• Fixed document collections → World Wide Web: What are the differences?

• Linkage-based algorithms
  – PageRank (Brin and Page, 1998)
  – HITS (Kleinberg, 1998)
Differences closed-world/web: data on web is...

- **Large-volume**
  - Estimates of 40 billion pages for 2005 (800 TB)
    (1TB = 1024 GB = $2^{43}$B)
  - Google indexed 8 billion pages in 2004; coverage 15-20% of web
  - Size of the web is doubling every half a year (Lawrence and Giles, “Searching the world wide web”, Science, 1998)

- **Redundant** (copied or dynamically generated)

- **Unstructured/differently structured documents**

- **Heterogenous** (length, quality, language, contents)

- **Volatile/dynamic**
  - 1 M new pages per day; average page changes every 2-3 weeks
  - 2-9% of indexed pages are invalid

- **Hyperlinked**
Differences closed-world/web: search algorithms

- Different syntactic features in query languages
  - Ranked with proximity, phrase units, order relevant, with or without stemming
- Different indexing (“web-crawling”)
  - Heuristic enterprise; not all pages are indexed (est. 15-20% (2005); 28-55% (1999) of web covered)
- Different heuristics used (in addition to standard IR measures)
  - Proximity and location of search terms (Google)
  - Length of URL (AltaVista)
  - Anchor text pointing to a page (Google)
  - Quality estimates based on link structure
Web Crawling

- At search time, browsers do not access full text
- Index is built off-line; crawlers/spiders find web pages
  - Start with popular URLs and recursively follow links
  - Send new/updated pages to server for indexing
  - Search strategy: breadth-first, depth-first, backlink count, estimated popularity
- Parallel crawling
  - Avoid visiting the same page more than once
  - Partition the web and explore each partition exhaustively
- Agreement robots.txt: directories off-limits for crawlers
- In 1998, Google processed 4 M pages/day (50 pages, 500 links per second); fastest crawlers today: 10 M pages/day
- In 1998, AltaVista used 20 processors with 130G RAM and 500 GB disk each for indexing.
Link structure as a quality measure

- Links contain valuable information: latent human judgement
- Idea: derive quality measure by counting links
- Cf. citation index in science: papers which are cited more are considered to be of higher quality
- Similarity to scientific citation network
  - Receiving a “backlink” is like being cited (practical caveat: on the web, there is no certainty about the number of backlinks)
Simple backlink counting

**Suggestion:** of all pages containing the search string, return the pages with the most backlinks

- Generalisation problem
  - Many pages are not sufficiently self-descriptive
  - Example: the term “car manufacturer” does not occur anywhere on Honda homepage
  - No endogenous information (i.e. information found in the page itself, rather than elsewhere) will help

- Page quality not considered at all, only raw backlink number
  - Overall popular page (Yahoo, Amazon) would be wrongly considered an expert on every string it contains
  - A page pointed to by an important page is also important (even if it has only that one single backlink)
  - Possible to manipulate this measure
Additional problem: manipulatibility

- Web links are not quite like scientific citations
  - Large variation in web pages: quality, purpose, number of links, length (scientific articles are more homogeneous)
    * No publishing/production costs associated with web sites
    * No quality check (cf. peer review in scientific articles)
    * No cost associated with links (cf. length restrictions in scientific articles)
  - Therefore, linking is gratuitous (replicable), whereas citing is not
  - Any quality evaluation strategy which counts replicable features of web pages is prone to manipulation
- Therefore, raw counting will work less well than it does in scientific area
- Must be more clever when using link structure: PageRank, HITS


• Goal: estimate overall relative importance of web pages

• Simulation of a random surfer
  – Given a random page, follows links for a while (randomly), with probability $q$ — assumption: never go back on already traversed links
  – Gets bored after a while and jumps to the next random page, with probability $1 - q$
  – Surfs infinitely long

• PageRank is the number of visits to each page
PageRank formula

\[ R(u) = (1 - q) + q \sum_{v \in B_u} \frac{R(v)}{N_v} \]

- \( u \) a web page
- \( F_u \) set of pages \( u \) points to (“Forward” set)
- \( B_u \) set of pages that point to \( u \)
- \( N_u = |F_u| \) number of pages \( u \) points to
- \( q \) probability of staying locally on page

Simplified PageRank (q=1.0):
The amount of pagerank in the web should be equal to $N$ (so that the average page rank on the web is 1).

