WEB SEARCHING AND INFORMATION RETRIEVAL

The first Web information services were based on traditional information retrieval algorithms, which were originally developed for smaller, more coherent collections than the Web. Due to the Web's continued growth, today's Web searches require new techniques—exploiting or extending linkages among Web pages, for example.

Although we can think of the Web as a huge semistructured database that provides us with a vast amount of information, no one knows exactly how many Web pages are out there. Google reports more than 3.3 billion textual documents indexed up to September 2003, but that same month had at least 5.2 billion documents with the word "the" in their Google listings (www.webmasterworld.com/forum3/16779.htm). We can assume that many additional documents and Web pages—perhaps in other languages—do not contain the word "the."

Most people believe they can easily find the information they're looking for on the Web. They simply browse from the prelisted entry points in hierarchical directories (like yahoo.com) or start with a list of keywords in a search engine. However, many Web information services deliver inconsistent, inaccurate, incomplete, and often irrelevant results.

For many reasons, existing Web search techniques have significant deficiencies with respect to robustness, flexibility, and precision. For example, although general search engines crawl and index thousands of Web pages (the so-called *surface Web*), they typically ignore valuable pages that require authorization or prior registration—the ones whose contents are not directly available for crawling through links. This is the *hidden* (or *deep* or *invisible*) Web. Public information on the hidden Web is currently estimated to be 400 to 550 times larger than the surface Web.[1]

Another unpleasant feature of the Web is its volatility. Web documents typically undergo two kinds of change. The first—*persistence*—is the existence or disappearance of Web pages and sites during a Web document's life cycle. According to one study,[2] a Web page's "half-life" seems to be somewhat less than two years, with a Web site's half-life being somewhat more than two years. The second type of change is page or site content modification. Another study[3] notes that 23 percent of of all Web pages change daily (40 percent of commercial Web pages change daily); it also reports a half-life of 10 days for the commercial Web pages. Some pages disappear completely, though, which means the data gathered by a search engine can quickly become stale or out of date. Crawlers must regularly revisit Web pages to maintain the freshness of the search engine's data.

The first Web information services were based on traditional information retrieval (IR) algorithms and techniques (a critical summary and review appears elsewhere[4]). However, most IR algorithms were de-

1521-9615/04/$20.00 © 2004 IEEE
Copublished by the IEEE CS and the AIP

JAROSLAV POKORNÝ
Charles University

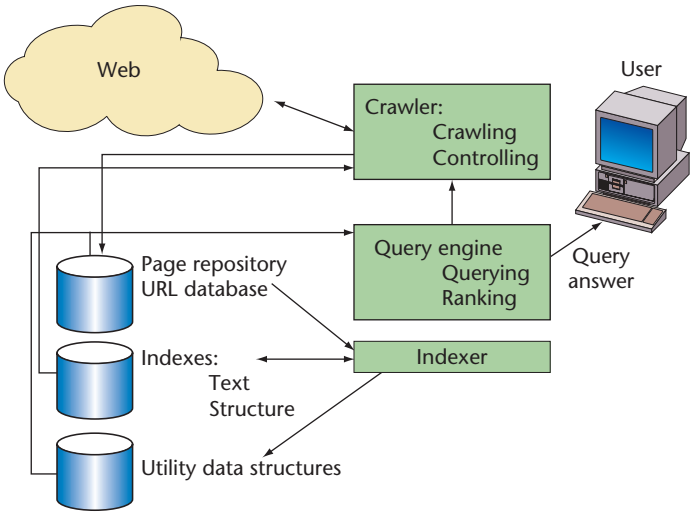

Figure 1. Architecture of a search engine. The modules are the crawler, query engine, and indexer; the data includes the page repository, URL databases, indexes, and utility data structures.

veloped for smaller, more coherent collections than what the Web has become: today's Web searching requires new techniques. This article offers an overview of current search-engine architectures and techniques in the context of IR and discusses some of the difficult problems in maintaining or enhancing search-engine performance quality.

Search-Engine Architectures

We can distinguish three architectures for Web searching: traditional (or centralized), metasearch, and distributed search. Search engines can also be part of the more general architectures such as search services or portals.

Centralized Architecture

The goal of general-purpose search engines is to index a sizeable portion of the Web, independently of topic and domain. Each such engine consists of several components, as Figure 1 shows.

A *crawler* (also called a *spider* or *robot*) is a program controlled by a crawl control module that "browses" the Web. It collects documents by recursively fetching links from a set of start pages; the retrieved pages or their parts are then compressed and stored in a *page repository*. URLs and their links, which form a Web graph, are transferred to the crawler control module, which decides the movement in this graph. Obviously, off-site links are of interest. To save space, documents' identifiers (docIDs) represent pages in the index and other data structures; the crawler uses a database of URLs for this purpose.

The *indexer* processes the pages collected by the crawler. It first decides which pages to index—for example, it might discard duplicate documents. Then, it builds various auxiliary data structures. Most search engines build some variant of an inverted index data structure for words (*text index*) and links (*structure index*). The inverted index contains for each word a sorted list of couples (such as docID and position in the document).

