Data Mining

Google Link Analysis

Web Search

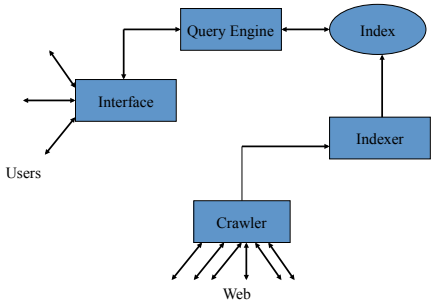
Goal

Provide information discovery for large amounts of open access material on the web

Challenges

- Volume of material -- several billion items, growing steadily
- Items created dynamically or in databases
- Great variety -- length, formats, quality control, purpose, etc.
- Inexperience of users -- range of needs
- Economic models to pay for the service

A Typical Web Search Engine



Index construction

- Documents are parsed to extract words and these are saved with the Document ID.

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Index construction

- After all documents have been parsed, the inverted file is sorted by terms.

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Index construction

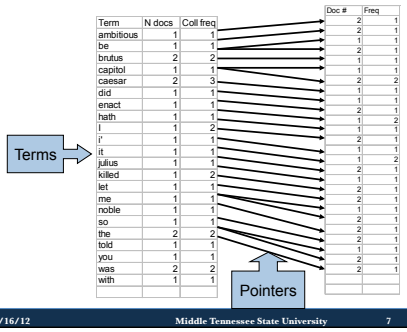
The result is split into a *Dictionary* file and a *Postings* file.

Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Term	N docs	Coll freq
ambitious	1	1
be	1	1
brutus	2	2
capitol	1	1
caesar	2	3
did	1	1
enact	1	1
hath	1	1
I	1	2
i'	1	1
it	1	1
julius	1	1
killed	1	2
let	1	1
me	1	1
noble	1	1
so	1	1
the	2	2
told	1	1
you	1	1
was	2	2
with	1	1

Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1

Index construction



Motivation for Link Analysis

- Early search engines mainly compare *content similarity* of the query and the indexed pages. i.e.,
  - They use information retrieval methods, cosine, TF-IDF, ...
- From mid 90's, it became clear that content similarity alone was no longer sufficient.
  - The number of pages grew rapidly in the mid-late 1990's.
    - Try "classification methods", Google estimates: million relevant pages.
    - How to choose only 30-40 pages and rank them suitably to present to the user?
  - Content similarity is easily spammed.
    - A page owner can repeat some words and add many related words to boost the rankings of his pages and/or to make the pages relevant to a large number of queries.

Early hyperlinks

- Starting mid 90's, researchers began to work on the problem, resorting to hyperlinks.
  - In Feb, 1997, Yanhong Li (Scotch Plains, NJ) filed a hyperlink based search patent. The method uses words in anchor text of hyperlinks.
- Web pages on the other hand are connected through hyperlinks, which carry important information.
  - Some hyperlinks: organize information at the same site.
  - Other hyperlinks: point to pages from other Web sites. Such out-going hyperlinks often indicate an implicit conveyance of authority to the pages being pointed to.
- Those pages that are pointed to by many other pages are likely to contain authoritative information.

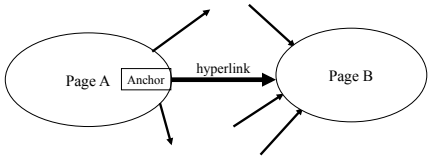
Hyperlink algorithms

- During 1997-1998, two most influential hyperlink based search algorithms PageRank and HITS were reported.
- Both algorithms are related to **social networks**. They exploit the hyperlinks of the Web to rank pages according to their levels of "prestige" or "authority".
  - **HITS**: Jon Kleinberg (Cornel University), at *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1998
  - **PageRank**: Sergey Brin and Larry Page, PhD students from Stanford University, at *Seventh International World Wide Web Conference* (WWW7) in April, 1998.
- PageRank powers the Google search engine.

Other uses

- Apart from search ranking, hyperlinks are also useful for finding Web communities.
  - A Web community is a cluster of densely linked pages representing a group of people with a special interest.
- Beyond explicit hyperlinks on the Web, links in other contexts are useful too, e.g.,
  - for discovering communities of named entities (e.g., people and organizations) in free text documents, and
  - for analyzing social phenomena in emails.

