Fundamentals of Web Searching

Disclaimer: These abbreviated notes DO NOT substitute the textbook for this class. They should be used IN CONJUNCTION with the textbook and the material presented in class. If there is a discrepancy between these notes and the textbook, ALWAYS consider the textbook to be correct. Report such a discrepancy to the instructor so that he resolves it. These notes are only distributed to the students taking this class with A. Gerbessiotis in Spring 2006 ; distribution outside this group of students is NOT allowed.

(Based on Chapters 1 and 13 of BYRN)

Overview

- **Information retrieval** (IR) deals with the representation, storage, organization of, and access to information items.
- **Web-searching** can be realized by a search engine which is an IR sytem and deals also with locating and retrieving the relevant information if it is not stored locally and ranking results of the retrieval process. Also web-searching is syntactic search i.e. we search user-specified words or patterns in the text of document; we do not do a natural language analysis of the text to say extract the text semantics. A difference beween standard IR systems and the Web is that in the web all queries must be answered without accessing the text (only indices) since otherwise we would need either local copies of all the Web or a too slow response time to query all the Web.
- Information vs Data retrieval. Differences?
- Detail and precision.
- Databases are for data retrieval. Data are organized, have structure, and semantics.
- Until 1995 IR was library-related
- Nowadays the only thing we deal with is TEXT Retrieval.
- Future/Other directions involve MULTIMEDIA (photos, video, sound).
- How does WWW affect it?
  - Locate useful information. Is it linked? Where is it?
  - Is it at www.nice-site.edu or at 138213.nice-site.edu?
  - WWW is disorganized. No data model.
- Our approach: Not human-centric (e.g. IS approach) trying to understand how people use and interpret information, but computation centric (e.g. CS) on how to structure, store, retrieve and rank relevant information.

Retrieval vs Browsing. Document Logical View.

- A user of a retrieval system formulates an initially imprecise natural language specification into a more structured and less imprecise query.
- The query might or might not be equivalent to the originally defined space of relevant answers.
- A retrieval task is then executed.
- Many times a browsing task is performed that is interchanged with the retrieval task e.g. a hit or an answer of the retrieval task, is further explored (eg. an html document).

What is a document? Definition

- A **document** is a single unit of information in digital form.
- Documents are usually represented through a set of index terms or **keywords**.
- Keywords can be automatically extracted from the document or be user-specified (eg. an expert decides them). The keywords form the **logical view** of a given document.
- **Full-text** logical view is one in which all the words of a document become keywords.
- Because of space problems, an **index-term** logical view is more preferable. This can be accomplished by the use of **text operations** that
  - eliminate **stopwords** such as articles and connectives,
  - use **stemming** to reduce distinct words to their common grammatical root,
  - identify **noun groups**, and
  - identify **similar (in meaning) words** called **synonyms**.
- To accommodate such a view in a way that is viable, **compression** might also be used.

Common StopWords

```
 I a about an are as at be by com en for from
how in is it  of on or that the this to was what
when where who will with www
```

- Does google use stopwords? Does google index stopwords? Can you search for "the"?
- How about other search engines?

Challenges

- Difficulty to get the information one wants! (Retrieval quality should be improved).
- Faster indexing, quicker response times?
- User interaction.

0. Preprocessing: Define Text Database
    - Type of documents to be used (eg. html,txt,zip,pdf,ps,tex)
    - Operations to be performed on text
    - Text model (i.e. text structure, what elements can be retrieved).
1. Grab the documents.
2. Index the documents using the logical view of each one of them. [Build inverted and forwards indexes, lexicons of documents and words, repository of actual documents in compressed format].
3. Use information of steps 1 and 2 to build ranking mechanisms for the data.
4. Identify and make available query possibilities for users and define text operations to facilitate them.
5. A *query need* is translated into a *query* on which *query operations* are applied to generate a list of *retrieved documents* which are listed after a possible *ranking function* is applied to them. Such a ranking mechanism can combine general information already available (step 3 e.g. Google's PageRank) and information generated by the query (step 5 e.g. ranking of query results per document).

