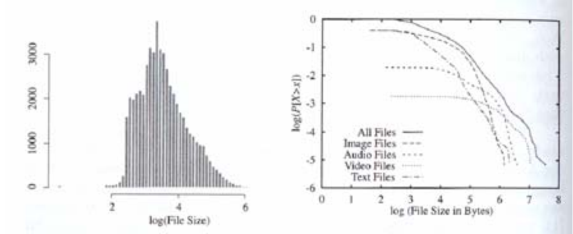
Searching the Web

Challenges of Web Searching

- The Web is huge
  - Google indexes over 4 billion pages
  - Some estimate search engines only reach 16% of web
- The Web is dynamic
  - About 23% of pages change daily
  - In .com domain, pages average 10 day half-life (half the pages gone in 10 days)
- The Web is open
  - Anyone can post pages on any topic at any time

Modeling the Web



- Heaps’ and Zipf’s laws are valid in the Web too, but:
  - Vocabulary grows faster (β is larger);
  - Words distribution is more biased (θ value is larger);
- Distribution of document sizes:
  - Probability of finding a document of size *x* bytes:

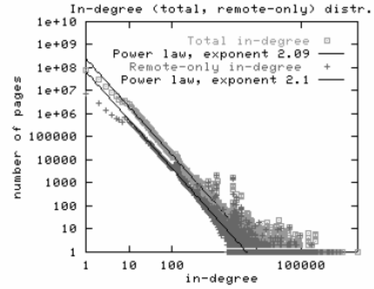
$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp - (\ln x - \mu)^2 / 2\sigma^2$$

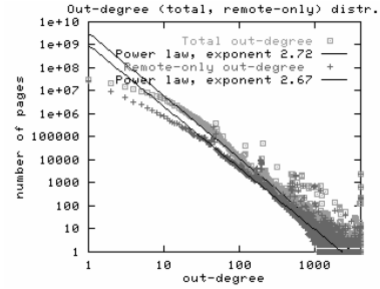
    - μ - average (9.357) σ - standard deviation (1.318)
  - Pareto distribution:

$$p(x) = \frac{\alpha k^{\alpha}}{x^{1+\alpha}}$$

    - α and *k* are parameters of the distribution: α=1.1 and k=9.3 Kb for the Web

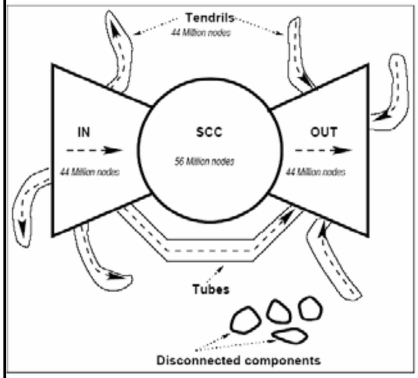
Modeling the Web (Cont.)





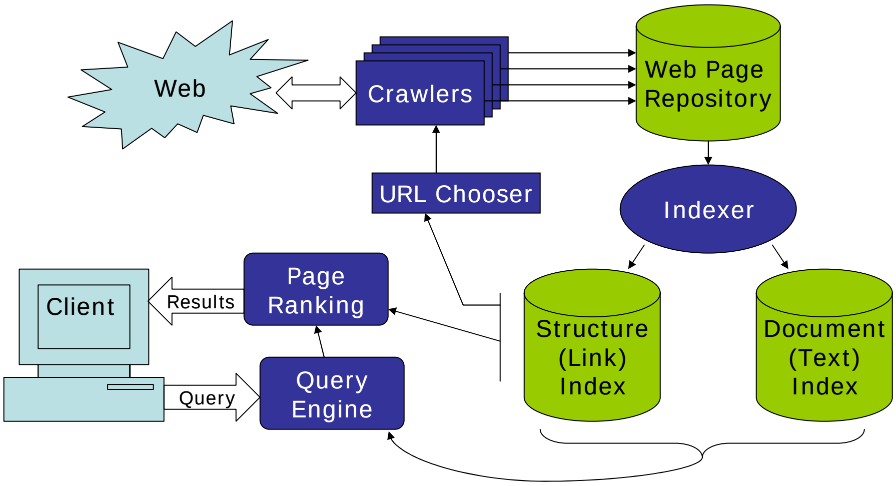
- In-degree of page: the number of pages pointing to this page
- Out-degree of page: the number of hyperlinks contained in this page
- The fraction of web pages with in-degree/out-degree i is proportional to $1/i^x$ for some x > 1:
  - x=2.1 for in-degree distribution
  - x=2.72 for out-degree distribution
- The average out-degree is about 7.2.

Modeling the Web (Cont.)



- Strongly-connected component: set of pages such that for all pairs of pages (u,v) in the set, there exists a directed path from u to v:
  - a surfer can follow hyperlinks to surf from u to v
- The web as a bowtie:
  - SCC is a giant strongly connected component.
  - IN consists of pages with paths to SCC, but no path from SCC:
    - new pages that link to interesting destinations on the web, but which have not yet been discovered by the core of the web and are therefore not reachable from the SCC.
  - OUT consists of pages with paths from SCC, but no path to SCC:
    - pages of corporate internets which are well-known, but whose links point only internally
  - TENDRILS consists of pages that cannot surf to SCC, and which cannot be reached by surfing from SCC:
    - the web has not yet discovered these pages, and these pages do not contain interesting links back to better-known regions of the web

Search Engine Architecture



Web

Crawlers

Web Page Repository

URL Chooser

Indexer

Client

Results

Page Ranking

Query

Query Engine

Structure (Link) Index

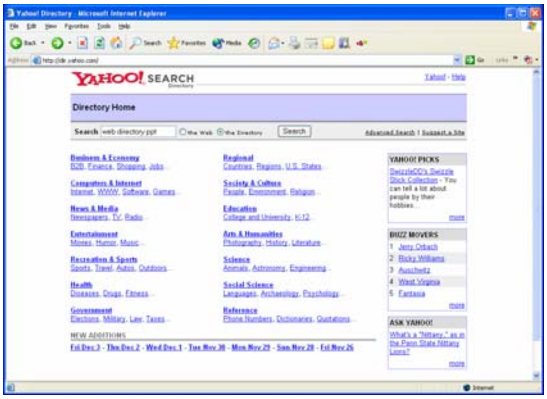
Document (Text) Index

Search Engine Components

- **Crawler** – downloads pages from the web
  - **URL Chooser** – decides which page to download next
- **Web Page Repository** – stores full HTML of downloaded pages
- **Indexer** – extracts words and links from pages, associating them with URLs
- **Document (Text) Index** – stores mapping from keywords to URLs
- **Link (Structure) Index** – stores a representation of the directed graph formed from links between pages
- **Query Engine** – receives keyword search terms from user to find matches in Document Index
- **Page Ranking** – uses hyperlink analysis to rank the pages that match a particular query

Browsing

- Web coverage provided by directories is very low; (less than 1% of all pages)
- Yahoo! – the largest directory:
  - ~1 mil.pages classified;
  - Pages are submitted, reviewed and added;
  - Advantage:
    - if found, the answer will be useful in most cases;
  - Disadvantage:
    - classification is not specialized enough;
    - not all Web pages are classified;



Yahoo! Directory - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://dir.yahoo.com/
Go Links
YAHOO! SEARCH Directory
Yahoo! - Help
Internet

Directory Home

Search web directory ppt ○ the Web ● the Directory Search Advanced Search | Suggest a Site

Business & Economy B2B, Finance, Shopping, Jobs...

Computers & Internet Internet, WWW, Software, Games...

News & Media Newspapers, TV, Radio...

Entertainment Movies, Humor, Music...

Recreation & Sports Sports, Travel, Autos, Outdoors...

Health Diseases, Drugs, Fitness...

Government Elections, Military, Law, Taxes...

Regional Countries, Regions, U.S. States...

Society & Culture People, Environment, Religion...

Education College and University, K-12...

Arts & Humanities Photography, History, Literature...

Science Animals, Astronomy, Engineering...

Social Science Languages, Archaeology, Psychology...

Reference Phone Numbers, Dictionaries, Quotations...

NEW ADDITIONS

Fri Dec 3 - Thu Dec 2 - Wed Dec 1 - Tue Nov 30 - Mon Nov 29 - Sun Nov 28 - Fri Nov 26

YAHOO! PICKS

SwizzleOD's Swizzle Stick Collection - You can tell a lot about people by their hobbies...

more

BUZZ MOVERS

1 Jerry Orbach
2 Ricky Williams
3 Auschwitz
4 West Virginia
5 Fantasia

more

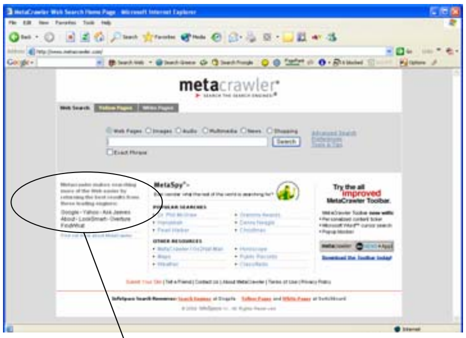
ASK YAHOO!