The *query engine* processes user queries—and returns matching answers—in an order determined by a ranking algorithm. The algorithm produces a numerical score expressing an importance of the answer with respect to the query. Its capabilities and features depend on additional data structures (called *utility* data structures) such as lists of related pages, similarity indexes, and so forth. The numerical score is usually a combination of query-independent and query-dependent criteria. The former judge the document regardless of the actual query; typical examples include its length and vocabulary, publication data (such as the site to which it belongs, the date of the last change to it, and so on), and various connectivity-based data such as the number of links pointing to a page (called *in-degree*). Query-dependent criteria include a cosine measure for similarity in the vector space model (which is well known from traditional IR techniques) and all connectivity-based techniques. All defined measures can contribute to the resulted measure.

Metasearch Architecture

One way to provide access to the information in the hidden Web's text databases is through *metasearchers*, which can be used to query multiple databases simultaneously. A metasearcher performs three main tasks. After receiving a query, it finds the best databases to evaluate the query (*database selection*), translates the query in a suitable form for each database (*query translation*), and then retrieves and merges the results from the different databases (*result merging*) and returns them to the user. A metasearcher's database selection component is crucial in terms of both query processing efficiency and effectiveness.

Database selection algorithms are traditionally based on pre-collected statistics that characterize each database's contents. These statistics, often called *content summaries*, usually include at least the *document frequencies* of the words that appear in the database.

To obtain a database's content summary, a metasearcher relies on the database to supply the summary (for example, by using Semantic Web tags). Unfortunately, many Web-accessible text

databases are completely autonomous and don't report any detailed metadata about their contents that would facilitate metasearching. With such databases, only manually generated descriptions of the contents are usable, so this approach is not scalable to the thousands of text databases available on the Web today. Moreover, we wouldn't get the good-quality, fine-grained content summaries required by database selection algorithms. Some researchers recently presented a technique to automate content-summary extraction from searchable text databases:[5] it seems that the deeper recesses of the Web aren't really hidden. By systematically retrieving small sample contents, we can model information sources.

Distributed Search Architecture

Whatever successful global ranking algorithms for centralized search engines are, two potential problems occur: high computational costs and potentially poor rankings. Additional semantic problems are related to the exclusive use of global context and the instability of ranking algorithms.

Distributed heterogeneous search environments are an emerging phenomenon in Web search. Although the original Internet was designed to be a peer-to-peer (P2P) system, Web search engines have yet to make full use of this potential. Most major Web search engines are currently based on cluster architectures.

Earlier attempts to distribute processes suffered many problems—for example, Web servers got requests from different search-engine crawlers that increased the servers' load. Most of the objects the crawlers retrieved were useless and subsequently discarded; compounding this, there was no coordination among the crawlers. Fortunately, this bleak picture has improved: a new completely distributed and decentralized P2P crawler called Apoidea is both self-managing and uses the resource's geographical proximity to its peers for a better and faster crawl.[6] Another recent work[7] explores the possibility of using document rankings in searches. By partitioning and combining the rankings, the decentralized crawler manages to compute document rankings of large-scale Web data sets in a localized fashion.

The most general approach is a *federation* of independently controlled metasearchers along with many specialized search engines. These engines provide focused search services in a specific domain (for example, in a particular topic).

Page Importance and Its Use in Retrieval

In general, we must measure a page's importance in order to rank it. Three approaches help with this process: *link*, *content* (*similarity*), and *anchor*. In terms of IR, these measures reflect a *model* of Web documents.

The best-known link-based technique used on the Web today is a variant of the PageRank algorithm[8] implemented in the Google search engine. It tries to infer a Web page's importance from just the topological structure of a directed graph associated with the Web.

A page's rank depends on the ranks of all the pages pointing to it, with each rank divided by the number of out-links those pages have. In the most simple variant, the PageRank of a page k, $P_r(k)$ is a nonnegative real number given by

$$P_r(k) = \Sigma_{(b,k)} P_r(b)/o(b), \qquad k = 1, 2, \ldots, n,$$

where $P_r(b)$ is the PageRank of page b, $o(b)$ is the out-degree of page b, and the sum is extended to all Web pages b pointing to page k (n is the number of pages on the Web). If a page b has more out-links to the same page k, all these out-links count as one. According to this definition, then, $P_r(b)$ depends not only on the number of pages pointing to it, but also on their importance. This definition raises some problems—something like a *rank sink* can occur (a group of pages pointing to each other could have some links going to the group but no links going out).

Another interesting technique—Kleinberg's algorithm,[9] also called HITS (Hypertext Induced Topic Search)—is used at query time and processed on a small subset of relevant documents, but not all of them. It computes two scores per document. *Authoritative pages* relevant to the initial query have large in-degree: they are all the authorities on a common topic, and a considerable overlap in the sets of pages point to them. The algorithm then finds *hub pages*, which have links to multiple relevant authoritative pages: if a page were a good authority, many hubs would point to it. These ideas are not new. Some were exploited decades ago in bibliographic citation analysis and later in the field of hypertext systems.