An example of a Retrieval Process: Google

```
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.
8. Sort the documents that have matched by rank and return the top k.

Figure 4 (Google Query Evaluation) from http://www-db.stanford.edu/~backrub/google.html
```

What is Web searching?

- Web-searching can be realized in three forms

1— Use search engines to index a portion of the Web-documents into a full-text database.

2— Use Web-directories which classify a subset of Web documents by subject.

3— Search the Web by exploiting the hyperlink structure of it.

Challenges of Web Searching

- Modeling.
- Querying. User Interfaces.
- Distributed Data/Architectures. Data spans over many computers and platforms. Web addresses of web servers not informative. Network reliability and topology unpredictable and varying.
- High percentage of volatile data. Internet is a dynamic entity; things get updated often, links become dangling, data get destroyed.
- Quality of data. (false, typos, out-of-date. Errors from 1 in 200 to 1 in 3).
- Large Volume!
- Ranking.
- Dynamic Pages.
- Indexing. What should be indexed?
- Unstructured and Redundant/Duplicate Data. The Web is NOT a distributed hypertext; hypertext assumes the existence of a conceptual model behind it which organizes and adds consistency to the data and hyperlinks. Approximately 30
- Multimedia
- Heterogeneous data (not only in document types but also languages).
- Browsing. Further unify searching with browsing.

TABLE 1: Search Engines.

-1- = Search hours per month (in millions)
-2- = Search minutes per day (in millions)
-3- = Searches per day (in millions)

Engine	-1-	-2-	-3-
Google	18.7	37	112
AOL Search	15.5	31	93
Yahoo	7.1	14	42
MSN Search	5.4	11	32
Ask Jeeves	2.3	5	14
InfoSpace	1.1	2	7
.... others			
TOTALS	53.2	106	319

Source: SearchEngineWatch.com, Feb. 25, 2003

TABLE 2: Web Size

Size of Web	~32,000,000	Based on domain name registration
	~42,800,000	Web servers responding to HTTP requests
	~ 9,000,000	IP addresses responding to HTTP request (each IP can maintain many virtual domain names).


```
TABLE 3: Web statistics
-----------------------

Average page size  18.7Kbytes
Number of links per page       > 8
Surface Web 167Tbytes
Deep    Web                   91850Tbytes
Email      440606Tbytes
Instant messaging               274Tbytes

----------------------------------------------------------------------------
TABLE 4: Web File formats
-------------------------

 IMAGES                 23%
 HTML                   18%
 PHP                    13%
 PDF                     9%
 Movies                  4%
 Compressed              4%
 Executables           1.4%
 Powerpoint            0.8%
 Word                  0.4%
 Text                  0.1%
 Java                  0.1%

All data in Tables 1-4 from
  http://www.sims.berkeley.edu/research/projects/how-much-info-2003/internet.htm
```

Measuring the Web according to Google

```
Google's Crawler @ Stanford [circa 2000]
  Active crawlers       :  3-4 typically
  Open Connections      : 300
  Web-page read         : 100 concurrently
  Data                  : 600Kbytes/second

Google @ Stanford size statistics (in millions)
  # of Web pages        : 24
  #  of URLs            : 76
  #  of Email addresses : 1.7
  # of 404messgs        : 1.6

Google Index Data
  Fetched Information   : 147Gbytes
  Compressed            :  53Gbytes (3 to 1)
  Inverted Index Full   :  37Gbytes
  Lexicon               :  .3Gbytes
  DocIndex              : 9.7Gbytes
  links                 : 3.9Gbytes

   All data from Brin & Page [ http://www-db.stanford.edu/~backrub/google.html ]
```


Web document size

How can we approximate the distribution of Web-based document sizes?

Size distribution changes between text and image documents and depends on the document type.

Size distribution might follow a Pareto distribution with $p(x) = ak^a/x^{1+a}$, where x is measured in bytes and k, a are parameters of the distribution.