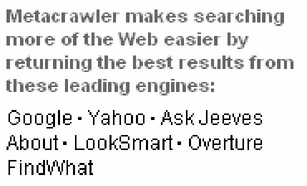
What's a "Nittany," as in the Penn State Nittany Lions?

more

Metasearch

- Web server that sends query to
  - Several search engines
  - Web directories
  - Databases
- Example: Metacrawler
- Collect results
- Unify them (Data fusion)
- Aim: better coverage
- Issues
  - Translation of query
  - Uniform result (fusion rankings, e.g. pages retrieved by several engines)

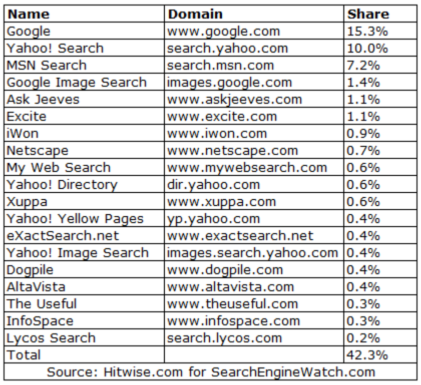




Metacrawler makes searching more of the Web easier by returning the best results from these leading engines:

Google • Yahoo • Ask Jeeves
About • LookSmart • Overture
FindWhat

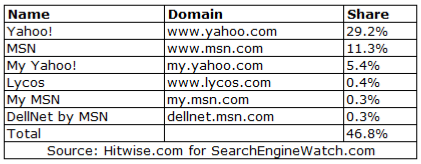
Popularity of Search engines and Portal Sites (April 2004)



Name	Domain	Share
Google	www.google.com	15.3%
Yahoo! Search	search.yahoo.com	10.0%
MSN Search	search.msn.com	7.2%
Google Image Search	images.google.com	1.4%
Ask Jeeves	www.askjeeves.com	1.1%
Excite	www.excite.com	1.1%
iWon	www.iwon.com	0.9%
Netscape	www.netscape.com	0.7%
My Web Search	www.mywebsearch.com	0.6%
Yahoo! Directory	dir.yahoo.com	0.6%
Xuppa	www.xuppa.com	0.6%
Yahoo! Yellow Pages	yp.yahoo.com	0.4%
eXactSearch.net	www.exactsearch.net	0.4%
Yahoo! Image Search	images.search.yahoo.com	0.4%
Dogpile	www.dogpile.com	0.4%
AltaVista	www.altavista.com	0.4%
The Useful	www.theuseful.com	0.3%
InfoSpace	www.infospace.com	0.3%
Lycos Search	search.lycos.com	0.2%
Total		42.3%

Source: Hitwise.com for SearchEngineWatch.com

Search engines



Name	Domain	Share
Yahoo!	www.yahoo.com	29.2%
MSN	www.msn.com	11.3%
My Yahoo!	my.yahoo.com	5.4%
Lycos	www.lycos.com	0.4%
My MSN	my.msn.com	0.3%
DellNet by MSN	dellnet.msn.com	0.3%
Total		46.8%

Source: Hitwise.com for SearchEngineWatch.com

Portal sites

- In addition to searching, portal sites usually offer web directories, web mail, news etc.

Information Retrieval Applied to the Web

- Web pages contain additional information besides just plain text
- Use HTML formatting tags to infer importance of terms in page, and weight accordingly
  - Headings and bold terms are more important than paragraph text
  - Include link text in page that is *linked to*
    - Links often constitute better description of the page than the page itself does
- Construct inverted index
  - Set of inverted lists, one for each word
  - Maps word to a sorted list of locations (page IDs and position of word in page)

Classic Information Retrieval Assumptions

- Set of documents well-managed
  - Size relatively fixed (and not that large)
  - Most documents are high quality, not much junk
  - Documents are relatively self-describing

Problems using IR for Web Search

- The Abundance Problem
  - The web is too big, number of pages that can reasonably be returned as relevant is far too much for a human to process
  - The web contains a lot of junk, anyone can post any page
- Many pages aren't sufficiently self-descriptive
  - Pages may contain little text, mostly images or other content
  - Might not contain text that matches their description, e.g. "search engine"
- Much benefit from manipulating results
  - Alter pages so that relevant terms appear more frequently (spamming)

Solution: Hyperlink Analysis

- Need a way to rank query results by document quality
- Hyperlink assumptions:
  - Link from page A to B is a recommendation of page B by author of page A
  - If two pages are connected by a link, they might be on the same topic
  - anchor text of a link describes its target
- Types of hyperlink ranking
  - Query-independent, each page gets a quality or importance score based on the links to it
    - PageRank
  - Query-dependent, importance score assigned to pages only in the context of queries
    - HITS

Page Rank

- Designed by Brin and Page at Stanford University
- Used to implement Google
- Main idea:
  - a page has a high PageRank if the sum of the PageRanks of its in-links is high
  - a high PageRank page has many in-links or few highly PageRanked in-links
- Retrieval:
  - standard IR score (cosine product using content - term weights, etc)
  - combined with PageRank value

Algebraic Background Eigenvalues & Eigenvectors

- **Eigenvectors** (for a square *m*×*m* matrix **S**)

$$\mathbf{S}\mathbf{v} = \lambda\mathbf{v}$$

(right) eigenvector

$\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0}$

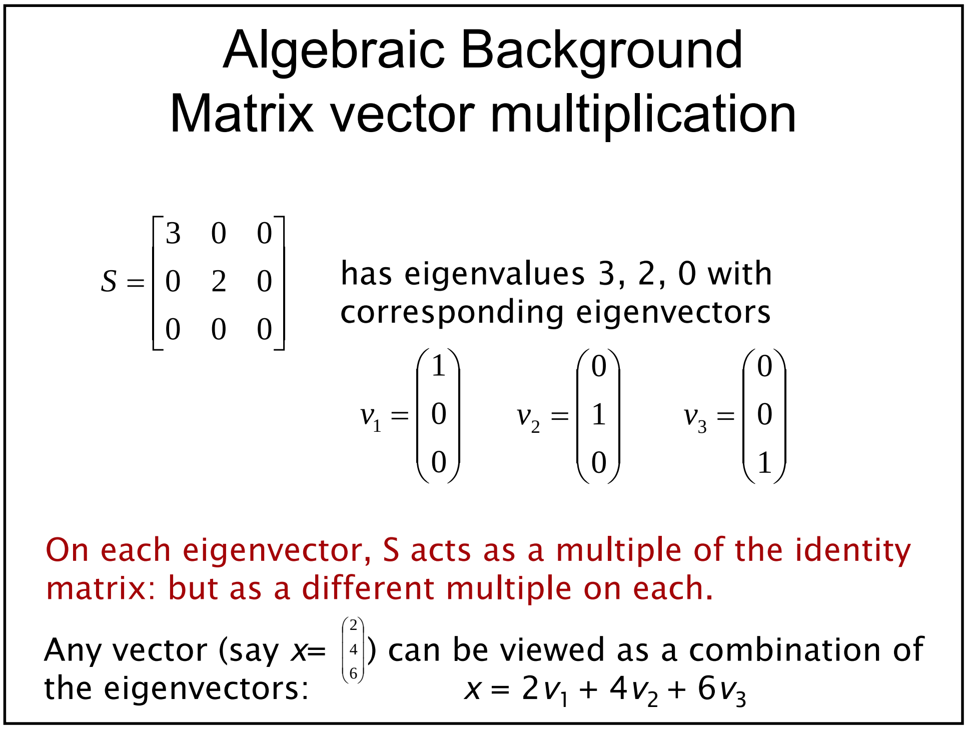
eigenvalue

$\lambda \in \mathbb{R}$

Example

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2\begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- How many eigenvalues are there at most?



Algebraic Background
Matrix vector multiplication

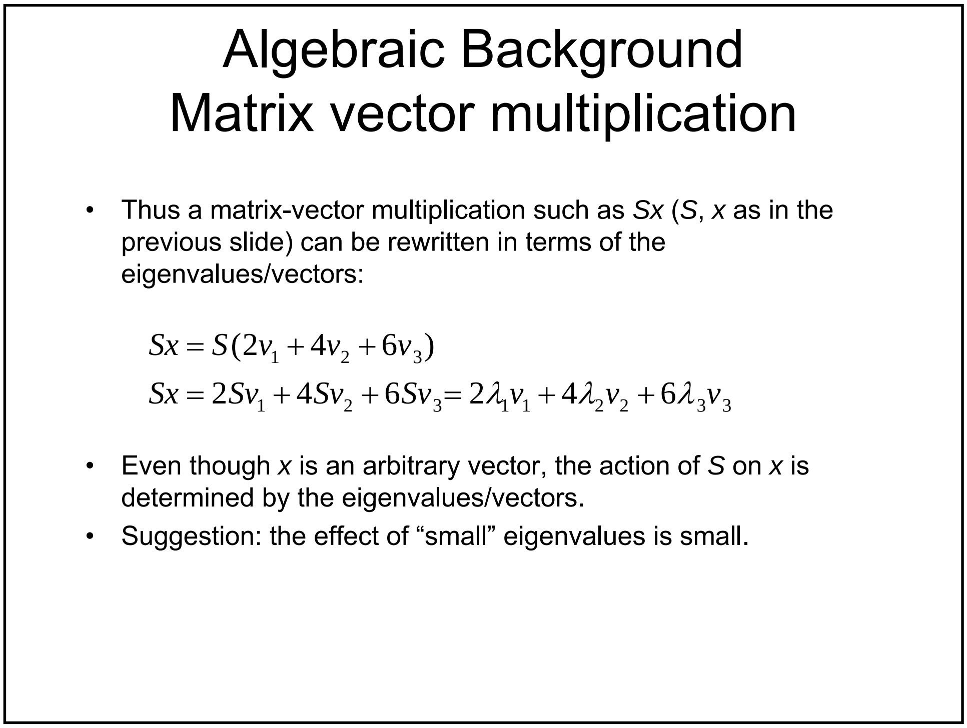
$$S = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

has eigenvalues 3, 2, 0 with corresponding eigenvectors

$$v_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad v_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

On each eigenvector, S acts as a multiple of the identity matrix: but as a different multiple on each.