Rank must stay constant in each step, but rank sinks lose infinitely much rank.

Rank also gets lost in each step for pages without onward links.

Solution: rank source $\vec{e}$ counteracts rank sinks.

$\vec{e}$ is the vector of the probability of random jumps of random surfer to a random page.
An example: PageRank computation

\[ R(u) = (1 - q) + q \sum_{v \in B_u} \frac{R(v)}{N_v} \]
## Pagerank for the “mini-web” (q=.85)

<table>
<thead>
<tr>
<th>Iteration</th>
<th>PR(X)</th>
<th>PR(Y)</th>
<th>PR(Z)</th>
<th>$\pi(PR)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>3.00000</td>
</tr>
<tr>
<td>2</td>
<td>1.00000</td>
<td>0.575000</td>
<td>1.06375</td>
<td>2.63875</td>
</tr>
<tr>
<td>3</td>
<td>1.05418</td>
<td>0.598029</td>
<td>1.10635</td>
<td>2.75857</td>
</tr>
<tr>
<td>4</td>
<td>1.09040</td>
<td>0.613420</td>
<td>1.13482</td>
<td>2.83865</td>
</tr>
<tr>
<td>5</td>
<td>1.11460</td>
<td>0.623706</td>
<td>1.15385</td>
<td>2.89216</td>
</tr>
<tr>
<td>6</td>
<td>1.13077</td>
<td>0.630581</td>
<td>1.16657</td>
<td>2.92793</td>
</tr>
<tr>
<td>7</td>
<td>1.14158</td>
<td>0.635175</td>
<td>1.17507</td>
<td>2.95183</td>
</tr>
<tr>
<td>8</td>
<td>1.14811</td>
<td>0.638245</td>
<td>1.18075</td>
<td>2.96781</td>
</tr>
<tr>
<td>9</td>
<td>1.15363</td>
<td>0.640292</td>
<td>1.18454</td>
<td>2.97846</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>1.16336</td>
<td>0.64443</td>
<td>1.19219</td>
<td>2.99999</td>
</tr>
<tr>
<td>83</td>
<td>1.16336</td>
<td>0.64443</td>
<td>1.19219</td>
<td>3.00000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration</th>
<th>PR(X)</th>
<th>PR(Y)</th>
<th>PR(Z)</th>
<th>$\pi(PR)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>2</td>
<td>0.15000</td>
<td>0.21375</td>
<td>0.39543</td>
<td>0.75918</td>
</tr>
<tr>
<td>3</td>
<td>0.48612</td>
<td>0.35660</td>
<td>1.50243</td>
<td>1.50243</td>
</tr>
<tr>
<td>4</td>
<td>0.71075</td>
<td>0.45203</td>
<td>0.83633</td>
<td>1.99915</td>
</tr>
<tr>
<td>5</td>
<td>0.86088</td>
<td>0.51587</td>
<td>0.95436</td>
<td>2.33112</td>
</tr>
<tr>
<td>6</td>
<td>0.96121</td>
<td>0.55853</td>
<td>1.03325</td>
<td>2.55298</td>
</tr>
<tr>
<td>7</td>
<td>1.02826</td>
<td>0.58701</td>
<td>1.08597</td>
<td>2.70125</td>
</tr>
<tr>
<td>8</td>
<td>1.07307</td>
<td>0.60605</td>
<td>1.12120</td>
<td>2.80034</td>
</tr>
<tr>
<td>9</td>
<td>1.10302</td>
<td>0.61878</td>
<td>1.14475</td>
<td>2.86656</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>1.16336</td>
<td>0.64443</td>
<td>1.19219</td>
<td>2.99999</td>
</tr>
<tr>
<td>83</td>
<td>1.16336</td>
<td>0.64443</td>
<td>1.19219</td>
<td>3.00000</td>
</tr>
</tbody>
</table>
Matrix notation of PageRank

