	0.000	0.000	0.000
relationship	0.000	0.000	0.256	0.000	0.000	0.000	0.000	0.000
research	0.000	0.000	0.000	0.000	0.000	0.000	0.537	0.000
science	0.000	0.000	0.374	0.000	0.000	0.000	0.000	0.000
technology	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
the	0.000	0.000	0.374	0.000	0.000	0.000	0.000	0.000
to	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
using	0.000	0.432	0.000	0.000	0.000	0.000	0.000	0.000
website	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
your	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Mastering **Data Mining**:
The Art and Science
of Customer Relationship
Management

Data mining
your website

Data

0.187

0.316

Information filtering

- A customer is interested in the following book:
« Building data mining applications for CRM »
- The system computes distances between this book and the 7 others
- The « closest » books are recommended:
 - **#1:** Data Mining Your Website
 - **#2:** Accelerating Customer Relationships: Using CRM and Relationship Technologies
 - **#3:** Mastering Data Mining: The Art and Science of Customer Relationship Management
 - **Not recommended:** Introduction to marketing
 - **Not recommended:** Consumer behavior
 - **Not recommended:** marketing research, a handbook
 - **Not recommended:** Customer knowledge management

Information filtering

- Pros:
 - No need for past purchase history
 - Not extremely difficult to implement

- Cons:
 - « Static » recommendations
 - Not efficient if content is not very informative
e.g. information filtering is more suited to recommend technical books than novels or movies

7. Classifiers

- **Classifiers** are general computational models
- They may take in inputs:
 - Vector of item features (action / adventure, Bruce Willis)
 - Preferences of customers (like action / adventure)
 - Relations among items
- They may give as outputs:
 - Classification
 - Rank
 - Preference estimate
- That can be a neural network, Bayesian network, rule induction model, etc.
- The classifier is trained using a training set

Classifiers

- Pros:
 - Versatile
 - Can be combined with other methods to improve accuracy of recommendations
- Cons:
 - Need a relevant training set

Recommender Systems (some) References

- J. Ben Schafer, Joseph A. Konstan, John Riedl. E-Commerce Recommendation Applications. Data Mining and Knowledge Discovery, Springer, 2004.
- P. Resnick, H. R. Varian. Recommender Systems, special session of Communications of the ACM, 1997.
- N. Good, B. Schafer, J. Konstan, A. Borchers, J. Herlocker, B. Sarwar, J. Riedl. Combining Collaborative Filtering with Personal Agents for Better Recommendations. Sixteenth National Conference on Artificial Intelligence (AAAI '99), 1999.
- M. McLaughlin, J. Herlocker. A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience. Conference on Research and Development in Information Retrieval SIGIR 2004.
- M. Brambilla, C. Tziviskou. Modeling Ontology-Driven Personalization of Web Contents, Proceedings of ICWE 2008, IEEE Press, July 2008, Yorktown Heights, USA, pp. 247-260.