Any vector (say $x= \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$) can be viewed as a combination of the eigenvectors: $x = 2v_1 + 4v_2 + 6v_3$



Algebraic Background
Matrix vector multiplication

- Thus a matrix-vector multiplication such as *Sx* (*S*, *x* as in the previous slide) can be rewritten in terms of the eigenvalues/vectors:

$$Sx = S(2v_1 + 4v_2 + 6v_3)$$
$$Sx = 2Sv_1 + 4Sv_2 + 6Sv_3 = 2\lambda_1 v_1 + 4\lambda_2 v_2 + 6\lambda_3 v_3$$

- Even though *x* is an arbitrary vector, the action of *S* on *x* is determined by the eigenvalues/vectors.
- Suggestion: the effect of “small” eigenvalues is small.

Algebraic Background

Consider a square *nxn* matrix **A**.

- Eigenvectors of **A**

  $X_1, . . . ,X_n$

are a special set of (column) vectors associated with **A**

- Each eigenvector is associated with a corresponding eigenvalue:

  $\lambda_1 , . . . , \lambda_n$

- Basic property:

  i=1…n, $AX_i = \lambda_i X_i$

- Here, we assume that A has n distinct eigenvalues
- Also assume that $|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n|$
- Eigenvector X_1 corresponds to the largest eigenvalue λ_1: principal eigenvector
- Eigenvectors X_1, X_2, …, X_n are linearly independent and form a basis of R^n.

Thus any vector q_0 can be presented as

$$\mathbf{q_0 = b_1X_1+b_2X_2+…+b_nX_n}$$

Assuming **$b_1 \neq 0$**, vector **q_k=A^k q_0** is equal to:

$$q_k = b_1\lambda_1^k\left(X_1 + \sum_{j=2}^{n}\frac{b_j}{b_1}\left(\frac{\lambda_j}{\lambda_1}\right)^k X_j\right)$$

Algebraic Background

So, q_k converges to the following limit

$$\lim_{k\to\infty} q_k = \lambda_1^k b_1 X_1 \Rightarrow \lim_{k\to\infty}\frac{q_k}{\lambda_1^k} = b_1 X$$

$\mathrm{O}\left(\frac{\lambda_2}{\lambda_1}\right)$ the rate of convergence

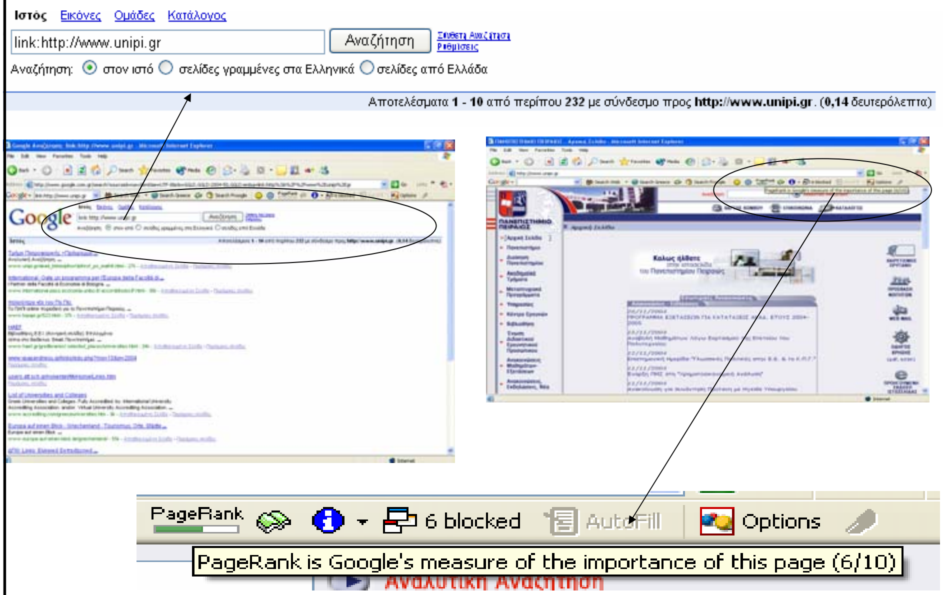
Page rank (cont.)

- Random Surfer Model : user randomly navigates
  - Initially the surfer is at a random page
  - At each step the surfer proceeds to
    - a randomly chosen Web page with probability *d* ("damping factor") e.g. probability of random jump = 0.15
    - to a randomly chosen page linked to the current page with probability 1-*d* e.g. probability of following a random outlink = 0.85
    - User cannot use the back button of the web browser
- **PageRank** PR of a page **p** = probability that the surfer is at page **p** on a given time

$$PR(p) = \frac{d}{N} + (1-d) \sum_{q:q \to p} \frac{PR(q)}{N_q}$$

- *where d* set by system, page q points to page p, N_q the number of out-links of q, N the number of the web pages
- Recursive formula: start with any set of values and iterate until convergence
- Process modelled by Markov Chain

Page rank (cont.)



Ιστός Εικόνες Ομάδες Κατάλογος

link:http://www.unipi.gr

Αναζήτηση

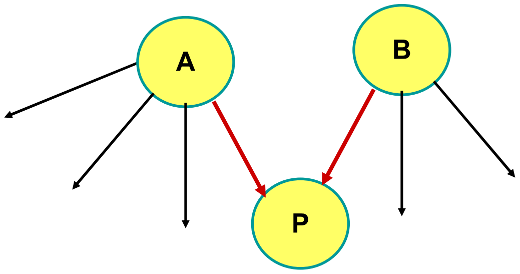
Σύνθετη Αναζήτηση
Ρυθμίσεις

Αναζήτηση: ● στον ιστό ○ σελίδες γραμμένες στα Ελληνικά ○ σελίδες από Ελλάδα

Αποτελέσματα **1 - 10** από περίπου **232** με σύνδεσμο προς **http://www.unipi.gr**. (**0,14** δευτερόλεπτα)

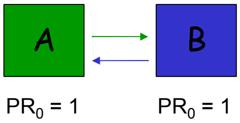
PageRank is Google's measure of the importance of this page (6/10)

An example of Page rank computation



- PageRank of P is (1-d)∗ (1/4th the PageRank of A + 1/3rd the PageRank of B)+d/N
- In HITS, a webpage with a large number of out-going links will have influence on the final ranking.
- In PageRank, each out-going hyperlink from q is weighted by $1/N_q$, thus every webpage has the same total out-going weights:
  - Internet Democracy: each webpage has a total of one vote

Examples



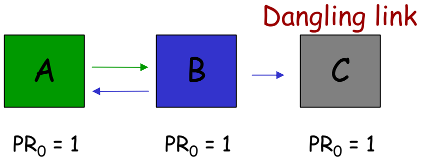
d=0.15

$PR_1(A)= 0.15/2+0.85(PR_0(B)/1) = 0,925$

$PR_1(B)= 0.15/2+0.85(PR_0(A)/1) = 0,925$

lim $PR_k(A)= 0.5$

lim $PR_k(B)= 0.5$



d=0.15

$PR_1(A)=0.15/3+0.85(PR_0(B) /2) = 0.475$

$PR_1(B)=0.15/3+0.85(PR_0(A)/1) = 0,9$

$PR_1(C)=0.15/3+0.85(PR_0(B)/2) = 0.475$

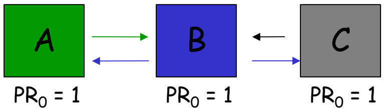
lim $PR_k(A)= 0.1115$

lim $PR_k(B)= 0.1448$

lim $PR_k(C)= 0.1115$

d=0

PR_1 (A)=1/2
PR_1 (B)= 1
PR_1 (C)=1/2
lim $PR_k(A)= 0$
lim $PR_k(B)= 0$
lim $PR_k(C)= 0$



d=0.15

$PR_1(A)=0.15/3+0.85(PR_0(B) /2) = 0.4750$

$PR_1(B)=0.15/3+0.85(PR_0(A) +PR_0(C)) = 1.75$

$PR_1(C)=0.15/3+0.85(PR_0(B)/2) = 0.4750$

lim PR_k (A)= 0.2568
lim PR_k (B)= 0.4865
lim PR_k (C)= 0.2568

d=0

$PR_1(A)=0.5$

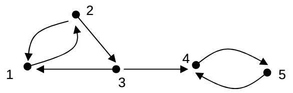
$PR_1(B)=2$

$PR_1(C)=0.5$

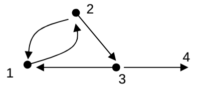
lim PR_k (A)= 0.5 or 1
lim PR_k (B)= 2 or 1
lim PR_k (C)= 0.5 or 1