\[ \vec{r} = c(A\vec{r} + \vec{e}) \]

such that \( c \) is maximised and \( ||\vec{r}||_1 = 1 \) (\( ||\vec{r}||_1 \) is the \( L_1 \) norm of \( \vec{r} \))

\( A \) normalised link matrix of the web:

\[ A_{uv} = \begin{cases} \frac{1}{N_v} & \text{if } \exists u \rightarrow v \\ 0 & \text{otherwise} \end{cases} \]

\( \vec{r} \) PageRank vector (over all web pages), the desired result.

- \( \vec{r} \) is the eigenvector of \((A + \vec{e} \times 1)\) (\( 1 \) is the vector consisting of all ones; outer product!)

- \( \vec{e} \) the uniform vector with \( L_1 \) norm equal to \( 1-q \)

We know from linear algebra that \( r := Ar \); normalise \( (r) \); \( r := Ar \) ... will make \( r \) converge to the dominant eigenvector of \( A \) (independently of \( r \)’s initial value), with eigenvalue \( c \).
Pagerank, matrix algorithm

1. Initialize $\vec{r}$, $\vec{e}$, A

2. Loop:
   
   + $\vec{r} = c(qA + \vec{e} \times 1) \vec{r}$
   + Stop criterion: $||\vec{r}_{i+1}|| - ||\vec{r}_i|| < \epsilon$ (total difference in PageRank between two iterations)
   + This will result in a Page rank vector $\vec{r}$ whose average PageRank is 1: $||\vec{r}_{i+1}||_1 = N$

In our case:

$$A = \begin{bmatrix} 0 & 0 & 1 \\ .5 & 0 & 0 \\ .5 & 1 & 0 \end{bmatrix}$$

$$\vec{e} = \begin{bmatrix} 0.05 \\ 0.05 \\ 0.05 \end{bmatrix}; \vec{r}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$B = (1 - q)A + e \times [111]$$

$$B = \begin{bmatrix} .050 & .050 & .900 \\ .475 & .050 & .050 \\ .475 & .900 & .050 \end{bmatrix}$$

Now iterate $\vec{r}_n = Br_{n-1}$
Iterative matrix-based PageRank computation

\[
B = \begin{pmatrix}
0.050 & 0.050 & 0.900 \\
0.475 & 0.050 & 0.050 \\
0.475 & 0.900 & 0.050 \\
\end{pmatrix}
\]

Iterate \( \mathbf{r}_n = B \mathbf{r}_{n-1} \):

\[
\mathbf{r}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}; \quad \mathbf{r}_1 = \begin{bmatrix} 1.0000 \\ 0.5750 \\ 1.4250 \end{bmatrix}; \quad \mathbf{r}_2 = \begin{bmatrix} 1.3613 \\ 0.5750 \\ 1.0637 \end{bmatrix}; \quad \mathbf{r}_3 = \begin{bmatrix} 1.0542 \\ 0.7285 \\ 1.2173 \end{bmatrix}; \quad \mathbf{r}_4 = \begin{bmatrix} 1.1847 \\ 0.5980 \\ 1.2173 \end{bmatrix}; \quad \mathbf{r}_5 = \begin{bmatrix} 1.1847 \\ 0.6535 \\ 1.1618 \end{bmatrix};
\]