For text files $a = 1.36$ and even smaller for images and other binary formats.

For ALL documents types $a = 1.1$ and $k = 9.3Kb$.

Parameter a might change slowly with time; parameter k might grow signficantly if say, video/audio files become more frequent.

Search Engines

```
       |-----------|             |--------------|
       |           |             |              |
  User-|Interface  |<----------->| Query Engine |<-----> Index
       |-----------|             |--------------|          ^
                                                           |
                                                           |
                                                       Indexer
             Crawler -------------------------------------^
                |
                |
               Web

   A Centralized Architecture (eg. Altavista)
```

Crawler: Collects information by following hyperlinks.
Indexer : Creates data structure for fast searching of documents based on the indexed information.
Query Engine: Interacts with query interface (and thus the user) and also the index to find hits, and rank the results and also provide summary information.
Interface : User entry point to the system.

Crawlers

Crawlers are programs (eg. software agents) that traverse the Web in a methodical automated manner sending new or updated pages to a repository for post processing. A web crawlers traverses the web according to a set of policies that include a (a) selection policy (which pages to visit eg. only html), (b) a visit policy, (c) an observance, and (d) a parallelization/coordination policy.

Crawlers are also referred to as robots or just bots, or spiders. Techniques for crawling the Web or visit policies are variants of the two fundamental graph search operations : breadth-first search (BFS) and depth-firsth search (DFS). Between the two the former is the prevalent one.

- A simple way to crawl the web is to give the crawler program a list of URLs to visit and to BFS/DFS from these.
- A variant is to start with the most popular URL instead of using an arbitrary set.
- Another way is to initiate a visit based on exhaustive search of URLs or more formally on IP addresses. However this is tedious, and among almost 4 billion IP addresses, few of them correspond to Web-servers.

In case there are more than one crawlers active at any moment their activity needs to be synchronized.

- A central URLserver might send groups of URLs to individual crawlers (eg. the Google approach).
- The space of 4 billion potential IP addresses is cleverly partitioned based say on numeric properties, country codes etc.

Crawlers (continued)

Other issues that can affect crawling are

- URLs are assigned priorities that decide which URL are visited more often and how often than other ones. URLs that change less often than others do not have to be visited very often.
- Nowaday crawlers visit almost 10,000,000 pages a day.
- Web-server load. Guidelines for robot behavior (robots.txt). Password protection. These determine the observance policies.

Example of Crawlers include RBSE (Eichmann, 1994), WebCrawler (Pinkerton, 1994) , WebSPHINX (Miller and Bharat, 1998) written in Java (more on PS1), the Google Crawler (Brin and Page, 1998), CobWeb (da Silva et al, 1999) written in Perl, Mercator (Heydon and Najork, 1999) written in Java, Webfountain (Edwards et al., 2001) written in C++, PolyBOT (Shlapenyuk and Suel, 2002), WebRACE (Zeinalipour-Yazti and Dikaiakos, 2002), Ubicrawler (Boldi et al., 2004), FAST Crawler (Risvik and Michelsen, 2002), WIRE (Baeza-Yates and Castillo, 2002) written in C++.

Digimark searches only images (with watermarks).

A timeline

```
  ?? Dewey System
1950 IR
1963 IR/ G. Salton Work / Vector Space model
1971 SMART system
198x Lexis Nexis other similar systems
1990 Archie
1991 Gopher
1992 Veronica
1993 Wonderer (first crawler)
1994 Lycos
1994 Yahoo directory
1995 Meta Crawler (mete search engine)
1998 Google
200? Era of Google ?
```

Searching Issues

- Scalability of design
- file types
- Query language
- Keyword/Phrase search
- Nearness of phrase keywords
- stopwords/stemming
- synonyms
- language issues
- subversion/spamming
- Indexing.
  - Disk access time: around 5 miliseconds.
  - CPU/instruction time: 4Ghz @ 2-4 instructions per cycle.
  - Index stored in a DB vs standard file system (UNIX) vs specialized system (eg. Google's GFS)
- Ranking