Problems with Basic PageRank (d=0)

- Web is not a strongly connected graph
  - Rank sink – strongly connected component with no outward links

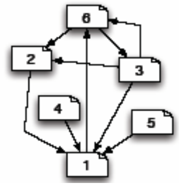


    - Nodes not part of sink get rank of 0
  - Rank leak – single page (node) with no outward links



    - All nodes eventually converge to rank of 0

Algebraic interpretation

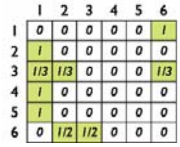


Equivalently, we can use the following normalized page rank update:

$$\overline{PR}_p^k = \frac{d}{N} + (1-d)\sum_{q:q\to p}\frac{PR_q^{k-1}}{N_q} \text{ and after normalization}$$

$$PR_p^k = \frac{\overline{PR}_p^k}{\sum_{i=1}^{N}\overline{PR}_i^k}$$

A=



	1	2	3	4	5	6
1	0	0	0	0	0	1
2	1	0	0	0	0	0
3	1/3	1/3	0	0	0	1/3
4	1	0	0	0	0	0
5	1	0	0	0	0	0
6	0	1/2	1/2	0	0	0

Using matrix notation:

$$\overline{PR}^k = P\cdot PR^{k-1} \text{ and } PR^k = \frac{1}{\sum_{i=1}^{N}\overline{PR}_i^k}\cdot\overline{PR}^k \text{ where } P = \left((1-d)A + d\cdot ev^T\right)^T$$

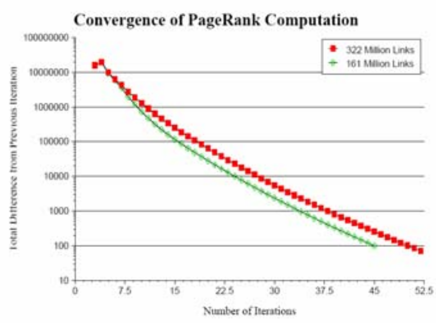
and

$$\overline{\mathbf{PR}}^k = \begin{bmatrix}\overline{PR}_1^k\\ \overline{PR}_2^k\\ \vdots\\ \overline{PR}_N^k\end{bmatrix}, \mathbf{PR}^k = \begin{bmatrix}PR_1^k\\ PR_2^k\\ \vdots\\ PR_N^k\end{bmatrix}, \sum_{i=1}^{n} PR_i^k = 1,\ \mathbf{v} = \begin{bmatrix}v_1\\ v_2\\ \vdots\\ v_N\end{bmatrix}, \sum_{i=1}^{n} v_i = 1,\ \mathbf{e} = \begin{bmatrix}1\\ 1\\ \vdots\\ 1\end{bmatrix}, A_{u,v} = \begin{cases}\frac{1}{N_u} & if\ u\to v\\ 0 & otherwise\end{cases}$$

v_i=1/N for this particular update

Properties of Matrix P

- λ_1=1 is the largest eigenvalue
- $|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n|$
- $|\lambda_2| \leq (1\text{-}d)$
- PR^k converges to the primary eigenvector of the matrix P
- The rate of convergence: $O(|\lambda_2/\lambda_1|)$, i.e. O(1-d)
- Rank values typically converge in 50-100 iterations
- Rank orders converge even faster
- Faster convergence if using random jumps more:
  - higher d value: the true hyperlink structure of the web is used less to determine web importance
- Page Rank estimation:
  - the estimation of the primary eigenvector of a huge matrix NxN where N=4,300,000,000
  - It has been reported that Google computes PageRank once every few weeks for all documents in its Web collection

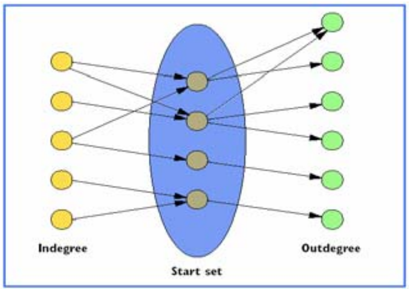


Problems with PageRank

- Pages related to a topic may not contain that topic's keywords
  - "Search engine" example again
- No balance between relevancy and popularity
  - Very popular pages (such as search engines and web portals) may be returned artificially high due to their popularity (even if not very related to the query)
- Despite these problems, seems to work fairly well in practice

Query-Dependent Ranking

- Importance score assigned only in response to user query
  - Can account for both quality and relevance
- Construct query-specific neighborhood graph
  - Start with set of documents matching query (maybe top 200)
  - Add to this its neighborhood: set of documents that link to it and that it links to
  - Can now perform query-dependent ranking on neighborhood



Hyperlink-Induced Topic Search (HITS)

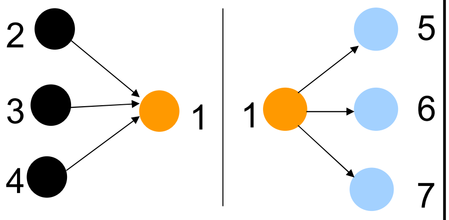
- Creates a small graph of focused hypertext documents e.g. a particular topic of interest
  - Authoritative pages are pages that have received many citations and are considered the best source of information. Citations from important pages should be weighted higher than citations from less-important pages
  - Hub pages are pages that contain links to authoritative pages and is used as a measure of importance. A good hub page should allow us to reach many authoritative pages

HITS Algorithm

- For each node A in a neighborhood N
  - Hub(A) = hub score
  - Auth(A) = authority score
- Initialize Hub(A) to 1 (for all nodes)
- For all nodes A in N, compute

$$Auth(p) = \sum_{q \in N: q \to p} Hub(q)$$

$$Hub(p) = \sum_{q \in N: p \to q} Auth(q)$$



a(1) = h(2) + h(3) + h(4)   h(1) = a(5) + a(6) + a(7)

- Normalize after each iteration and repeat until Hub and Authority vectors converge

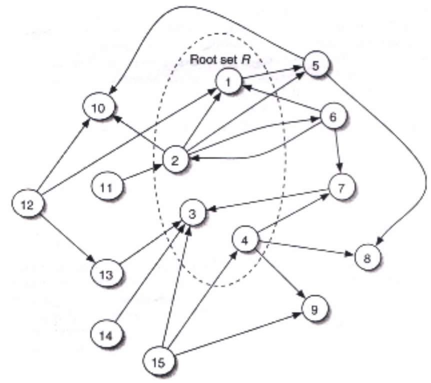
$$Auth(p) = \frac{Auth(p)}{\sqrt{\sum_{q \in N} Auth^2(q)}}, \quad Hub(p) = \frac{Hub(p)}{\sqrt{\sum_{q \in N} Hub^2(q)}}$$

HITS Example

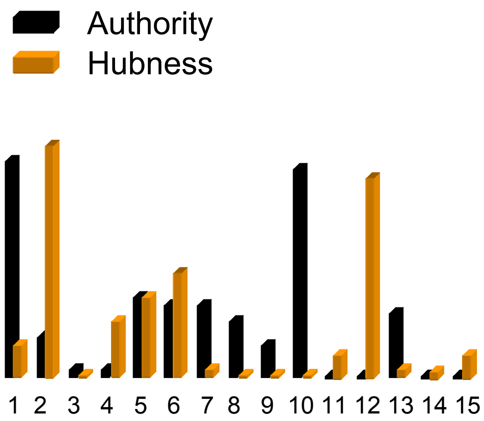
Find a base subgraph:

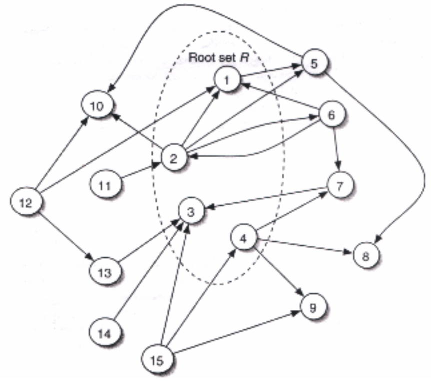
- Start with a root set R {1, 2, 3, 4}
- {1, 2, 3, 4} - nodes relevant to the topic
- Expand the root set R to include all the children and a fixed number of parents of nodes in R

→ A new set S (base subgraph)



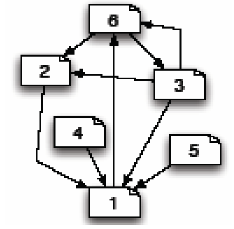
HITS Example Results





Authority and hubness weights

Algebraic Interpretation



Neighborhood graph N

$$A_{u,v} = \begin{cases} 1\, if\ u \rightarrow v \\ 0 \quad otherwise \end{cases}$$

$$A = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}$$

$$Auth(p) = \sum_{q \in N: q \rightarrow p} Hub(q) \qquad Hub(p) = \sum_{q \in N: p \rightarrow q} Auth(q)$$

$$\mathbf{Auth} = \mathbf{A}^T \cdot \mathbf{z} \qquad \mathbf{Hub} = \mathbf{A} \cdot \mathbf{Auth}$$

where **Auth** and **Hub** are the column of authority and hub values respectively and z a column vector of 1s.

