Information Retrieval

Information Retrieval on the Web

Ulf Leser

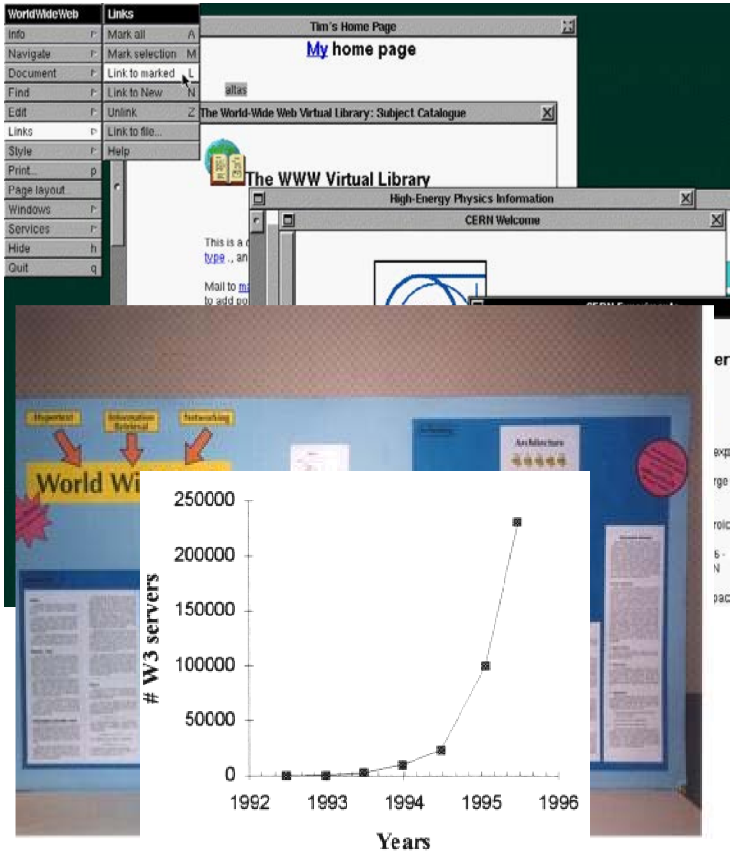
Content of this Lecture

- The Web
- Web Crawling
- Exploiting Web Structure for IR
- A Different Flavor: WebSQL

- Much of today's material is from:
  Chakrabarti, S. (2003). *Mining the Web: Discovering Knowledge from Hypertext Data*: Morgan Kaufmann Publishers.

The World Wide Web

- 1965: Hypertext: „A File Structure for the Complex, the Changing, and the Indeterminate“ (Ted Nelson)
- 1969: ARPANET
- 1971: First email
- 1978: TCP/IP
- 1989: "Information Management: A Proposal" (Tim Berners-Lee, CERN)
- 1990: First Web Browser
- 1991: WWW Poster
- 1993: Browsers (Mosaic->Netscape->Mozilla)
- 1994: W3C creation
- 1994: Crawler: “World Wide Web Wanderer“
- 1995: Search engines such as Excite, Infoseek, AltaVista, Yahoo, ...
- 1997: HTML 3.2 released (W3C)
- 1999: HTTP 1.1 released (W3C)
- 2000: Google, Amazon, Ebay, ...



See http://www.w3.org/2004/Talks/w3c10-HowItAllStarted

HTTP: Hypertext Transfer Protocol

- Stateless, very simple protocol
- Many clients (e.g. browsers, telnet, …) talk to one server
  - GET: Request a file (e.g., a web page)
  - POST: Request file and transfer data block
  - PUT: Send file to server (deprecated, see WebDAV)
  - HEAD: Request file metadata (e.g. to check currentness)
- HTTP 1.1: Send many requests over one TCP connection
- Transferring parameters: URL rewriting or POST method
- Keeping state: URL rewriting or cookies
- Example
  - ```
    GET /wiki/Spezial:Search?search=Katzen&go=Artikel HTTP/1.1
    Host: de.wikipedia.org
    ```
    ```

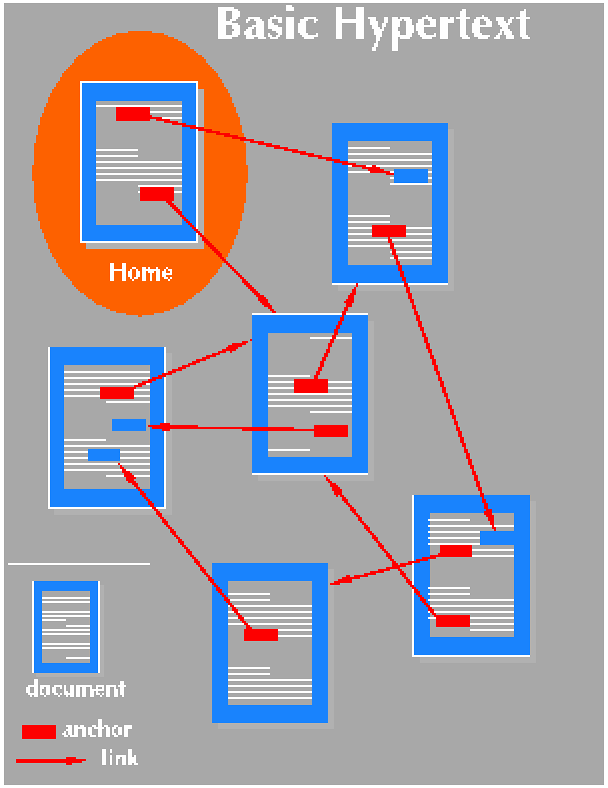
HTML: Hypertext Markup Language

- Web pages originally are ASCII files with markup
  - Things change(d): Images, SVG, JavaScript, Web2.0/AJAX, ...
- HTML: strongly influenced by SGML, but much simpler
- Focus on layout; no semantic information

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN„
  „…/strict.dtd">
<html>
  <head>
  <title>
        Titel of web page
  </title>
  <!– more metadata -->
  </head>
  <body>
        Content of web page
  </body>
</html>
```

Hypertext

- Most interesting feature of HTML: Links between pages
- The concept is old: Hypertext
  – Generally attributed to Bush, V. (1945). As We May Think. *The Atlantic Monthly*
  – Suggests "Memex: A system of storing information linked by pointers in a graph-like structure"
- Links have an anchor and a target
- Allows for associative browsing