The Web as a Directed Graph



Assumption 1: A hyperlink between pages denotes author perceived relevance (quality signal)

Assumption 2: The anchor of the hyperlink describes the target page (textual context)

Anchor Text Indexing

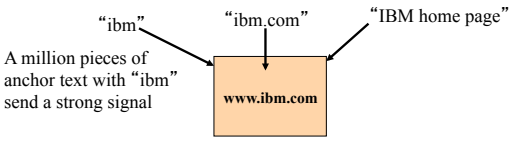
- Extract anchor text (between <a> and) of each link followed.
- Anchor text is usually descriptive of the document to which it points.
- Add anchor text to the content of the destination page to provide additional relevant keyword indices.
- Used by Google:
  - Evil Empire
  - IBM

Anchor text

Anchor Text

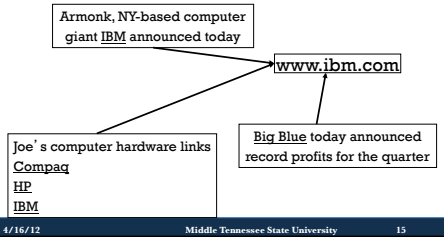
WWW Worm - McBryan [Mcbr94]

- For ***ibm*** how to distinguish between:
  - IBM's home page (mostly graphical)
  - IBM's copyright page (high term freq. for 'ibm')
  - Rival's spam page (arbitrarily high term freq.)



Indexing anchor text

- When indexing a document *D*, include anchor text from links pointing to *D*.



History of link analysis

- Bibliometrics
  - Citation analysis since the 1960's
  - Citation links to and from documents
    - Basis of pagerank idea

Bibliometrics

Techniques that use citation analysis to measure the similarity of journal articles or their importance

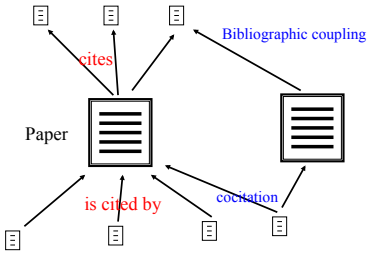
Bibliographic coupling: two papers that cite many of the same papers

Co-citation: two papers that were cited by many of the same papers

Impact factor (of a journal): frequency with which the average article in a journal has been cited in a particular year or period

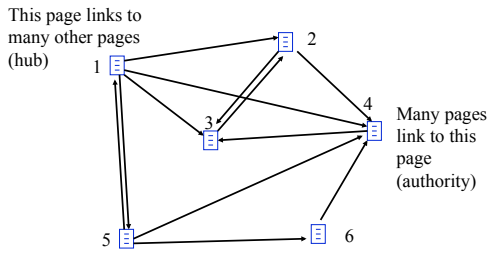
Citation frequency

Citation Graph



Note that journal citations nearly always refer to earlier work.

Graphical Analysis of Hyperlinks on the Web



Bibliometrics: Citation Analysis

- Many standard documents include *bibliographies* (or *references*), explicit *citations* to other previously published documents.
- Using citations as links, standard corpora can be viewed as a graph.
- The structure of this graph, independent of content, can provide interesting information about the similarity of documents and the structure of information.
- Impact of paper!

Impact Factor

- Developed by Garfield in 1972 to measure the importance (quality, influence) of scientific journals.
- Measure of how often papers in the journal are cited by other scientists.
- Computed and published annually by the Institute for Scientific Information (ISI).
- The *impact factor* of a journal *J* in year *Y* is the average number of citations (from indexed documents published in year *Y*) to a paper published in *J* in year *Y*-1 or *Y*-2.
- Does not account for the quality of the citing article.

Citations vs. Links

- Web links are a bit different than citations:
  - Many links are navigational.
  - Many pages with high in-degree are portals not content providers.
  - Not all links are endorsements.
  - Company websites don't point to their competitors.
  - Citations to relevant literature is enforced by peer-review.

Authorities

- *Authorities* are pages that are recognized as providing significant, trustworthy, and useful information on a topic.
- *In-degree* (number of pointers to a page) is one simple measure of authority.
- However in-degree treats all links as equal.
- Should links from pages that are themselves authoritative count more?

