Web Search Engines

Chapter 27, Part C
Based on Larson and Hearst’s slides at UC-Berkeley

http://www.sims.berkeley.edu/courses/is202/f00/

Search Engine Characteristics

- Unedited – anyone can enter content
  - Quality issues; Spam
- Varied information types
  - Phone book, brochures, catalogs, dissertations, news reports, weather, all in one place!
- Different kinds of users
  - Lexis-Nexis: Paying, professional searchers
  - Online catalogs: Scholars searching scholarly literature
  - Web: Every type of person with every type of goal
- Scale
  - Hundreds of millions of searches/day; billions of docs

Web Search Queries

- Web search queries are short:
  - ~2.4 words on average (Aug 2000)
  - Has increased, was 1.7 (~1997)
- User Expectations:
  - Many say “The first item shown should be what I want to see!”
  - This works if the user has the most popular/common notion in mind, not otherwise.
Directories vs. Search Engines

- **Directories**
  - Hand-selected sites
  - Search over the contents of the *descriptions* of the pages
  - Organized in advance into categories

- **Search Engines**
  - All pages in all sites
  - Search over the contents of the *pages themselves*
  - Organized in response to a query by relevance rankings or other scores

What about Ranking?

- Lots of variation here
  - Often messy; details proprietary and fluctuating

Combining subsets of:

- IR-style relevance: Based on term frequencies, proximities, position (e.g., in title), font, etc.
- Popularity information
- Link analysis 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

Relevance: Going Beyond IR

- Page “popularity” (e.g., DirectHit)
  - Frequently visited pages (in general)
  - Frequently visited pages as a result of a query

- Link “co-citation” (e.g., Google)
  - Which sites are linked to by other sites?
  - Draws upon sociology research on bibliographic citations to identify “authoritative sources”
  - Discussed further in Google case study
Web Search Architecture

Standard Web Search Engine Architecture

user query

crawl the web

check for duplicates, store the documents

create an inverted index

Search engine servers

Inverted index

DocIds

Inverted Indexes the IR Way
How Inverted Files Are Created

- Periodically rebuilt, static otherwise.
- Documents are parsed to extract tokens. These are saved with the Document ID.

Now is the time for all good men to come to the aid of their country

It was a dark and stormy night in the country manor. The time was past midnight

Multiple term entries for a single document are merged.
Within-document term frequency information is compiled.
How Inverted Files are Created

- Finally, the file can be split into
  - A Dictionary or Lexicon file
  - A Postings file

Inverted indexes

- Permit fast search for individual terms
- For each term, you get a list consisting of:
  - document ID
  - frequency of term in doc (optional)
  - position of term in doc  (optional)
- These lists can be used to solve Boolean queries:
  - country -> d1, d2
  - manor -> d2
  - country AND manor -> d2
- Also used for statistical ranking algorithms
Inverted Indexes for Web Search Engines

- Inverted indexes are still used, even though the web is so huge.
- Some systems partition the indexes across different machines. Each machine handles different parts of the data.
- Other systems duplicate the data across many machines; queries are distributed among the machines.
- Most do a combination of these.

In this example, the data for the pages is partitioned across machines. Additionally, each partition is allocated multiple machines to handle the queries.

Each row can handle 120 queries per second.
Each column can handle 7M pages.
To handle more queries, add another row.

From description of the FAST search engine, by Knut Risvik
http://www.infornetics.com/searchengine/00/h08/faith_files/frame.html

Cascading Allocation of CPUs

- A variation on this that produces a cost-savings:
  - Put high-quality/common pages on many machines
  - Put lower quality/less common pages on fewer machines
  - Query goes to high quality machines first
  - If no hits found there, go to other machines
Web Crawling

Web Crawlers

- How do the web search engines get all of the items they index?
- Main idea:
  - Start with known sites
  - Record information for these sites
  - Follow the links from each site
  - Record information found at new sites
  - Repeat

Web Crawling Algorithm

- More precisely:
  - Put a set of known sites on a queue
  - Repeat the following until the queue is empty:
    - Take the first page off of the queue
    - If this page has not yet been processed:
      - Record the information found on this page
      - Positions of words, links going out, etc
      - Add each link on the current page to the queue
    - Record that this page has been processed
- Rule-of-thumb: 1 doc per minute per crawling server
Web Crawling Issues

- Keep out signs
  - A file called norobots.txt lists “off-limits” directories
  - Freshness: Figure out which pages change often, and recrawl these often.
- Duplicates, virtual hosts, etc.
  - Convert page contents with a hash function
  - Compare new pages to the hash table
- Lots of problems
  - Server unavailable; incorrect html; missing links;
    attempts to “fool” search engine by giving crawler a version of the page with lots of spurious terms added ...
- Web crawling is difficult to do robustly!