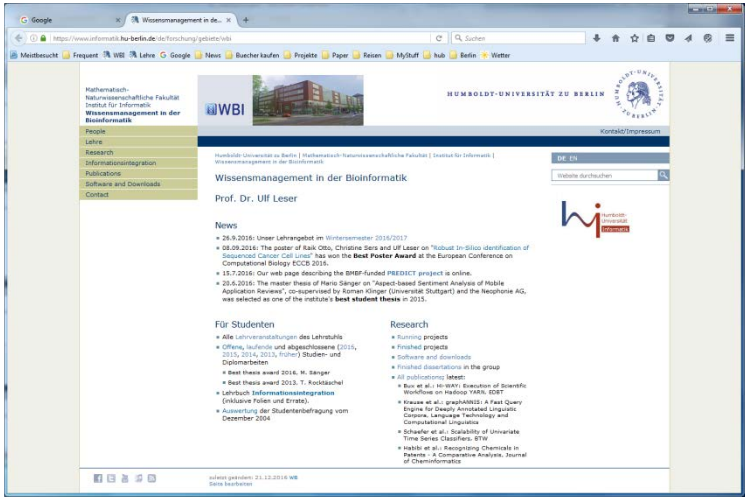


http://www.w3.org

```
<a href=„05_ir_models.pdf“>IR Models</a>:
Probabilistic and vector space model
```

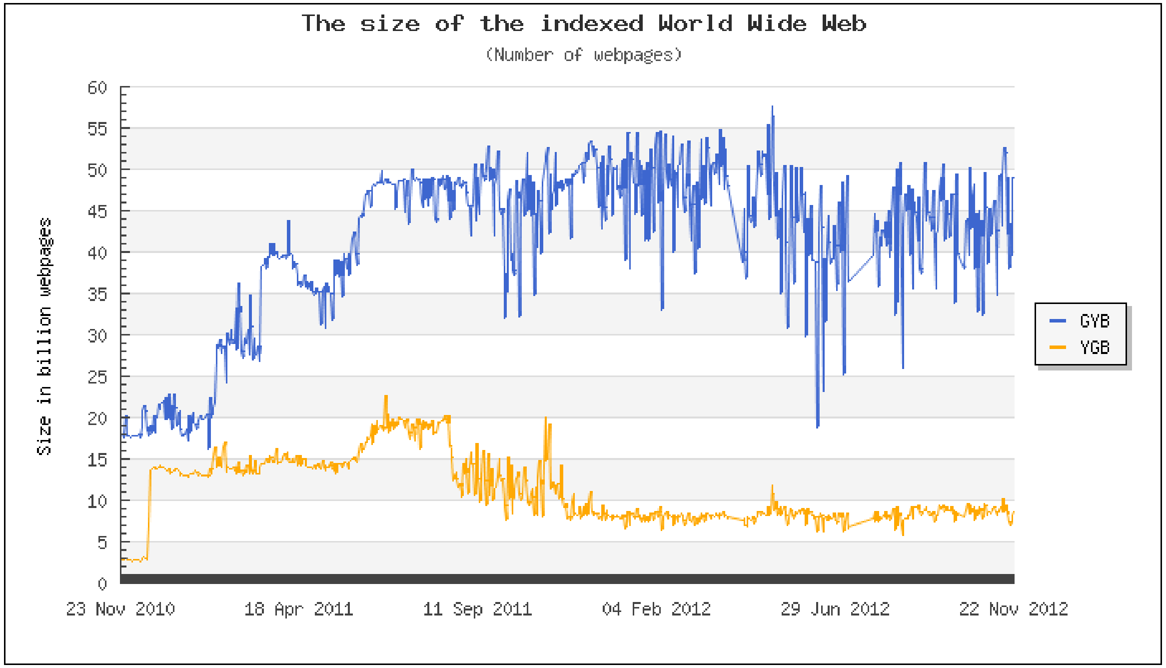
Deep Web

- Most of the data “on” the web is not stored in HTML
- Surface web: Static web pages = files on a web server
- Deep web: Accessible only through forms, logins, ...
  - Most content of databases (many are periodically dumped)
  - Accessible through CGI scripts, servlets, web services, ...
- Crawls only reach the surface web
  - Plus individual solutions/contracts for specific information: product catalogues, news, ...
- Deep != computer generated
  - Many systems create pages only when accessed
  - Access by ordinary link: Surface web



It’s Huge

- Jan 2007: Number of hosts estimated 100 - 500 Million
- 2005: App. 12M web pages (Guli, Signorini, WWW 2005)
- 2013: App. 13 Trillion web pages (www.factshunt.com)



Source: http://www.worldwidewebsize.com/

Accesses per Month (as of 2012)

- Google: 88 billion per month
  - Means: ~3 billion per day
  - 12-fold increase over 7 years
- Twitter: 19 billion per month
- Yahoo: 9.4 billion per month
- Bing: 4.1 billion per month

Source: www.searchengineland.com

Zuckerberg says Facebook processes ‘a billion searches per day’

Published on September 12, 2012 by Brafton Editorial

in 3 11 f 0 g+ 1 0 0

Marketers usually talk about the intersection of search and social marketing on Google SERPs, but Facebook wants to remind companies and users that it has an important search utility. At TechCrunch Disrupt on Tuesday, Facebook CEO Mark Zuckerberg said that the social network processes more than one billion searches every day. While most users look for specific pages and profiles using Facebook search, the results also

Search Engines World Wide

Searching the Web

- In some sense, the Web is a single, large corpus
- But searching the web is different from traditional IR
  - Recall is nothing
    - Most queries are too short to be discriminative for a corpus of that size
    - Usual queries generate very many hits: Information overload
    - We never know “the” web: A moving target
  - Ranking is more important than high precision
    - Users rarely go to results page 2
  - Intentional cheating: Precision of search badly degraded
  - Mirrors: Concept of “unique” document is not adequate
  - Much of the content is non-textual
  - Documents are linked