Hubs

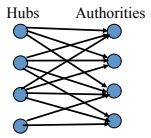
- *Hubs* are index pages that provide lots of useful links to relevant content pages (topic authorities).
- Ex: pages are included in the course home page

HITS

- Algorithm developed by Kleinberg in 1998.
- IBM search engine project
- Attempts to computationally determine hubs and authorities on a particular topic through analysis of a relevant subgraph of the web.
- Based on mutually recursive facts:
  - Hubs point to lots of authorities.
  - Authorities are pointed to by lots of hubs.

Hubs and Authorities

- Together they tend to form a bipartite graph:

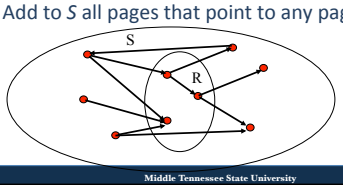


HITS Algorithm

- Computes hubs and authorities for a particular topic specified by a normal query.
- First determines a set of relevant pages for the query called the *base* set *S*.
- Analyze the link structure of the web subgraph defined by *S* to find authority and hub pages in this set.

Constructing a Base Subgraph

- For a specific query *Q*, let the set of documents returned by a standard search engine be called the *root* set *R*.
- Initialize *S* to *R*.
- Add to *S* all pages pointed to by any page in *R*.
- Add to *S* all pages that point to any page in *R*.



Base Limitations

- To limit computational expense:
  - Limit number of root pages to the top 200 pages retrieved for the query.
  - Limit number of "back-pointer" pages to a random set of at most 50 pages returned by a "reverse link" query.
- To eliminate purely navigational links:
  - Eliminate links between two pages on the same host.
- To eliminate "non-authority-conveying" links:
  - Allow only *m* (*m* ≅ 4-8) pages from a given host as pointers to any individual page.

Authorities and In-Degree

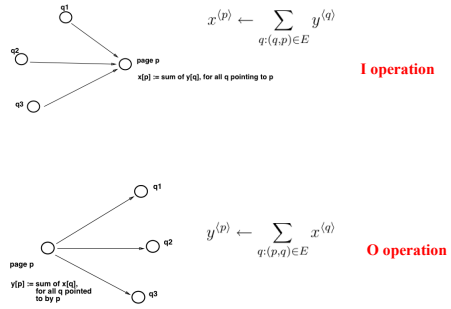
- Even within the base set *S* for a given query, the nodes with highest in-degree are not necessarily authorities (may just be generally popular pages like Yahoo or Amazon).
- True authority pages are pointed to by a number of hubs (i.e. pages that point to lots of authorities).

Iterative Algorithm

- Use an iterative algorithm to slowly converge on a mutually reinforcing set of hubs and authorities.
- Maintain for each page $p \in S$:
  - Authority score: x_p (vector $\boldsymbol{x}$)
  - Hub score: y_p (vector $\boldsymbol{y}$)
- Initialize all $x_p = y_p = 1$
- Maintain normalized scores:

$$\sum_{p \in S} (x_p)^2 = 1 \qquad \sum_{p \in S} (y_p)^2 = 1$$

Iterative Algorithm (I and O operation)



Convergence

- Algorithm converges to a *fix-point* if iterated indefinitely.
- Define A to be the adjacency matrix for the subgraph defined by S.
  - $A_{ij} = 1$ for $i \in S$, $j \in S$ iff $i \rightarrow j$
- Authority vector, $\boldsymbol{a}$, converges to the principal eigenvector of $A^T A$
- Hub vector, $\boldsymbol{h}$, converges to the principal eigenvector of AA^T
- In practice, 20 iterations produces fairly stable results.