Algebraic Interpretation

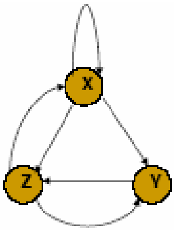
$$\mathbf{Auth}^k \approx \left(\mathbf{A}^T\mathbf{A}\right)^{k-1}\mathbf{A}^T \cdot \mathbf{z} \qquad \mathbf{Hub}^k \approx \left(\mathbf{A}\cdot\mathbf{A}^T\right)^{k-1}\cdot\mathbf{z}$$

- No normalization at each step

$\mathbf{A}^T\mathbf{A}, \quad \mathbf{A}\mathbf{A}^T$ symmetric matrices

- Linear Algebra:
  - If **M** is a symmetric matrix, and *v* is a vector not orthogonal to the principal eigenvector X_1 of **M**, then the unit vector in the direction of $\mathbf{M}^k v$ converges to X_1 as *k* increases without bounds.
- Hub and authority weights are truly an intrinsic feature of the linked pages, not an artifact of the choice of initial weights or the tuning of arbitrary parameters
- Pages with large weights represent a very dense pattern of linkage from pages of large hub weight to pages of large authority weight
- For sample query “search engines” the top authorities returned by HITS were Yahoo Excite Magellan Lycos and AltaVista even though none of these pages contained the phrase search engines

HITS Example



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

$$\mathbf{AUTH}=\begin{bmatrix}2\\2\\2\end{bmatrix}\to\begin{bmatrix}10\\10\\8\end{bmatrix}\to\begin{bmatrix}48\\48\\36\end{bmatrix}\to\begin{bmatrix}228\\228\\168\end{bmatrix}\cdots$$

$$\mathbf{HUB}=\begin{bmatrix}1\\1\\1\end{bmatrix}\to\begin{bmatrix}6\\2\\4\end{bmatrix}\to\begin{bmatrix}28\\8\\20\end{bmatrix}\to\begin{bmatrix}132\\36\\96\end{bmatrix}\to\begin{bmatrix}624\\168\\456\end{bmatrix}\cdots$$

$$\frac{\mathbf{AUTH}}{\|\mathbf{AUTH}\|}=\begin{bmatrix}0.5774\\0.5774\\0.5774\end{bmatrix}\to\begin{bmatrix}0.6155\\0.6155\\0.4924\end{bmatrix}\to\begin{bmatrix}0.6247\\0.6247\\0.4685\end{bmatrix}\to\begin{bmatrix}0.6271\\0.6271\\0.4621\end{bmatrix}\cdots$$

$$\frac{\mathbf{HUB}}{\|\mathbf{HUB}\|}=\begin{bmatrix}0.5774\\0.5774\\0.5774\end{bmatrix}\to\begin{bmatrix}0.8018\\0.2673\\0.5345\end{bmatrix}\to\begin{bmatrix}0.7926\\0.2265\\0.5661\end{bmatrix}\to\begin{bmatrix}0.7898\\0.2154\\0.5744\end{bmatrix}\to\begin{bmatrix}0.7890\\0.2124\\0.5766\end{bmatrix}\cdots$$

The normalized authority and hub values clearly converge

Problems with HITS

- Only a small set of pages (in a neighborhood) are considered
  - Rankings are more vulnerable to manipulation (spamming)
- Hub pages can link to authorities on several different topics
  - Hub score increases, even though many of the links are irrelevant
- Topic drift
  - Majority of pages in neighborhood might focus on a topic different from the query topic
    - Top-ranked pages would then be on this different topic as well
    - topic drift (e.g. from „Jaguar car“ to „car“ in general)
    - Potential Solution: Weighting edges based on how well document content matches query
- Requires computation for each query
- Unlike PageRank, no commercial implementations; real-world performance still uncertain

Ranking in Search Engines

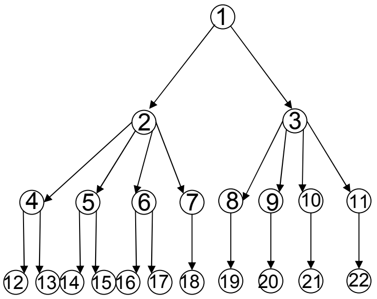
- Lots of variation here
  - Pretty messy in many cases
  - Details usually proprietary and fluctuating
- Combining subsets of:
  - Term frequencies
  - Term proximities
  - Term position (title, top of page, etc)
  - Term characteristics (boldface, capitalized, etc)
  - Link analysis information
  - Category information
  - Popularity information
- Most use a variant of vector space ranking to combine these
- Here’s how it might work:
  - Make a vector of weights for each feature
  - Multiply this by the counts for each feature

Crawling

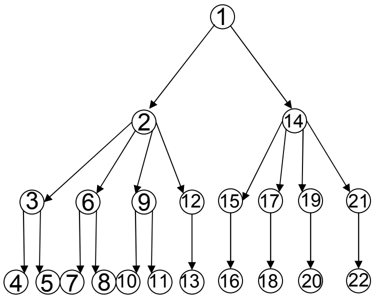
- Begin with an initial set of ‘start’ pages
- Follow all links on these pages recursively to find additional pages.
- Index/Process all **novel** found pages in an inverted index as they are encountered.
- May allow users to directly submit pages to be indexed (and crawled from).

Crawling - Search Strategies

Breadth-first Search



Depth-first Search



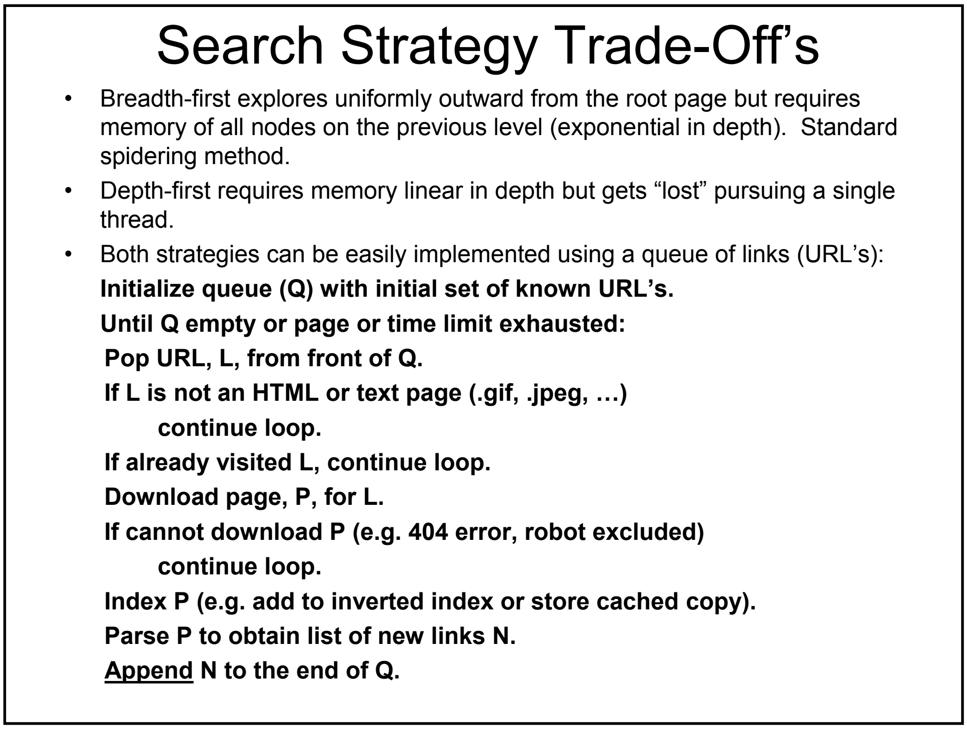
Queueing Strategy

How new links are added to the queue determines search strategy.

FIFO (append to end of Q) gives breadth-first search.

LIFO (add to front of Q) gives depth-first search.