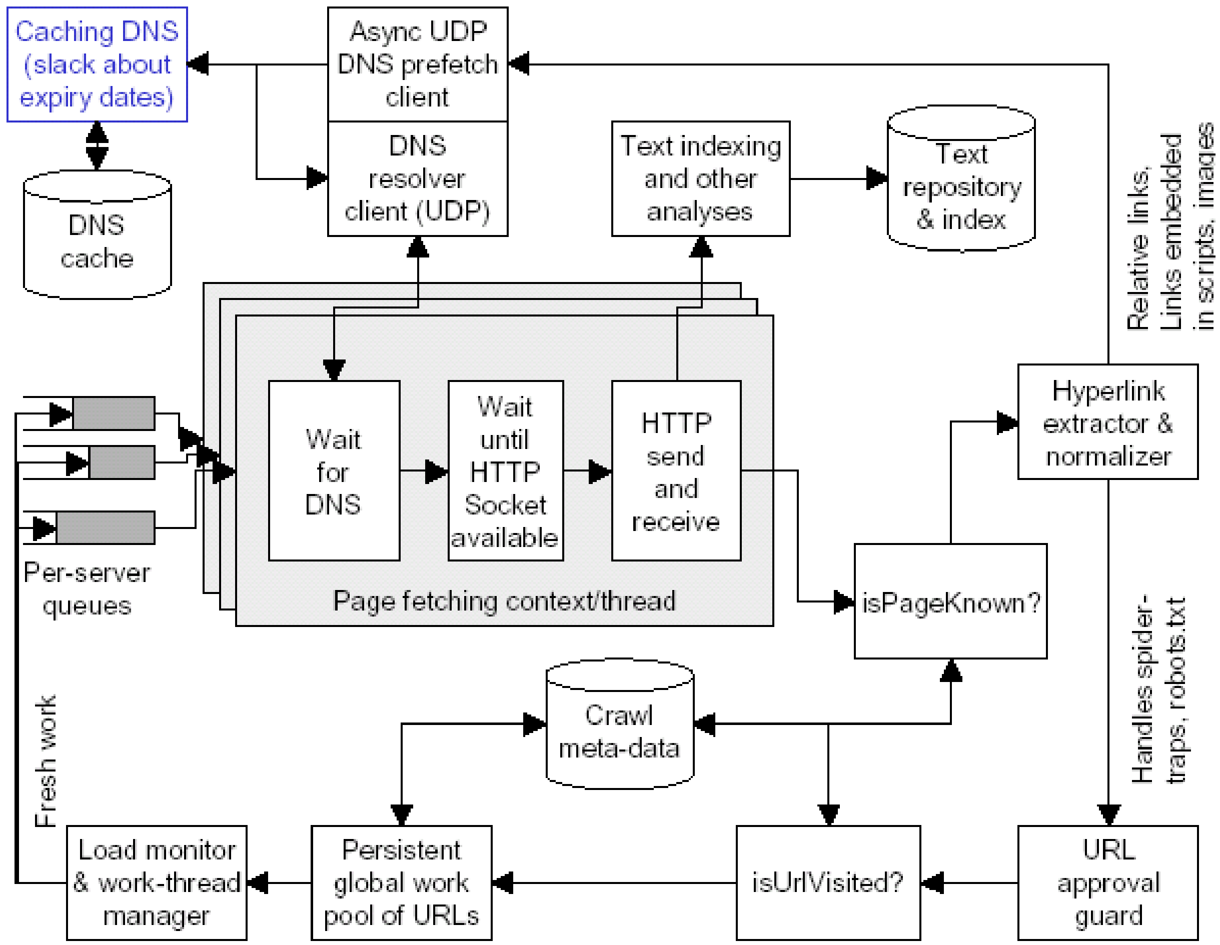
Content of this Lecture

- The Web
- Web Crawling
- Exploiting Web Structure for IR
- A Different Flavor: WebSQL

Web Crawling

- We want to search a constantly changing set of documents
  - Note: www.archive.org: The Wayback Machine: „Browse through 150 billion pages archived from 1996 to a few months ago....“
- There is no list of all web pages
- Solution
  - Start from a given set of URLs
  - Iteratively fetch and scan web pages for outlinking URLs
  - Put links in fetch queue sorted by some magic
  - Take care of not fetching the same page again and again
    - Relative links, URL-rewriting, multiple server names, ...
  - Repeat forever

Architecture of a Web Crawler



Issues

- Key trick: Parallelize everything
  - Use multiple DNS servers (and cache resolutions)
  - Use many, many download threads
  - Use HTTP 1.1: Multiple fetches over one TCP connection
- Take care of your bandwidth and of load on remote servers
  - Do not overload server (DoS attack)
  - Robot-exclusion protocol
- Usually, bandwidth and IO-throughput are more severe bottlenecks than CPU consumption

More Issues

- Before analyzing a page, check if redundant (checksum)
- Re-fetching a page is not always bad
  - Pages may have changed
  - Revisit after certain period, use HTTP HEAD command
  - Individual periods can be adjusted automatically
    - Sites / pages usually have a rather stable update frequency
- Crawler traps, “google bombs”
  - Pages which are CGI scripts generating an infinite series of different URLs all leading to the same script
  - Difficult to avoid
    - Overly long URLs, special characters, too many directories, ...
    - Keep black list of servers

Focused Crawling

- One often is interested only in a certain topic
- Supervised domain-specific web crawling
  - Build a classifier assessing the relevance of a crawled page based on its textual input
  - Only put out-links of relevant documents in crawler queue
- Alternatives
  - Classify each link separately
  - Also follow irrelevant links, but not for too long

Content of this Lecture

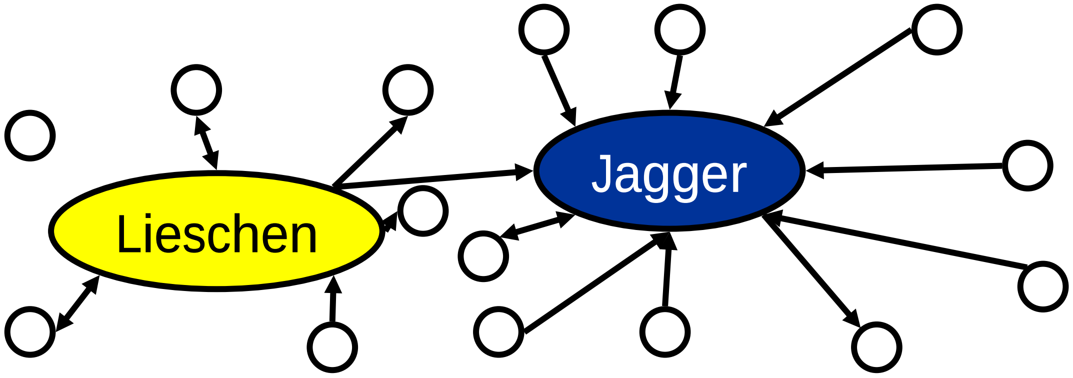
- The Web
- Web Crawlers
- Exploiting Web Structure for IR
  - Prestige in networks
  - Page Rank
  - HITS
- A Different Flavor: WebSQL