Google Background

"Our main goal is to improve the quality of web search engines"

- Google ← googol = 10^100
- Originally part of the Stanford digital library project known as WebBase, commercialized in 1999

Initial Design Goals

- Deliver results that have very high precision even at the expense of recall
- Make search engine technology transparent, i.e. advertising shouldn't bias results
- Bring search engine technology into academic realm in order to support novel research activities on large web data sets
- Make system easy to use for most people, e.g. users shouldn't have to specify more than a couple words

Google Search Engine Features

Two main features to increase result precision:

- Uses link structure of web (PageRank)
- Uses text surrounding hyperlinks to improve accurate document retrieval

Other features include:

- Takes into account word proximity in documents
- Uses font size, word position, etc. to weight word
- Storage of full raw html pages

PageRank in Words

Intuition:

- Imagine a web surfer doing a simple random walk on the entire web for an infinite number of steps.
- Occasionally, the surfer will get bored and instead of following a link pointing outward from the current page will jump to another random page.
- At some point, the percentage of time spent at each page will converge to a fixed value.
- This value is known as the PageRank of the page.

PageRank

- Link-analysis method used by Google (Brin & Page, 1998).
- Does not attempt to capture the distinction between hubs and authorities.
- Ranks pages just by authority.
- Applied to the entire web rather than a local neighborhood of pages surrounding the results of a query.

Initial PageRank Idea

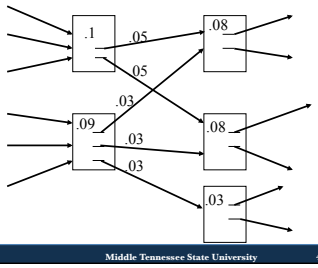
- Just measuring in-degree (citation count) doesn't account for the authority of the source of a link.
- Initial page rank equation for page *p*:

$$R(p) = c \sum_{q:q \to p} \frac{R(q)}{N_q}$$

  - N_q is the total number of out-links from page *q*.
  - A page, *q*, "gives" an equal fraction of its authority to all the pages it points to (e.g. *p*).
  - *c* is a normalizing constant set so that the rank of all pages always sums to 1.

Initial PageRank Idea (cont.)

- Can view it as a process of PageRank "flowing" from pages to the pages they cite.



Initial Algorithm

- Iterate rank-flowing process until convergence:

  Let *S* be the total set of pages.

  Initialize $\forall p \in S$: $R(p) = 1/|S|$

  Until ranks do not change (much) (*convergence*)

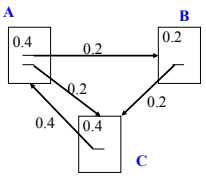
  For each $p \in S$:

$$R'(p) = \sum_{q:q \to p} \frac{R(q)}{N_q}$$

$$c = 1 / \sum_{p \in S} R'(p)$$

  For each $p \in S$: $R(p) = cR'(p)$ (*normalize*)

Sample Stable Fixed Point

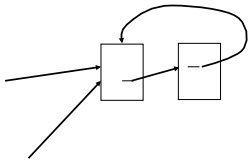


Linear Algebra Version

- Treat **R** as a vector over web pages.
- Let **A** be a 2-d matrix over pages where
  - $\mathbf{A}_{uv} = 1/N_u$ if $u \rightarrow v$ else $\mathbf{A}_{uv} = 0$
- Then $\mathbf{R} = c\mathbf{AR}$
- **R** converges to the principal eigenvector of **A**.

Problem with Initial Idea

- A group of pages that only point to themselves but are pointed to by other pages act as a "rank sink" and absorb all the rank in the system.



Rank flows into cycle and can't get out

Rank Source

- Introduce a "rank source" E that continually replenishes the rank of each page, p, by a fixed amount $E(p)$.

$$R(p) = c\left(\sum_{q:q\rightarrow p} \frac{R(q)}{N_q} + E(p)\right)$$

PageRank Algorithm

Let S be the total set of pages.
Let $\forall p \in S$: $E(p) = \alpha/|S|$ (for some $0<\alpha<1$, e.g. 0.15)
Initialize $\forall p \in S$: $R(p) = 1/|S|$
Until ranks do not change (much) *(convergence)*

For each $p \in S$:

$$R'(p) = \sum_{q:q\rightarrow p} \frac{R(q)}{N_q} + E(p)$$

$$c = 1/\sum_{p\in S} R'(p)$$

For each $p \in S$: $R(p) = cR'(p)$ *(normalize)*

Linear Algebra Version

- $\mathbf{R} = c(\mathbf{AR} + \mathbf{E})$
- Since $\mathbf{R} = c(\mathbf{A} + \mathbf{E}\times\mathbf{1})\mathbf{R}$
  - Where **1** is the vector consisting of all 1's.
- **R** is an eigenvector of $(\mathbf{A} + \mathbf{E}\times\mathbf{1})$

Random Surfer Model

- PageRank can be seen as modeling a "random surfer" that starts on a random page and then at each point:
  - With probability $E(p)$ randomly jumps to page p.
  - Otherwise, randomly follows a link on the current page.
- $R(p)$ models the probability that this random surfer will be on page p at any given time.
- "E jumps" are needed to prevent the random surfer from getting "trapped" in web sinks with no outgoing links.