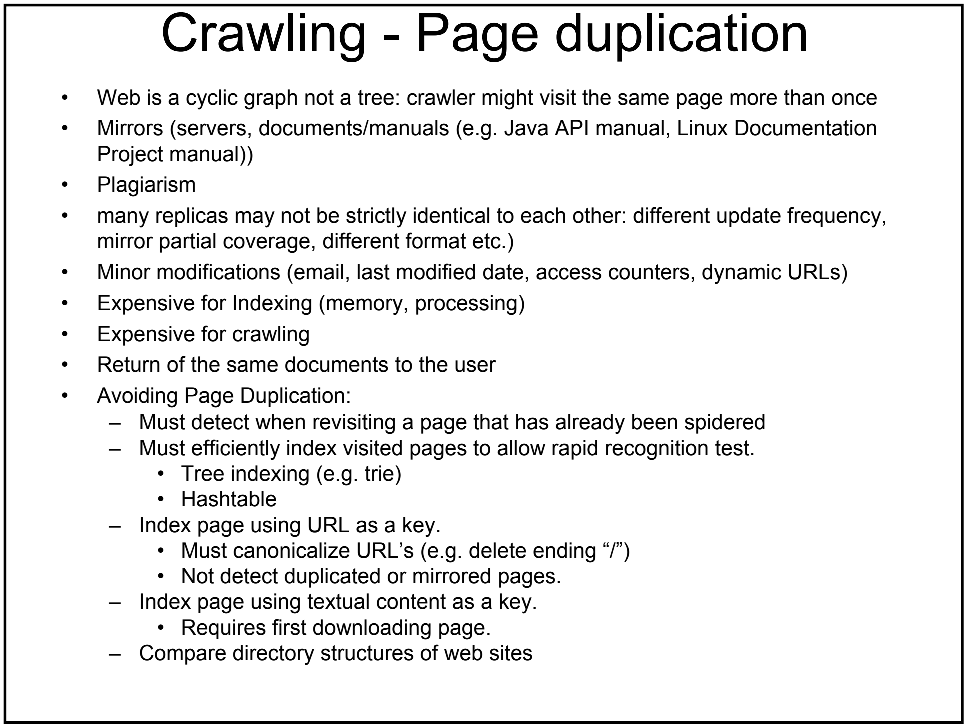
Heuristically ordering the Q gives a “focused crawler” that directs its search towards “interesting” pages.



Search Strategy Trade-Off's

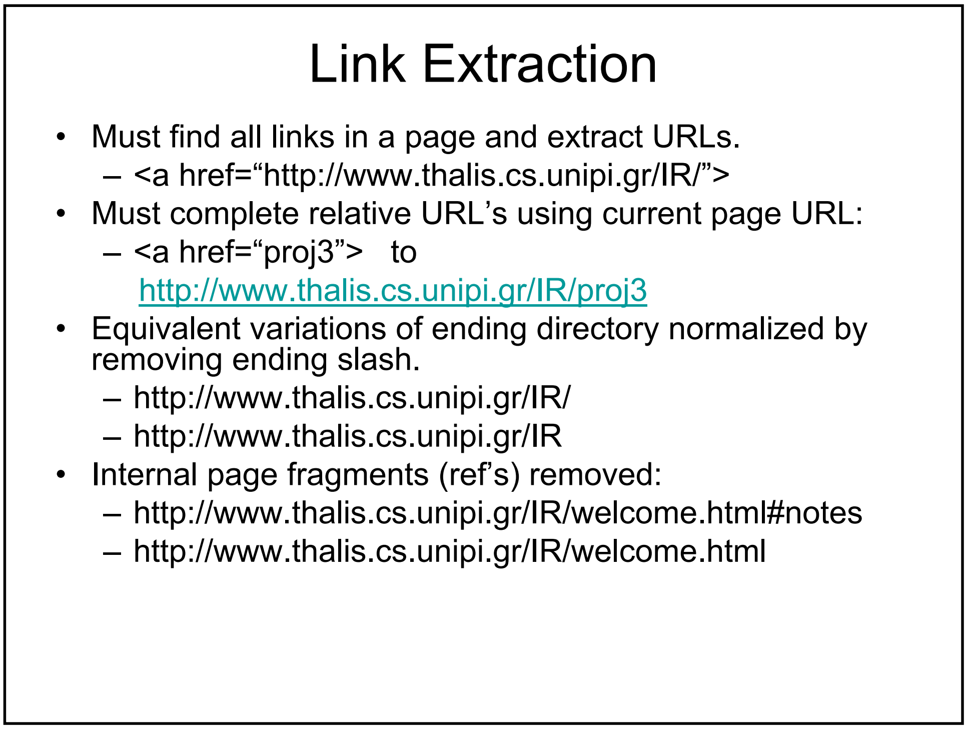
- Breadth-first explores uniformly outward from the root page but requires memory of all nodes on the previous level (exponential in depth). Standard spidering method.
- Depth-first requires memory linear in depth but gets "lost" pursuing a single thread.
- Both strategies can be easily implemented using a queue of links (URL's):

```
Initialize queue (Q) with initial set of known URL's.
Until Q empty or page or time limit exhausted:
  Pop URL, L, from front of Q.
  If L is not an HTML or text page (.gif, .jpeg, …)
      continue loop.
  If already visited L, continue loop.
  Download page, P, for L.
  If cannot download P (e.g. 404 error, robot excluded)
      continue loop.
  Index P (e.g. add to inverted index or store cached copy).
  Parse P to obtain list of new links N.
  Append N to the end of Q.
```



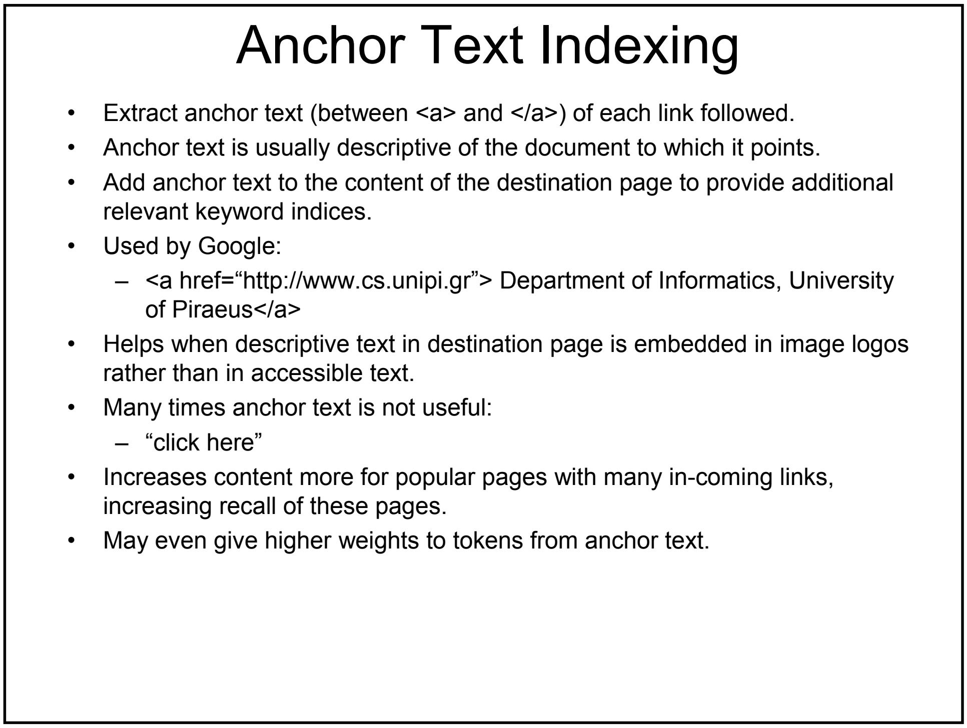
Crawling - Page duplication

- Web is a cyclic graph not a tree: crawler might visit the same page more than once
- Mirrors (servers, documents/manuals (e.g. Java API manual, Linux Documentation Project manual))
- Plagiarism
- many replicas may not be strictly identical to each other: different update frequency, mirror partial coverage, different format etc.)
- Minor modifications (email, last modified date, access counters, dynamic URLs)
- Expensive for Indexing (memory, processing)
- Expensive for crawling
- Return of the same documents to the user
- Avoiding Page Duplication:
  - Must detect when revisiting a page that has already been spidered
  - Must efficiently index visited pages to allow rapid recognition test.
    - Tree indexing (e.g. trie)
    - Hashtable
  - Index page using URL as a key.
    - Must canonicalize URL's (e.g. delete ending "/")
    - Not detect duplicated or mirrored pages.
  - Index page using textual content as a key.
    - Requires first downloading page.
  - Compare directory structures of web sites