Ranking and Prestige

- Classical IR ranks docs according to content and query
  - On the web, many queries generate too many “good” matches
  - “Cancer”, “daimler”, “car rental”, “newspaper”, ...
- Why not use other features?
  - Rank documents higher whose author is more famous
  - Rank documents higher whose publisher is more famous
  - Rank documents higher that have more references
  - Rank documents higher that are cited by documents which would be ranked high in searchers
  - Rank docs higher which have a “higher prestige”
- Prestige in social networks: The prestige of a person depends on the prestige of its friends

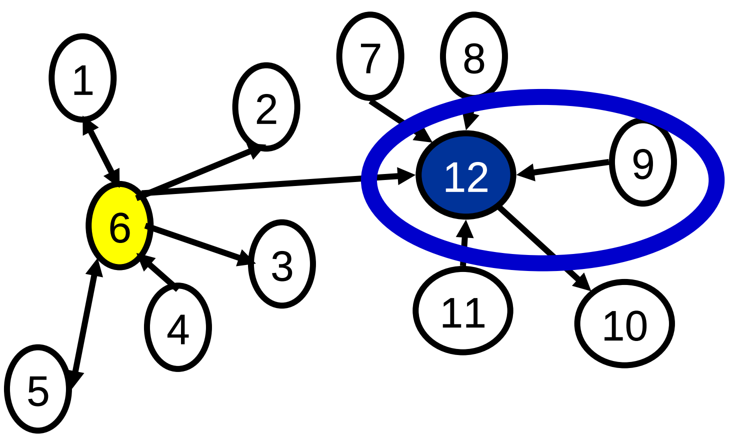
Prestige in a Network

- Consider a network of people, where a directed edge (u,v) indicates that person u knows person v
- Modeling prestige: A person “inherits” the prestige from all persons who known him/her
  - Your prestige is high if you are known by many other famous people, not the other way round
- Formally: Your prestige is the sum of the prestige values of people that know you



Computing Prestige

- Let E by the adjacency matrix, i.e., E[u,v]=1 if u knows v
- Let p be the vector storing the prestige of all nodes
  - Initialized with some small constants
- If we compute $p' = E^T * p$, p' is a new prestige vector which considers the prestige of all "incoming" nodes



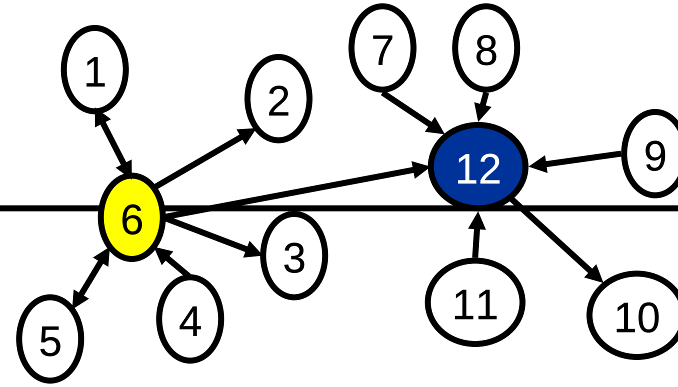
E^T

	1	2	3	4	5	6	7	8	9	0	1	2
1						1						
2						1						
3						1						
4												
5						1						
6	1			1	1							
7												
8												
9												
0												1
1												
2						1	1	1	1		1	

Iterative Multiplications

- Computing $p''=E^T*p'=E^T*E^T*p$ also considers indirect influences
- Computing $p'''=E^T*p''=E^T*E^T*E^T*p$ also ...
- We seek a prestige vector such that: $p=E^T*p$
- Note: Under some circumstances, iteratively multiplying E^T will make p converge
  - Math comes later

Example



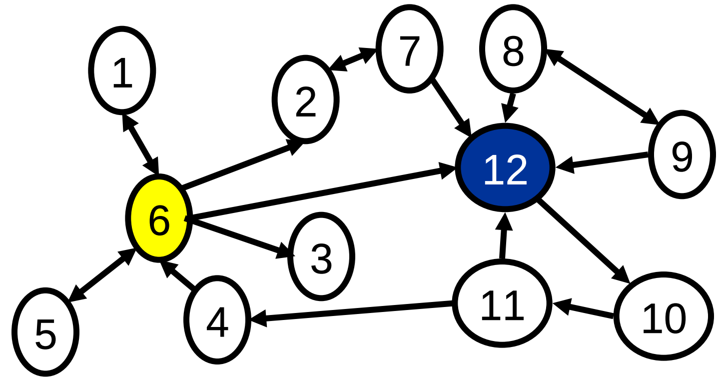
- Start with $p_0=(1,1,1,...)$
- Iterate: $p_{i+1}=E^T*p_i$
- Example
  - $p_1=(1,1,1,0,1,3,0,0,0,1,0,5)$
    - 6 and 12 are cool
  - $p_2=(3,3,3,0,3,2,0,0,0,5,0,3)$
    - To be known by 6/12 is cool
    - To be known be 4,7,8,.. doesn't help much

	1	2	3	4	5	6	7	8	9	0	1	2
1						1						
2						1						
3						1						
4												
5						1						
6	1			1	1							
7												
8												
9												
0												1
1												
2						1	1	1	1		1	

- Hmm – we punish "social sinks" quite hard...
  - Nodes who are not known by anybody

Example 2

- Modified graph: Every node has at least one incoming link
- Start with $p_0=(1,1,1,...)$
- Iterate
  - $p_1=(1,1,1,0,1,3,0,0,0,1,0,5)$
  - $p_2=(3,3,3,0,3,2,1,0,0,5,1,3)$
  - $p_3=(2,3,2,1,2,8,3,...$
  - ...
- Hmm – numbers grow to infinity
- Must be repaired



	1	2	3	4	5	6	7	8	9	0	1	2
1						1						
2						1	1					
3						1						
4											1	
5						1						
6	1			1	1							
7		1										
8									1			
9								1				
0												1
1										1		
2						1	1	1	1		1	

Prestige in Hypertext IR (= Web Search)

- PageRank uses the number of incoming links
  - Scores are query independent and can be pre-computed
  - Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The PageRank Citation Ranking: Bringing Order to the Web: Unpublished manuscript, Stanford University.
- HITS distinguishes authorities and hubs wrt. a query
  - Thus, scores cannot be pre-computed
  - Kleinberg, J. M. (1998). Authoritative Sources in a Hyperlinked Environment. ACM-SIAM Symposium on Discrete Mathematics.
- Many more suggestions
  - "Bharat and Henzinger" model ranks down connected pages which are very dissimilar to the query
  - "Clever" weights links wrt. the local neighborhood of the link in a page (anchor + context)
  - ObjectRank and PopRank rank objects (on pages), including different types of relationships