\[
\mathbf{r}_6 = \begin{bmatrix} 1.1375 \\ 0.6535 \\ 1.2090 \end{bmatrix}; \quad \mathbf{r}_7 = \begin{bmatrix} 1.1776 \\ 0.6335 \\ 1.1889 \end{bmatrix}; \quad \mathbf{r}_8 = \begin{bmatrix} 1.1606 \\ 0.6505 \\ 1.1889 \end{bmatrix}; \quad \mathbf{r}_9 = \begin{bmatrix} 1.1606 \\ 0.6432 \\ 1.1962 \end{bmatrix}; \quad \mathbf{r}_{10} = \begin{bmatrix} 1.1667 \\ 0.6432 \\ 1.1900 \end{bmatrix}; \ldots
\]
PageRank computation (practicalities)

- **Space**
  - Example: 75 M unique links on 25 M pages
  - Then: memory for PageRank 300MB
  - Link structures is compact (8B/link compressed)

- **Time**
  - Each iteration takes 6 minutes (for the 75 M links)
  - Whole process: 5 hours
  - Convergence after 52 iter. (322M links), 48 iter. (161M links)
  - Scaling factor linear in log $n$

- Pages without children removed during iteration

- Raw data can be obtained during web crawl; cost of computing PageRank is insignificant compared to the cost of building a full index
PageRank versus usage data

- Difference between linking behaviour (public) and actual usage data (web page access numbers from NLANR)
  - PageRank uses only public information; thus fewer privacy implications than usage data (pages that are accessed but not linked to)
  - PageRank produces a finer resolution compared to small usage sample
  - But: not all web users create links

- Propagation simulates word-of-mouth effects in complex network (ahead of time):
  - PageRank can change fast (one link on Yahoo)
    * Good pages often have only a few important backlinks (at first)
    * Those pages would not be found by simply back-link counting
  - Net traffic can change fast (one mention on the radio)
Model of collaborative trust; users want information from “trusted” sources

PageRank is immune to manipulation: it must convince an important site, or many unimportant ones, to point to it

- Spamming PageRank costs real money – a good property for a search algorithm
- Google’s business model: never sell PageRank (only advertising space)

PageRank is a good predictor of optimal crawling order
Top 15 PageRanks in July 1996

<table>
<thead>
<tr>
<th>Page</th>
<th>PageRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download Netscape Software</td>
<td>11589.00</td>
</tr>
<tr>
<td><a href="http://www.w3.org">http://www.w3.org</a></td>
<td>10717.70</td>
</tr>
<tr>
<td>Welcome to Netscape</td>
<td>8673.51</td>
</tr>
<tr>
<td>Point: It’s what you’re searching for</td>
<td>7930.92</td>
</tr>
<tr>
<td>Web-Counter home page</td>
<td>7254.97</td>
</tr>
<tr>
<td>The Blue Ribbon Campaign for Online Free Speech</td>
<td>7010.39</td>
</tr>
<tr>
<td>CERN Welcome</td>
<td>6562.49</td>
</tr>
<tr>
<td>Yahoo!</td>
<td>6561.80</td>
</tr>
<tr>
<td>Welcome to Netscape</td>
<td>6203.47</td>
</tr>
<tr>
<td>Wusage 4.1: A Usage Statistics System for Web Servers</td>
<td>5963.27</td>
</tr>
<tr>
<td>The World Wide Web consortium (W3C)</td>
<td>5672.21</td>
</tr>
<tr>
<td>Lycos, Inc. Home Page</td>
<td>4683.31</td>
</tr>
<tr>
<td>Starting Point</td>
<td>4501.98</td>
</tr>
<tr>
<td>Welcome to Magellan!</td>
<td>3866.62</td>
</tr>
<tr>
<td>Oracle Corporation</td>
<td>3587.63</td>
</tr>
</tbody>
</table>

Benefits for search with PageRank are greatest for underspecified queries
Hypertext Induced Topic Search (HITS)