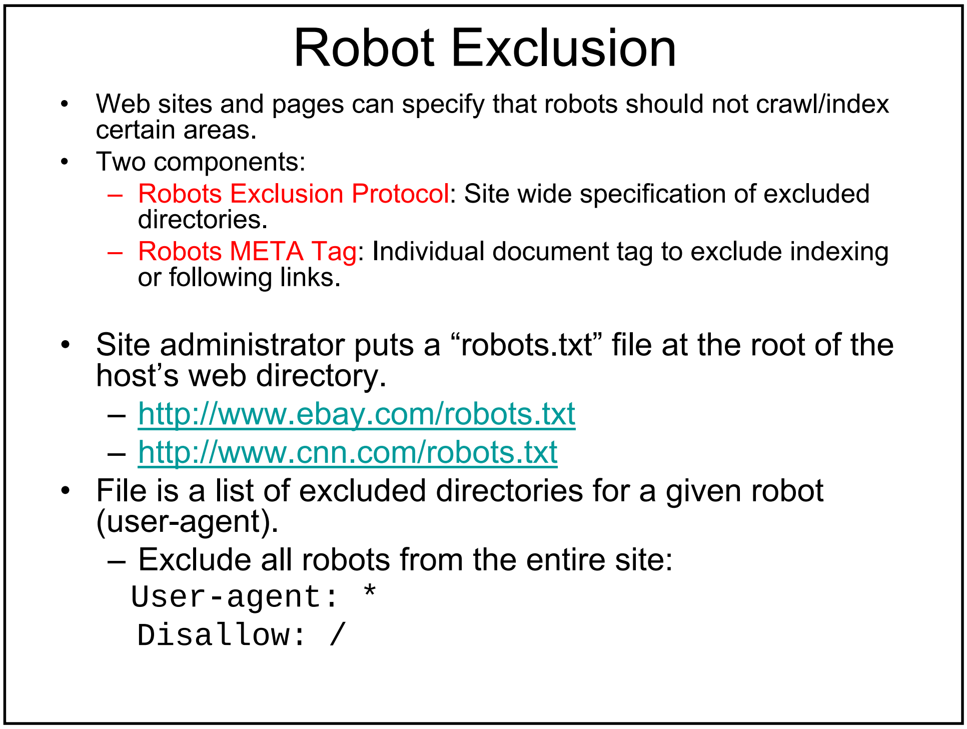
Link Extraction

- Must find all links in a page and extract URLs.
  - <a href=“http://www.thalis.cs.unipi.gr/IR/”>
- Must complete relative URL’s using current page URL:
  - <a href=“proj3”> to http://www.thalis.cs.unipi.gr/IR/proj3
- Equivalent variations of ending directory normalized by removing ending slash.
  - http://www.thalis.cs.unipi.gr/IR/
  - http://www.thalis.cs.unipi.gr/IR
- Internal page fragments (ref’s) removed:
  - http://www.thalis.cs.unipi.gr/IR/welcome.html#notes
  - http://www.thalis.cs.unipi.gr/IR/welcome.html



Anchor Text Indexing

- Extract anchor text (between <a> and </a>) of each link followed.
- Anchor text is usually descriptive of the document to which it points.
- Add anchor text to the content of the destination page to provide additional relevant keyword indices.
- Used by Google:
  - <a href=“http://www.cs.unipi.gr”> Department of Informatics, University of Piraeus</a>
- Helps when descriptive text in destination page is embedded in image logos rather than in accessible text.
- Many times anchor text is not useful:
  - “click here”
- Increases content more for popular pages with many in-coming links, increasing recall of these pages.
- May even give higher weights to tokens from anchor text.

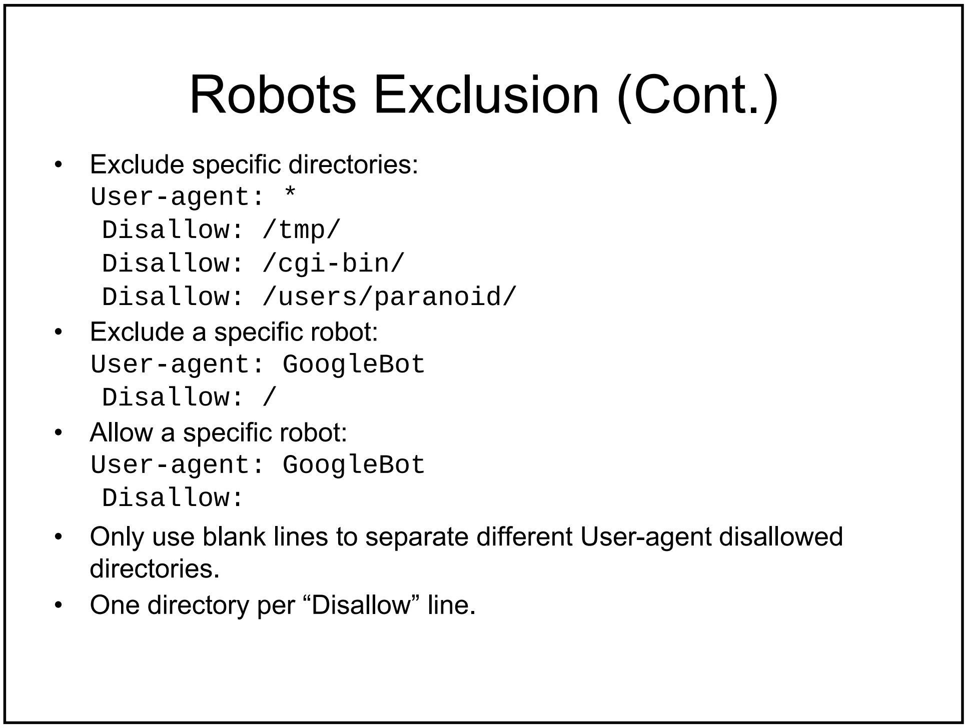


Robot Exclusion

- Web sites and pages can specify that robots should not crawl/index certain areas.
- Two components:
  - Robots Exclusion Protocol: Site wide specification of excluded directories.
  - Robots META Tag: Individual document tag to exclude indexing or following links.

- Site administrator puts a “robots.txt” file at the root of the host’s web directory.
  - http://www.ebay.com/robots.txt
  - http://www.cnn.com/robots.txt
- File is a list of excluded directories for a given robot (user-agent).
  - Exclude all robots from the entire site:

```
User-agent: *
 Disallow: /
```



Robots Exclusion (Cont.)

- Exclude specific directories:

```
User-agent: *
 Disallow: /tmp/
 Disallow: /cgi-bin/
 Disallow: /users/paranoid/
```

- Exclude a specific robot:

```
User-agent: GoogleBot
 Disallow: /
```

- Allow a specific robot:

```
User-agent: GoogleBot
 Disallow:
```

- Only use blank lines to separate different User-agent disallowed directories.
- One directory per “Disallow” line.

Robots META Tag

- Include META tag in HEAD section of a specific HTML document.
  - <meta name=“robots” content=“none”>
- Content value is a pair of values for two aspects:
  - index | noindex: Allow/disallow indexing of this page.
  - follow | nofollow: Allow/disallow following links on this page.
- Special values:
  - all = index,follow
  - none = noindex,nofollow
- Examples:

  <meta name=“robots” content=“noindex,follow”>

  <meta name=“robots” content=“index,nofollow”>

  <meta name=“robots” content=“none”>

Robot Exclusion Issues

- META tag is newer and less well-adopted than "robots.txt".
- Standards are conventions to be followed by "good robots."
- Companies have been prosecuted for "disobeying" these conventions and "trespassing" on private cyberspace.

Multi-Threaded Spidering

- Bottleneck is network delay in downloading individual pages.
- Best to have multiple threads running in parallel each requesting a page from a different host.
- Distribute URL's to threads to guarantee equitable distribution of requests across different hosts to maximize through-put and avoid overloading any single server.
- Early Google spider had multiple co-ordinated crawlers with about 300 threads each, together able to download over 100 pages per second.

Directed/Focused Spidering

- Sort queue to explore more "interesting" pages first.
- Three styles of focus:
  - Topic-Directed
    - Assume desired topic description or sample pages of interest are given.
    - Sort queue of links by the similarity (e.g. cosine metric) of their source pages and/or anchor text to this topic description.
  - Link-Directed
    - Monitor links and keep track of in-degree and out-degree of each page encountered.
    - Sort queue to prefer popular pages with many in-coming links (*authorities*).
    - Sort queue to prefer summary pages with many out-going links (*hubs*).
      - Google's PageRank algorithm
  - Location Directed:
    - (content doesn't matter, only the URL does
      - e.g. .com more important than .biz

Keeping Spidered Pages Up to Date

- Web is very dynamic: many new pages, updated pages, deleted pages, etc.
- Periodically check spidered pages for updates and deletions:
  - Just look at header info (e.g. META tags on last update) to determine if page has changed, only reload entire page if needed.
- Track how often each page is updated and preferentially return to pages which are historically more dynamic.
- Preferentially update pages that are accessed more often to optimize freshness of more popular pages.

Web Spam

- What are the types of Web spam?
  - Add extra terms to get a higher ranking
    - Repeat “cars” thousands of times
  - Add irrelevant terms to get more hits
    - Put a dictionary in the comments field
    - Put extra terms in the same color as the background of the web page
  - Add irrelevant terms to get different types of hits
  - Add irrelevant links to boost your link analysis ranking
- There is a constant “arms race” between web search companies and spammers

Web Spam

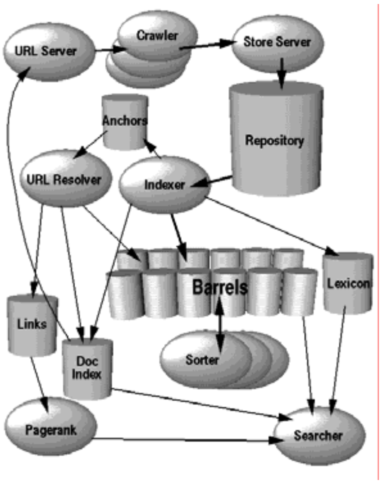
- Most search engines have rules against
  - invisible text
  - meta tag abuse
  - heavy repetition
  - "domain spam”
    - overtly submission of "mirror“ sites in an attempt to dominate the listings for particular terms

Web Spam

- Excite screens out spamming before adding a page to its web page index.
  - if it finds a string of words such as:
  - money money money money money money money
  - it will replace the excess repetition, so that essentially, the string becomes:
  - money xxxxx xxxxx xxxxx xxxxx xxxxx xxxxx
- The more Excite detects unusual repetition, the more heavily it will penalize a page.
- Excite does not penalize for the use of hidden text, but penalties will apply if hidden text is used to disguise spam content.
- Excite penalises "domain spam."