Justifications for using PageRank

- Attempts to model user behavior
- Captures the notion that the more a page is pointed to by "important" pages, the more it is worth looking at
- Takes into account global structure of web

Speed of Convergence

- Early experiments on Google used 322 million links.
- PageRank algorithm converged (within small tolerance) in about 52 iterations.
- Number of iterations required for convergence is empirically O(log *n*) (where *n* is the number of links).
- Therefore calculation is quite efficient.

Google Ranking

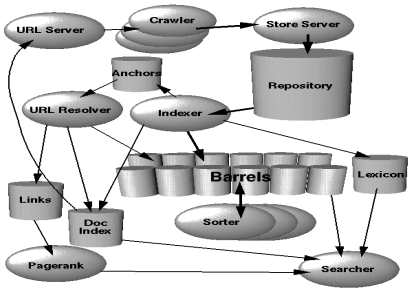
- Complete Google ranking includes (based on university publications prior to commercialization).
  - Vector-space similarity component.
  - Keyword proximity component.
  - HTML-tag weight component (e.g. title preference).
  - PageRank component.
- Details of current commercial ranking functions are trade secrets.
  - Pagerank becomes Googlerank!

Link Analysis Conclusions

- Link analysis uses information about the structure of the web graph to aid search.
- It is one of the major innovations in web search.
- It is the primary reason for Google's success.
- Still lots of research regarding improvements

Search Engine Basics

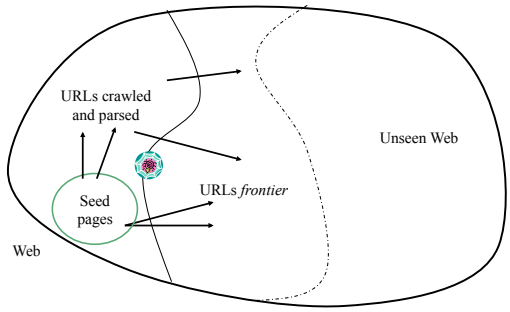
Implemented in Perl, C and C++ on Solaris and Linux



Basic crawler operation

- Begin with known "seed" pages
- Fetch and parse them
  - Extract URLs they point to
  - Place the extracted URLs on a queue
- Fetch each URL on the queue and repeat

Crawling picture



Simple picture – complications

- Web crawling isn't feasible with one machine
  - All of the above steps distributed
- Even non-malicious pages pose challenges
  - Latency/bandwidth to remote servers vary
  - Webmasters' stipulations
    - How "deep" should you crawl a site's URL hierarchy?
  - Site mirrors and duplicate pages
- Malicious pages
  - Spam pages
  - Spider traps – incl dynamically generated
- Politeness – don't hit a server too often

What any crawler *must* do

- Be Polite: Respect implicit and explicit politeness considerations
  - Only crawl allowed pages
  - Respect *robots.txt* (more on this shortly)
- Be Robust: Be immune to spider traps and other malicious behavior from web servers

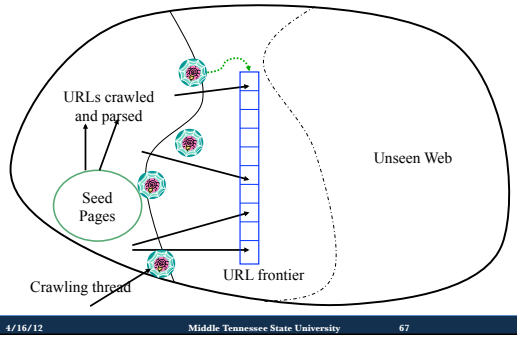
What any crawler *should* do

- Be capable of distributed operation: designed to run on multiple distributed machines
- Be scalable: designed to increase the crawl rate by adding more machines
- Performance/efficiency: permit full use of available processing and network resources