- Goal: find authorities on a certain topic (relevance, popularity)
- Idea: There are hubs and authorities on the web, which exhibit a mutually reinforcing relationship
  - Hubs: Recommendation pages with links to high-quality pages (authorities), e.g. compilations of favourite bookmarks, “useful links”
  - Authorities: Pages that are recognised by others (particularly by hubs!) as experts on a certain topic
- Authorities are different from universally popular pages (high backlink count), which are not particular experts on that topic
HITS

- Each page has two non-negative weights: an authority weight $a$ and a hub weight $h$
- At each iteration, update the weights:
  - If a page points at many good authorities, it is probably a good hub:
  \[
  h_p = \sum_{q: <p, q> \in A} a_q
  \]
  - If a page is pointed to by many good hubs, it is probably a good authority:
  \[
  a_p = \sum_{q: <q, p> \in A} h_q
  \]
- Normalise weights after each iteration
• Start with the root set: set of web pages containing the query terms
• Create the base set: root set plus all pages pointing to the root set (cut-off if too many), and being pointed to by the root set
• The base set typically contains 1000-5000 documents
HITS: Algorithm

Given:

- a set \( D = \{D_1 \ldots D_n\} \) of documents (base set)
- \( A \), the linking matrix: edge \( <i, j> \in A \) iff \( D_i \) points to \( D_j \)
- \( k \), the number of desired iterations

Initialise: \( \vec{a} = \{1, 1, \ldots, 1\}; \vec{h} = \{1, 1, \ldots, 1\} \)

Iterate: for \( c = 1 \ldots k \)

- for \( i = 1 \ldots n \): \( a_p = \sum_{q: <q, p> \in A} h_q \)
- for \( i = 1 \ldots n \): \( h_p = \sum_{q: <p, q> \in A} a_q \)

Normalise \( \vec{a} \) and \( \vec{h} \): \( \sum_{i \in D_i} a_i = \sum_{i \in D_i} h_i = 1 \)
HITS: Convergence

- Updates:
  \[ \vec{a} = A^T \vec{h} \quad \text{and} \quad \vec{h} = A \vec{a} \]

- After the first iteration:
  \[ \vec{a}_1 = A^T A \vec{a}_0 = (A^T A) \vec{a}_0 \quad \text{and} \quad \vec{h}_1 = A A^T \vec{h}_0 = (A A^T) \vec{h}_0 \]

- After the second iteration:
  \[ \vec{a}_2 = (A^T A)^2 \vec{a}_0 \quad \text{and} \quad \vec{h}_2 = (A A^T)^2 \vec{h}_0 \]

- Convergence to
  - \( \vec{a} \leftarrow \) dominant eigenvector\((A^T A)\)
  - \( \vec{h} \leftarrow \) dominant eigenvector\((A A^T)\)
Authorities on “java”

0.328 http://www.gamelan.com Gamelan
0.251 http://java.sun.com JavaSoft home page
0.190 http://www.digitalfocus.com/digital The Java Developer: How do I

Authorities on “censorship”

0.376 http://www.eff.org EFF – The Electronic Frontier Foundation
0.344 http://www.eff.org/blueribbon.html The Blue Ribbon Campaign for Online Free Speech
0.238 http://www.cdt.org The Center for Democracy and Technology
0.235 http://www.vtw.org Voters Telecommunication Watch
0.218 http://www.aclu.org ACLU: American Civil Liberties Union

Authorities on “search engine”

0.346 http://www.yahoo.com Yahoo
0.291 http://www.excite.com Excite
0.239 http://www.mckinley.com Welcome to Magellan
0.231 http://www.lycos.com Lycos Home Page
0.231 http://www.altavista.digital.com AltaVista: Main Page
Both HITS and PageRank infer quality/“expert-ness” from link structure of the web

- Link structure contains latent human judgement
- Use different models of type of web pages
- Iterative algorithms
- Use of these weights for search (in different ways)
- Other differences between closed-world assumption (IR) and world wide web: data, indexing, query constructs, search heuristics