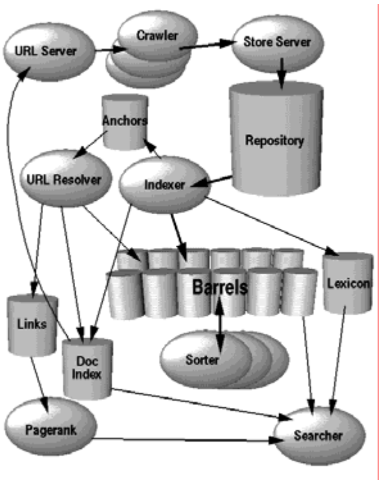
Google Architecture Overview

- URL server
  - Sends URLs to be fetched to crawlers
- Crawler
  - Downloads web pages
  - Done by several distributed crawlers
- Store Server
  - Compresses and stores web pages
- Repository
  - Each web page associated with docID



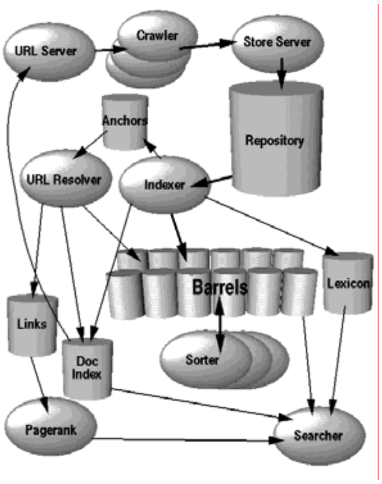
Google Architecture Overview

- Indexer
  - Reads repository, uncompresses docs, parses them
  - Each doc converted to set of word occurences, “Hits”
    - Record word, position, font, capitalization
    - Distributes hits into set of “Barrels” creating partially sorted forward index
  - Parses all links in web pages and stores them in “Anchor File”
    - Contains enough info to determine where each link points from and to and text of link



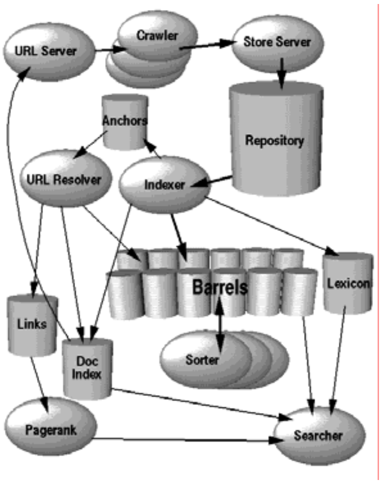
Google Architecture Overview

- URL Resolver
  - Reads anchor files
  - Converts relative URLs to absolute URLs and in turn into docIDs
  - Puts anchor text into forward index, associated with docID that the anchor points to
  - Generates database of link (pairs of docIDs, used to compute PR)



Google Architecture Overview

- Sorter
  - Takes barrels sorted by docID
  - Resorts by wordID to generate inverted index
  - Also produces list of wordIDs and offsets into inverted index
- DumpLexicon
  - Takes above list along with lexicon produced by indexer
  - Generates new lexicon for use by “Searcher”
- Searcher
  - Uses above lexicon with inverted index and PR to answer queries



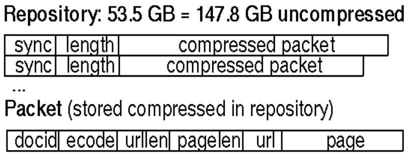
Major Data Structures

1. **Big Files**
   - Virtual files spanning multiple file systems
   - Addressable by 64 bit integers
   - File system allocation handled automatically
   - Handles allocation and deallocation of file descriptors
   - Support rudimentary compression options

Major Data Structures

2. Repository

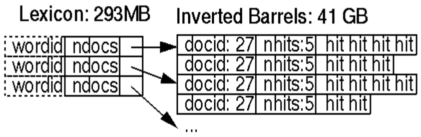
- Contains full HTML of every web page
- Each page compressed with zlib
- Docs stored one after another
- Prefix : docID, length, URL
- Requires no other data structure to be accessed



Major Data Structures

4. Lexicon

- Fits in main memory
- Contains excess of 14 million words
- Implemented as:
  - List of words (concatenated, but separated by nulls)
  - Hash table of pointers


Lexicon: 293MB
Inverted Barrels: 41 GB
wordid ndocs
wordid ndocs
wordid ndocs
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit hit
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit
...

Major Data Structures

3. Document Index

- Keeps information about each doc
- It's a fixed width ISAM (Index Sequential Access Mode) index, ordered by docID
- Each entry includes current doc status, pointer to repository, doc checksum
- If doc crawled, contains pointer to variable width file, docinfo (contains its URL, title)
- Else, just contains URL

Major Data Structures

5. Hit Lists

- Occurrences of words in doc with position, font, capitalization info
- Accounts for most space in forward, inverted indices
- Uses hand optimized compact encoding
- Types:
  - Fancy Hits
    - Hits occurring in URL, title, anchor text, meta tag
    - Capitalization Bit + Font Size 7 + 4 Bits to encode type, Position (8 Bits)
    - Anchor Hits: Position bits split as 4 bits anchor position + 4 Bits docID hash of anchor
  - Plain Hits
    - Capitalization bit + Font size (Relative, 3 Bits) + Word position (12 Bits)

Hit: 2 bytes

plain:	cap:1	imp:3	position: 12		
fancy:	cap:1	imp = 7	type: 4	position: 8	
anchor:	cap:1	imp = 7	type: 4	hash:4	pos: 4

Major Data Structures

6. Forward Index

- Partially sorted
- Stored in a no. of barrels (~64)
- Each barrel holds range of wordIDs
- Barrel stores docID of doc containing word + list of wordIDs + Hit lists
- Instead of actual wordID, relative difference from minimum Barrel wordID stored
- Leaves 8 bits for Hit list length

Forward Barrels: total 43 GB

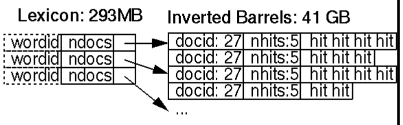
docid	wordid: 24	nhits: 8	hit hit hit hit
	wordid: 24	nhits: 8	hit hit hit hit
	null wordid		
docid	wordid: 24	nhits: 8	hit hit hit hit
	wordid: 24	nhits: 8	hit hit hit hit
	wordid: 24	nhits: 8	hit hit hit hit
	null wordid		

...

Major Data Structures

7. Inverted Index

- Consists of same barrels as forward index that are already processed by sorter
- For valid wordID, lexicon contains pointer to Barrel that wordID falls into
- Points to doclist of docIDs + hit lists
- Doclist represents all occurences of that word in all docs



Lexicon: 293MB

wordid	ndocs
wordid	ndocs
wordid	ndocs

Inverted Barrels: 41 GB

docid: 27	nhits:5	hit hit hit hit
docid: 27	nhits:5	hit hit hit
docid: 27	nhits:5	hit hit hit hit
docid: 27	nhits:5	hit hit

...

Major Applications

- Crawling the Web
  - Uses fast distributed crawling system
  - Each crawler maintains its own DNS cache
- Indexing the Web
  - Parsing
  - Indexing Documents into Barrels
  - Sorting
- Searching

Google Query Evaluation

1. Parse the query.
2. Convert words into wordIDs.
3. Seek to the start of the doclist in the short barrel for every word.
4. Scan through the doclists until there is a document that matches all the search terms.
5. Compute the rank of that document for the query.
6. If we are in the short barrels and at the end of any doclist, seek to the start of the doclist in the full barrel for every word and go to step 4.
7. If we are not at the end of any doclist go to step 4. Sort the documents that have matched by rank and return the top k.

Single Word Query Ranking

- Hitlist is retrieved for single word
- Each hit can be one of several types: title, anchor, URL, large font, small font, etc.
- Each hit type is assigned its own weight
- Type-weights make up vector of weights
- # of hits of each type is counted to form count vector
- Dot product of two vectors is used to compute IR score
- IR score is combined with PageRank to compute final rank

Multi-word Query Ranking

- Similar to single-word ranking except now must analyze proximity
- Hits occurring closer together are weighted higher
- Each proximity relation is classified into 1 of 10 values ranging from a phrase match to “not even close”
- Counts are computed for every type of hit and proximity

Summary of Key Optimization Techniques

– Each crawler maintains its own DNS lookup cache

– Parallelization of indexing phase

– In-memory lexicon

– Compression of repository

– Compact encoding of hitlists accounting for major space savings

– Indexer is optimized so it is just faster than the crawler so that crawling is the bottleneck

– Document index is updated in bulk

– Critical data structures placed on local disk

– Overall architecture designed to avoid disk seeks wherever possible