Content of this Lecture

- Searching the Web
- Search engines on the Web
- Exploiting the web structure
  - Prestige in networks
  - Page Rank
  - HITS
- A different flavor: WebSQL

PageRank Algorithm

- Major breakthrough: Ranking of Google was much better than that of other search engines
  - Before: Ranking only with page content and length of URL
    - The longer, the more specialized
- Ranking of current search engines result from prestige value, IR score, ...
- Computing PageRank for billions of pages requires more tricks than we present here
  - Especially approximation

Random Surfer Model

- Another view on "prestige"
- Random Surfer
  - Assume a "random" surfer S taking all decision by chance
  - S starts from a random page ...
  - ... picks and clicks a link from that page at random ...
  - ... and repeats this process forever
- At any point in time: What is the probability p(v) for S being on a page v?
  - After arbitrary many clicks? Starting from an arbitrary web page?

Random Surfer Model Math

- After one click, S is in v with probability

$$p_1(v) = \sum_{(u,v)\in V} \frac{p_0(u)}{|u|} = \sum_u E'[u,v] * p_0(u)$$

  - With |u| = "# of links outgoing from u" and E'[u,v]=E[u,v]/|u|
  - Components: Probability to start in a page u with a link to v and the probability of following link u→v

- Condensed representation for all v

$$\vec{p}_1 = E'^T * \vec{p}_0$$

Eigenvectors and PageRank

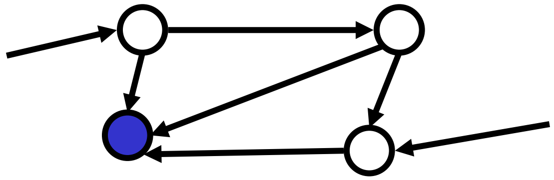
- Iteration: $p_{i+1}=E'^T*p_i$
- We search fixpoint: $p=E'^T*p$
- Recall: If $Mx-\lambda x=0$ for $x\neq 0$, then λ is called an Eigenvalue of M and x is his associated Eigenvector
- Transformation yields $\lambda x=Mx$
- We are almost there
  - Eigenvectors for Eigenvalue $\lambda=1$ solve our problem
  - But these do not always exist

Perron-Frobenius Theorem

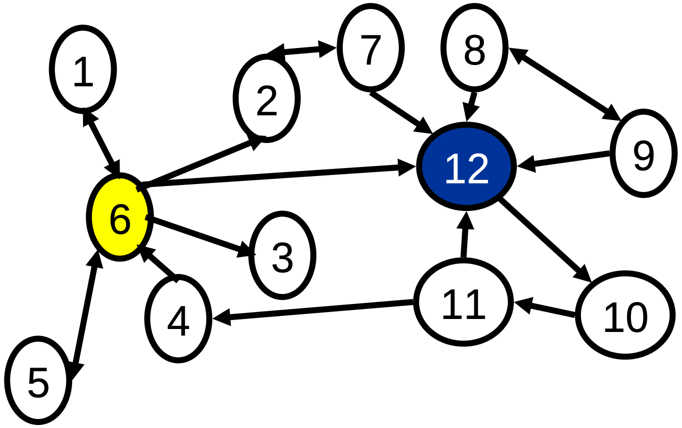
- When do Eigenvectors $\lambda=1$ exist?
- Let M be a stochastic quadratic irreducible aperiodic matrix
  - Quadratic: m=n
  - Stochastic: $M[i,j] \geq 0$, all column sums are 1
  - Irreducible: If we interpret M as a graph G, then every node in G can be reached by any other node in G
  - Aperiodic: $\exists n \in N$ such that for every u,v there is a path of length n between u and v
- For such M, the largest Eigenvalue is $\lambda=1$
  - Its corresponding Eigenvector x satisfies $x = Mx$
  - Can be computed using our iterative approach
    - PowerIteration Method

Real Links versus Mathematical Assumptions

1. The sum of the weights in each column equals 1
   - Not yet achieved – web pages may have no outgoing edge
   - "Rank sinks"



2. The matrix E' is irreducible
   - Not yet achieved – the web graph is not at all strongly connected
   - For instance, no path between 3 and 4



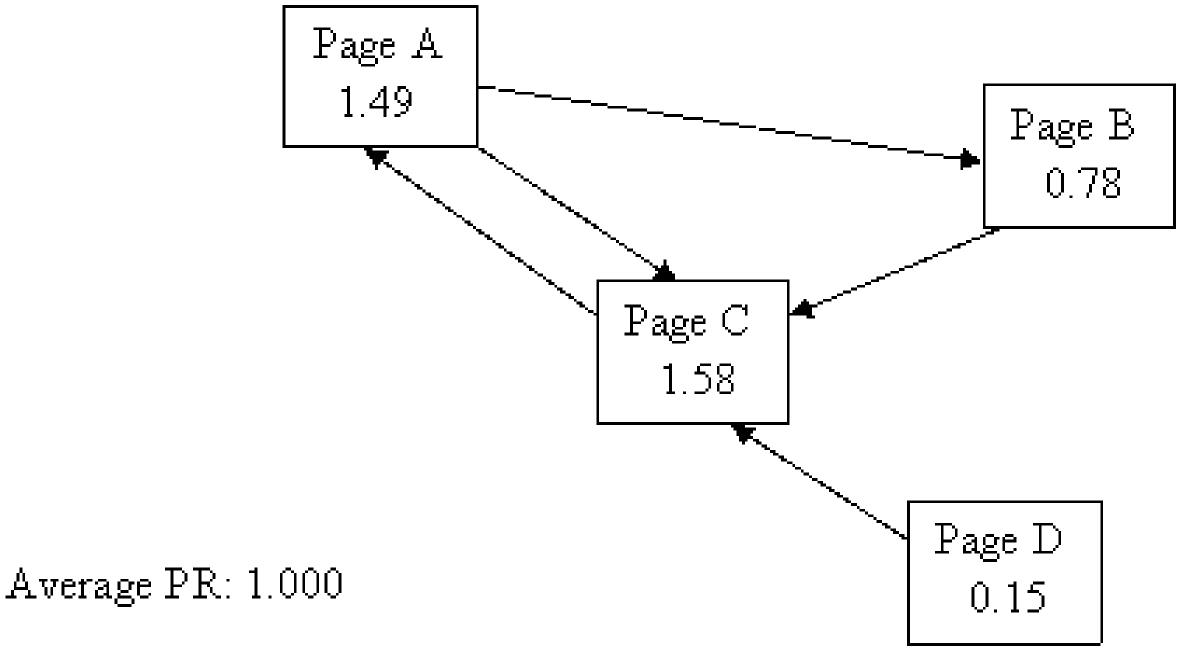
Repair

- Simple repair: We give every possible link a fixed, very small probability
  - No more 0 in E
  - If E'[u,v]=0, set E'[u,v]=1/n, with n~"total number of pages"
  - This also makes the matrix aperiodic (with n=1)
  - Normalize such that all column sums are 1
- Intuitive explanation: Random restarts
  - We allow our surfer S at each step, with a small probability, to jump to an arbitrary other page (instead of following a link)
  - Jump probability is the higher, the less outgoing links a page has