Google: A Case Study

Google’s Indexing

- The Indexer converts each doc into a collection of “hit lists” and puts these into “barrels”, sorted by docID. It also creates a database of “links”.
  - Hit: <wordID, position in doc, font info, hit type>
  - Hit type: Plain or fancy.
  - Fancy hit: Occurs in URL, title, anchor text, metatag.
  - Optimized representation of hits (2 bytes each).
- Sorter sorts each barrel by wordID to create the inverted index. It also creates a lexicon file.
  - Lexicon: <wordID, offset into inverted index>
  - Lexicon is mostly cached in-memory
Google’s Inverted Index

Each “barrel” contains postings for a range of wordids.

Lexicon (in-memory)  Postings (“Inverted barrels”, on disk)

Sorted by wordid

Sorted by Docid

Barrel i

Barrel i+1

Google

- Sorted barrels = inverted index
- Pagerank computed from link structure; combined with IR rank
- IR rank depends on TF, type of “hit”, hit proximity, etc.
- Billion documents
- Hundred million queries a day
- AND queries

Link Analysis for Ranking Pages

- Assumption: If the pages pointing to this page are good, then this is also a good page.
  - References: Kleinberg '98, Page et al. '98
- Draws upon earlier research in sociology and bibliometrics.
  - Kleinberg’s model includes “authorities” (highly referenced pages) and “hubs” (pages containing good reference lists).
  - Google model is a version with no hubs, and is closely related to work on influence weights by Pinski-Narin (1976).
Link Analysis for Ranking Pages

Why does this work?
- The official Toyota site will be linked to by lots of other official (or high-quality) sites
- The best Toyota fan-club site probably also has many links pointing to it
- Less high-quality sites do not have as many high-quality sites linking to them

PageRank

Let $A_1, A_2, \ldots, A_n$ be the pages that point to page $A$. Let $C(P)$ be the # links out of page $P$. The PageRank (PR) of page $A$ is defined as:

$$PR(A) = (1-d) + d \left( \frac{PR(A_1)}{C(A_1)} + \ldots + \frac{PR(A_n)}{C(A_n)} \right)$$

PageRank is principal eigenvector of the link matrix of the web.
- Can be computed as the fixpoint of the above equation.

PageRank: User Model

PageRanks form a probability distribution over web pages: sum of all pages' ranks is one.
- User model: “Random surfer” selects a page, keeps clicking links (never “back”), until “bored”: then randomly selects another page and continues.
  - PageRank($A$) is the probability that such a user visits $A$
  - $d$ is the probability of getting bored at a page
- Google computes relevance of a page for a given search by first computing an IR relevance and then modifying that by taking into account PageRank for the top pages.
Web Search Statistics

<table>
<thead>
<tr>
<th>Service</th>
<th>Searches Per Day</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AltaVista</td>
<td>51 million</td>
<td>(as of July 1999)</td>
</tr>
<tr>
<td>Allport</td>
<td>47 billion</td>
<td>(estimated based on AltaVista's peak)</td>
</tr>
<tr>
<td>Google</td>
<td>40 million</td>
<td>90% (of these are all Google.com)</td>
</tr>
<tr>
<td>Excite</td>
<td>3 million</td>
<td>(as of October 1998)</td>
</tr>
<tr>
<td>Lycos</td>
<td>4 million</td>
<td>(as of October 1998)</td>
</tr>
<tr>
<td>Yahoo</td>
<td>7 million</td>
<td>(as of October 1998)</td>
</tr>
</tbody>
</table>

Web Search Engine Visits

<table>
<thead>
<tr>
<th>Country</th>
<th>Visits</th>
</tr>
</thead>
<tbody>
<tr>
<td>U.S.</td>
<td>89%</td>
</tr>
<tr>
<td>Japan</td>
<td>9%</td>
</tr>
<tr>
<td>Canada</td>
<td>0%</td>
</tr>
<tr>
<td>U.K.</td>
<td>0%</td>
</tr>
<tr>
<td>Germany</td>
<td>2%</td>
</tr>
</tbody>
</table>

*Note: The data is based on a single day's visits and may vary over time.*
The chart below shows audience reach over time. Data for some of the less popular services is not available each month, which is why there are gaps on the chart. An analysis of changes in the past five months is shown below the chart.

Percentage of web users who visit the site shown.

The table below shows some search engines, ranked by size as of July 2000. Size is defined as the number of pages indexed. The first column shows search engines, the second shows the size in millions of indexed pages, the third shows the size of the site, and the fourth shows the ranking.