What any crawler *should* do

- Fetch pages of "higher quality" first
- Continuous operation: Continue fetching fresh copies of a previously fetched page
- Extensible: Adapt to new data formats, protocols

Updated crawling picture



URL frontier

- Can include multiple pages from the same host
- Must avoid trying to fetch them all at the same time
- Must try to keep all crawling threads busy

Explicit and implicit politeness

- Explicit politeness: specifications from webmasters on what portions of site can be crawled
  - robots.txt
- Implicit politeness: even with no specification, avoid hitting any site too often

Robots.txt

- Protocol for giving spiders ("robots") limited access to a website, originally from 1994
  - www.robotstxt.org/wc/norobots.html
- Website announces its request on what can(not) be crawled
  - For a URL, create a file `URL/robots.txt`
  - This file specifies access restrictions

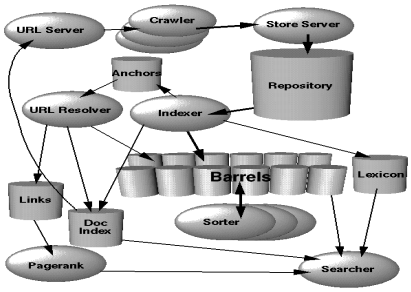
Robots.txt example

- No robot should visit any URL starting with "/yoursite/temp/", except the robot called "searchengine":

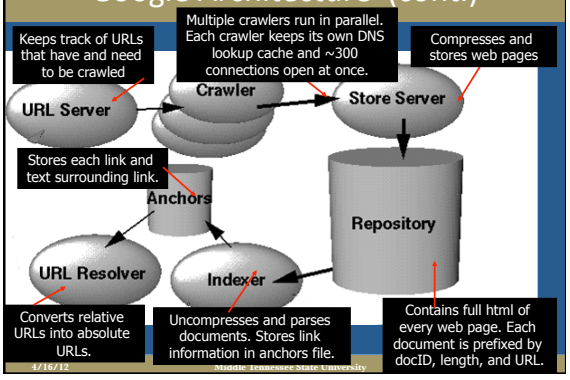
```
User-agent: *
Disallow: /yoursite/temp/

User-agent: searchengine
Disallow:
```


Google Architecture

Implemented in Perl, C and C++ on Solaris and Linux



Google Architecture (cont.)

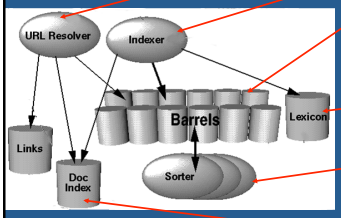


Preliminary

"Hitlist" is defined as list of occurrences of a particular word in a particular document including additional meta info:

- position of word in doc
- font size
- capitalization
- descriptor type, e.g. title, anchor, etc.

Google Architecture (cont.)



Maps absolute URLs into docIDs stored in Doc Index. Stores anchor text in "barrels". Generates database of links (pairs of docIds).

Parses & distributes hit lists into "barrels."

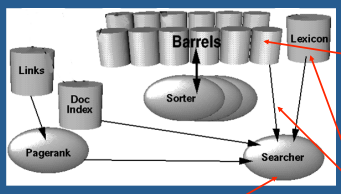
Partially sorted forward indexes sorted by docID. Each barrel stores hitlists for a given range of wordIDs.

In-memory hash table that maps words to wordIds. Contains pointer to doclist in barrel which wordId falls into.

Creates inverted index whereby document list containing docID and hitlists can be retrieved given wordID.

DocID keyed index where each entry includes info such as pointer to doc in repository, checksum, statistics, status, etc. Also contains URL info if doc has been crawled. If not just contains URL.

Google Architecture (cont.)



2 kinds of barrels. Short barrell which contain hit list which include title or anchor hits. Long barrell for all hit lists.

List of wordIDs produced by Sorter and lexicon created by Indexer used to create new lexicon used by searcher. Lexicon stores ~14 million words.

New lexicon keyed by wordID, inverted doc index keyed by docID, and PageRanks used to answer queries