PageRank

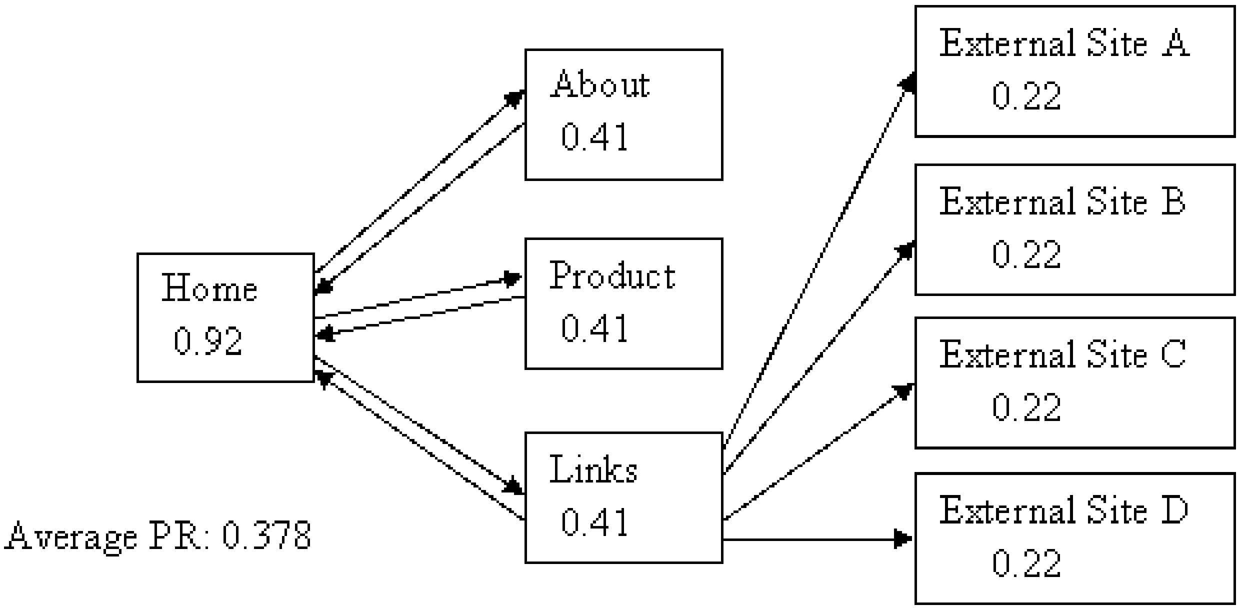
- Practice: Iterate until changes become small
  - We stop before fixpoint is reached
  - Faster at the cost of accuracy
- The original paper reports that ~50 iterations sufficed for a crawl of 300 Million links

Example 1 [Nuer07]



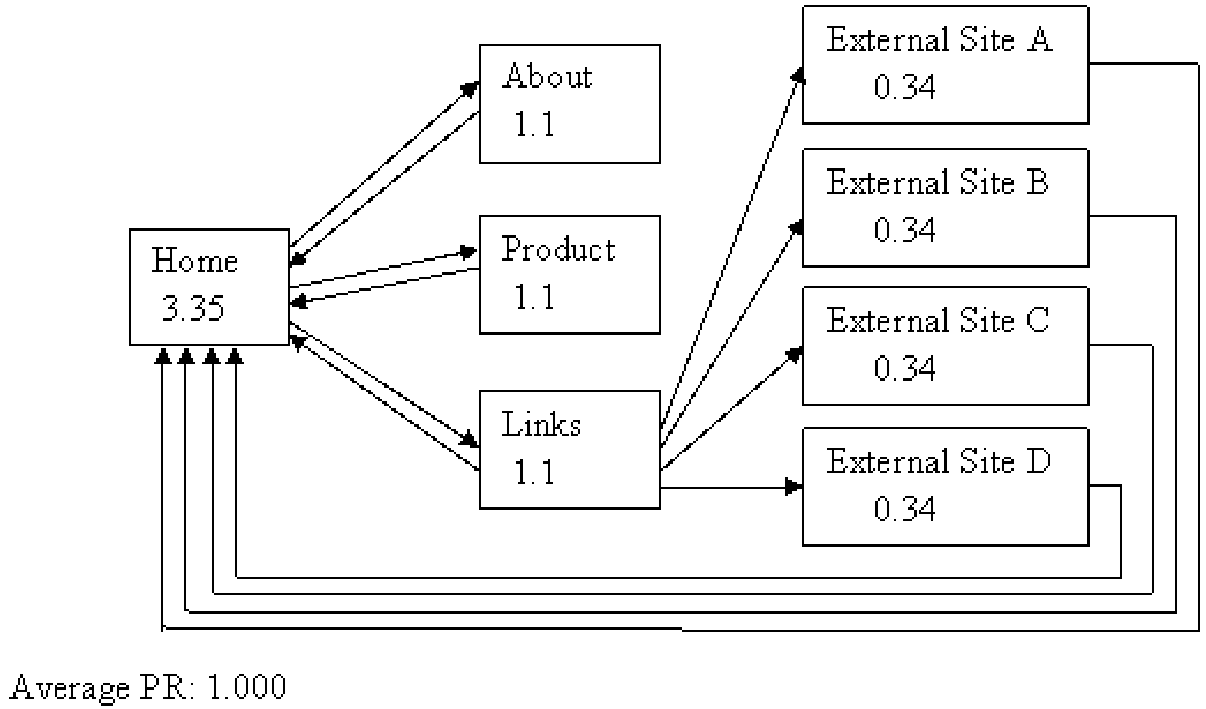
- C is very popular
- To be known by C (like A) brings more prestige than to be known by A (like B)