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Required Size</th>
<th>Expected Size</th>
<th>Actual Size</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>AltaVista</td>
<td>248</td>
<td>10</td>
<td>1.1</td>
<td>1.5</td>
</tr>
<tr>
<td>Yahoo</td>
<td>110</td>
<td>40</td>
<td>2.1</td>
<td>2</td>
</tr>
<tr>
<td>Earth</td>
<td>110</td>
<td>40</td>
<td>2.1</td>
<td>2</td>
</tr>
<tr>
<td>ArcNet</td>
<td>100</td>
<td>20</td>
<td>2.1</td>
<td>4</td>
</tr>
<tr>
<td>Zip2</td>
<td>100</td>
<td>10</td>
<td>3.1</td>
<td>5</td>
</tr>
<tr>
<td>Stamps</td>
<td>200</td>
<td>20</td>
<td>2.1</td>
<td>5</td>
</tr>
<tr>
<td>YellowPages</td>
<td>110</td>
<td>40</td>
<td>4.1</td>
<td>6</td>
</tr>
</tbody>
</table>

The first column shows you how many millions of pages each search engine claims to have indexed. The "Expected Size" column is based on the size of the site as of June 2000. The "Rank" column is based on the number of hits per month.

**Obscure Terms**

The first column in the table shows search engines, the second shows the size in millions of indexed pages, the third shows the size of the site, and the fourth shows the ranking.

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Required Size</th>
<th>Expected Size</th>
<th>Actual Size</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>AltaVista</td>
<td>248</td>
<td>10</td>
<td>1.1</td>
<td>1.5</td>
</tr>
<tr>
<td>Yahoo</td>
<td>110</td>
<td>40</td>
<td>2.1</td>
<td>2</td>
</tr>
<tr>
<td>Earth</td>
<td>110</td>
<td>40</td>
<td>2.1</td>
<td>2</td>
</tr>
<tr>
<td>AarcNet</td>
<td>100</td>
<td>20</td>
<td>2.1</td>
<td>4</td>
</tr>
<tr>
<td>Zip2</td>
<td>100</td>
<td>10</td>
<td>3.1</td>
<td>5</td>
</tr>
<tr>
<td>Stamps</td>
<td>200</td>
<td>20</td>
<td>2.1</td>
<td>5</td>
</tr>
<tr>
<td>YellowPages</td>
<td>110</td>
<td>40</td>
<td>4.1</td>
<td>6</td>
</tr>
</tbody>
</table>

The first column shows you how many millions of pages each search engine claims to have indexed. The "Expected Size" column is based on the size of the site as of June 2000. The "Rank" column is based on the number of hits per month.

**Does size matter? You can’t access many hits anyhow.**

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Max. Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>AltaVista</td>
<td>24800</td>
</tr>
<tr>
<td>Yahoo</td>
<td>1000</td>
</tr>
<tr>
<td>Earth</td>
<td>1000</td>
</tr>
<tr>
<td>AarcNet</td>
<td>2000</td>
</tr>
<tr>
<td>Zip2</td>
<td>1000</td>
</tr>
<tr>
<td>Stamps</td>
<td>1000</td>
</tr>
<tr>
<td>YellowPages</td>
<td>1000</td>
</tr>
<tr>
<td>MapQuest</td>
<td>1000</td>
</tr>
<tr>
<td>GeoCities</td>
<td>24800</td>
</tr>
</tbody>
</table>

For the curious, here’s a list of the maximum number of results you can possibly access from the search engines noted.
Increasing numbers of indexed pages, self-reported

Coverage Of The Web

NOTE: The chart assumes none of these numbers will go below 1 in the future. The chart shows that the percentage of the web currently covered by each search engine did not exceed 1%.

Size Growth

From description of the FAST search engine, by Knut Risvik
http://www.infonortics.com/searchengines/sh00/risvik_files/frame.htm
### Directory Sizes

Directories are small human-readable indexes to the web; their sizes are organized by category. The chart below compares the size of directories at various services, along with their byte size. A Total indicator refers to information at a service or search front end.

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Byte Size</th>
<th>Notes</th>
<th>Total</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>0.2MB</td>
<td>200MB</td>
<td>miXed</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Index</td>
<td>0.1MB</td>
<td>100MB</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>SSL/Sign</td>
<td>0.1MB</td>
<td>100MB</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>NCH/Check</td>
<td>0.1MB</td>
<td>100MB</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Address</td>
<td>0.1MB</td>
<td>100MB</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Cache</td>
<td>0.1MB</td>
<td>100MB</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Notes</td>
<td>0.1MB</td>
<td>100MB</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FileSystem</td>
<td>0.1MB</td>
<td>100MB</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Apache</td>
<td>0.1MB</td>
<td>100MB</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>