Example 2



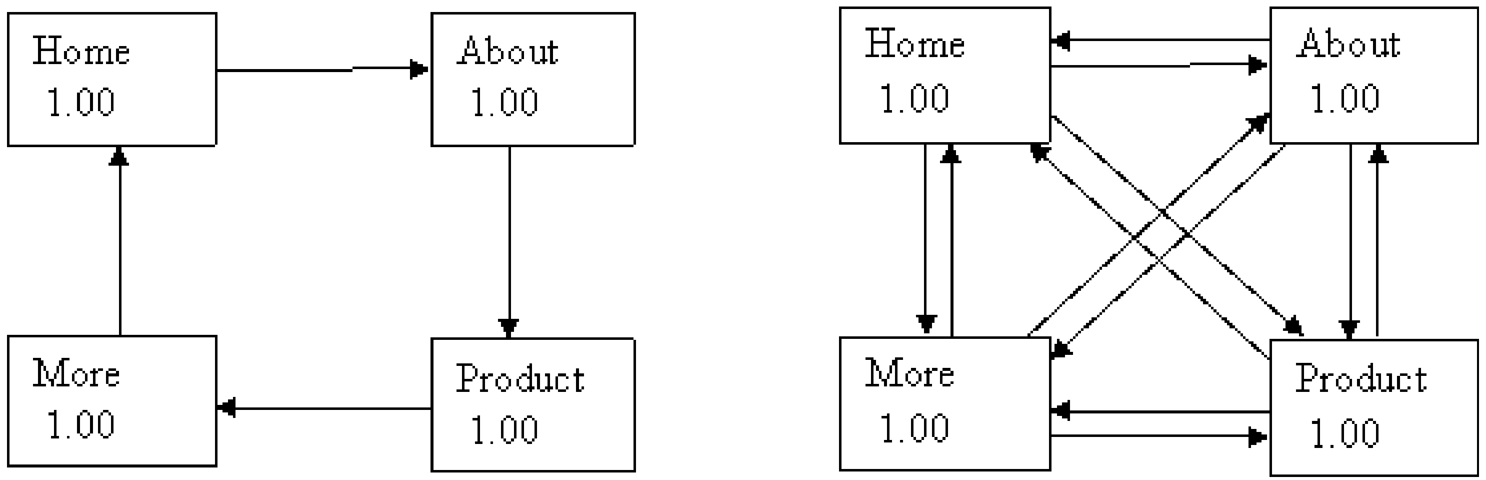
- Average PageRank dropped
- Sinks „consume“ PageRank mass

Example 3



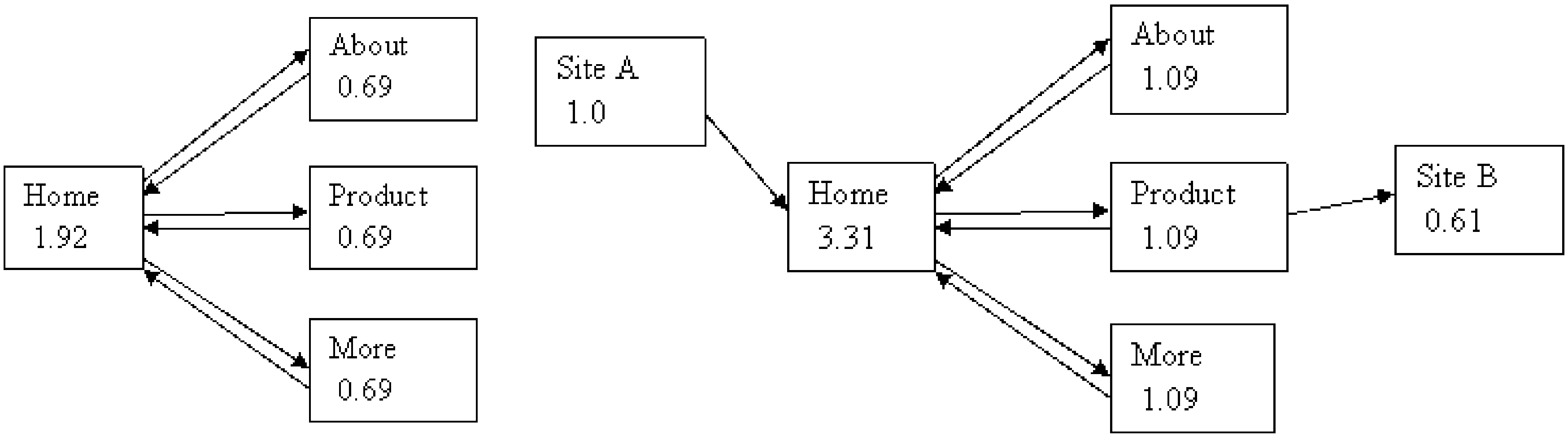
- Repair: Every node reachable from every node
- Average PageRank again at 1

Example 4



- Symmetric link-relationships bear identical ranks

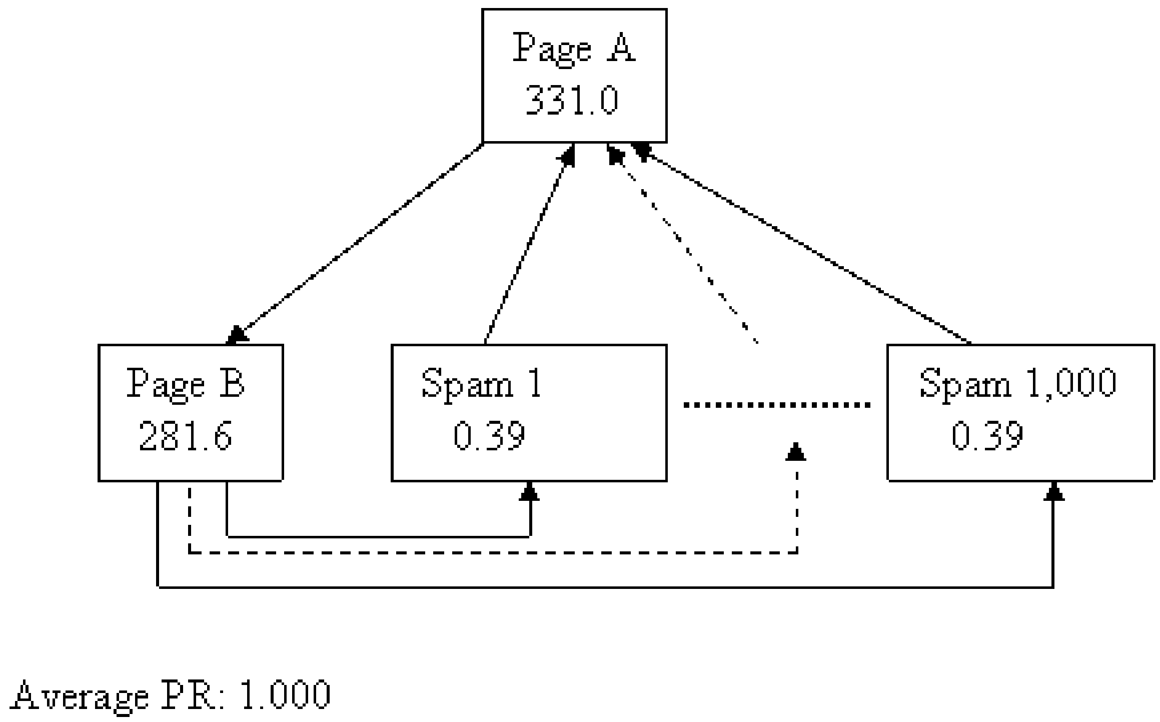
Example 5



Home page outperforms children

External links add strong weights

Example 6



Link spamming increases weights (A, B)

Content of this Lecture

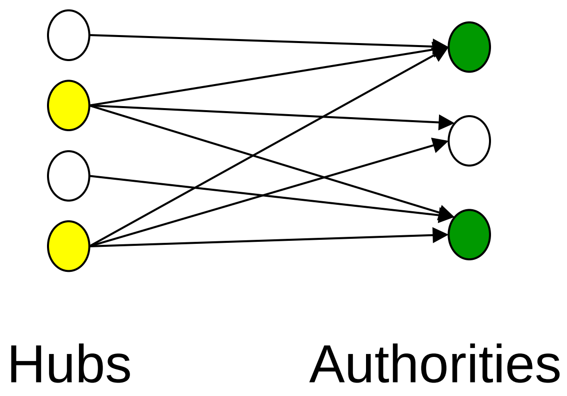
- Searching the Web
- Search engines on the Web
- Exploiting the web structure
  - Prestige in networks
  - Page Rank
  - HITS
- A different flavor: WebSQL

HITS: Hyperlink Induced Topic Search

- Two main ideas
  - Classify web pages into authorities and hubs
  - Use a query-dependent subset of the web for ranking
- Approach: Given a query q
  - Compute the root set R: All pages matching (conventional IR)
  - Expand R by all pages which are connected to any page in R with an outgoing or an incoming link
    - Heuristic – could as well be 2,3,... steps
  - Remove from R all links to pages on the same host
    - Tries to prevent "nepotistic" and purely navigational links
    - At the end, we rank sites rather than pages
  - Assign to each page an authority score and a hub score
  - Rank pages using a weighted combination of both scores

Hubs and Authorities

- Authorities
  - Web pages that contain high-quality, definite information
  - Many other web pages link to authorities
  - “Break-through articles”
- Hubs
  - Pages that try to cover an entire domain
  - Hubs link to many other pages
  - “Survey articles”



- Assumption: hubs preferentially link to authorities (to cover the new stuff), and authorities preferentially link to hubs (to explain the old stuff)

But ...

- Surveys are the most cited papers
- Most hubs are also authorities
- Search engines today don’t use this model

Computation

- A slightly more complicated model
  - Let a be the vector of authority scores of all pages
  - Let h be the vector of hub scores of all pages
- Define

$$a = E^T * h$$

$$h = E * a$$

- Solution can be computed in a similar iterative process as for PageRank

Pros and Cons

- Contra
  - Distinguishing hubs from authorities is somewhat arbitrary and not necessarily a good model for the Web (today)
  - How should we weight the scores?
  - HITS scores cannot be pre-computed; set R and status of pages changes from query to query
- Pro
  - The HITS score embodies IR match scores and links, while PageRank requires a separate IR module and has no rational way to combine the scores

Content of this Lecture

- Searching the Web
- Search engines on the Web
- Exploiting the web structure
- A different flavor of Web search: WebSQL

Side Note: Web Query Languages

- Deficits of search engines
  - No way of specifying structural properties of results
    - “All web pages linking to X (my homepage)”
    - “All web pages reachable from X in at most k steps”
  - No way of extracting specific parts of a web page
    - No “SELECT title FROM webpage WHERE ...”
- Idea: Structured queries over the web
  - Model the web as two relations: node (page) and edge (link)
  - Allow SQL-like queries on these relations
  - Evaluation is done “on the web”
  - Various research prototypes: WebLog, WebSQL, Araneus, W3QL, ...

WebSQL

- Mendelzon, A. O., Mihaila, G. A., & Milo, T. (1997). Querying the World Wide Web. *Journal on Digital Libraries, 1*, 54-67.
- Simple model: The web in 2 relations
  - page(url, title, text, type, length, modification_date, ...)
  - anchor(url, href, anchortext)
  - Could be combined with DOM (XPath) for fine-grained access
- Operations
  - Projection: Post-processing of search results
  - Selections: Pushed to search engine where possible
  - Following links: Call a crawler (or look-up a local crawl)

Example

- Find all web pages which contain the word „JAVA“ and have an outgoing link in whose anchor text the word „applet“ appears; report the target and the anchor text

```
SELECT y.anchortext, y.href
FROM Document x
         SUCH THAT x MENTIONS ‚JAVA‘,
     Anchor y
         SUCH THAT y.url = x.url
WHERE y.anchortext CONTAINS ‚applet‘;
```

Can be evaluated using a search engine

Local processing of pages

More Examples

```
SELECT d.url, d.title
FROM Document d
    SUCH THAT $HOME →|→→ d
WHERE
  d.title CONTAINS ‚Database‘;
```

Report url and title of pages containing “Database” in the title and that are reachable from $HOME in one or two steps

```
SELECT d.title
FROM Document d
    SUCH THAT $HOME (→)*(⇒)* d;
```

Find the titles of all web pages that are reachable (by first local, than non-local links) from $HOME (calls a crawler)

Self Assessment

- How does a Web Crawler work? What are important bottlenecks?
- Name some properties of the IR problem in the web
- What is the complexity of PageRank?
- For which matrices does the Power Iteration method converge to the Eigenvector for Eigenwert 1? Explain each property
- What is the difference between HITS and PageRank? Waht are other models of „importance“ in graphs?
- Could WebSQL be computed on a local copy of the web? What subsystems would be necessary?