In the content-based approach, we compute the similarity score between a page and a predefined topic in a way similar to the vector model. Topic vector q is constructed from a sample of pages, and each Web page has its own vector p. The similarity score $\mathrm{Sim}(p, q)$ is defined by the cosine similarity measure.

Anchor text is the visible hyperlinked text on the Web page. In the anchor-based approach, page quality can be judged by pattern matching between the query vector and the URL's anchor text, the text

around the anchor text (the anchor window), and the URL's string value.

Approaches used in isolation suffer various drawbacks. The usual content-based approach ignores links and is susceptible to spam, and the link-based approach is not adequate for pages with low in-degree. Due to the Web's dynamism, this problem appears most frequently when we attempt to discover new pages that have not been cited sufficiently. The approach relying on text near anchors seems to be the most useful for Web similarity-search tasks.[10] Similar to vector models, it must involve additional considerations concerning term weighting and anchor window width. With small anchor windows, for example, many documents that should be considered similar are in fact orthogonal (they don't have common words).

Obviously, all previously defined measures can contribute to the end page measure result for page ranking.

Issues and Challenges in Web Search Engines

search-engine problems are connected with each component of the engine's architecture and each process it performs—search engines can't update indexes at the same speed at which the Web evolves, for example. Another problem is the quality of the search results. We've already looked at their lack of stability, heterogeneity, high linking, and duplication (near 30 percent). On the other hand, because the hidden Web's contents' quality is estimated to be 1,000 to 2,000 times greater than that of the surface Web, search result quality can be expected to be higher in this case.

One of the core modules of each search engine is its crawler. Several issues arise when search engines crawl through Web pages:[3]

- *What pages should the crawler download?* Page importance metrics can help, such as interest-driven metrics (often used in focused crawlers), popularity-driven metrics (found in combination with algorithms such as PageRank), and location-driven metrics (based on URL).
- *How should the search engine refresh pages, and how often should it do so?* Most search engines update on a monthly basis, which means the Web graph structure obtained is always incomplete, and the global ranking computation is less accurate. In a *uniform refresh*, the crawler revisits all pages with the same frequency, regardless of how often they change. In a *proportional refresh*, the crawler revisits pages with a frequency proportional to the page's change rate (for example, if it changes more often, it visits more often).
- *How do we minimize the load on visited Web sites?* Collecting pages consumes resources (disks, CPU cycles, and so on), so the crawler should minimize its impact on these resources. Most Web users cite load time as the Web's single biggest problem.
- *How should the search engine parallelize the crawling process?* Suppose a search engine uses several crawlers at the same time (in parallel). How can we make sure they aren't duplicating their work?

A recent research study highlighted several problems concerning the quality of page ranking:[11]

- *Spam*. To achieve a better ranking, some Web authors deliberately try to manipulate their placement in the ranking order. The resulting pages are forms of spam. In text spam, erroneous or unrelated keywords are repeated in the document. Link spam is a collection of links that point to every other page on the site. *Cloaking* offers entirely different content to a crawler than to other users.
- *Content quality*. There are many examples of Web pages containing contradictory information, which means the document's accuracy and reliability are not automatically guaranteed. If we calculate page importance from the anchor text, for example, we would want at least this text to be of high quality (meaning accurate and reliable).
- *Quality evaluation*. Direct feedback from users is not reliable because such user environment capabilities are usually not at our disposal. So, search engines often collect implicit user feedback from log data. New metrics for ranking improvement, such as the number of clicks, are under development.
- *Web conventions*. Web pages are subject to certain conventions such as anchor text descriptiveness, fixed semantics for some link types, metatags for HTML metadata presentation, and so on. Search engines can use such conventions to improve search results.
- *HTML mark-up*. Web pages in HTML contain limited semantic information hidden in HTML mark-up. The research community is still working on streamlined approaches for extracting this information (an introductory approach appears elsewhere[12]).

Most search engines perform their tasks by using important keywords, but the user might not always know these keywords. Moreover, the user might want to submit a query with additional constraints such as searching a specific Web page or finding the pages within a Web graph structure.

Toward the Semantic Web

The idea behind the Semantic Web is to augment Web pages with mark-up that captures some of the meaning of the content on those pages (www.w3.org/2001/sw/). Automatic tools can collect and "understand" the knowledge annotated on a page, and ontologies help make such mark-up compatible across various information sources and queries. An *ontology* is an explicit specification of a vocabulary for a domain, and it includes definitions of classes, relations, functions, and constraints. Because the range and diversity of data on the Web is too extensive, most ontologies are domain-specific or personalized to express the specific interests of individuals or communities.

The Semantic Web is an extension of the current Web: it offers Web page documents as well as the relationships among resources denoting real-world objects. Some Web pages might contain semantic mark-up information, but today's crawlers do not use it yet.

An advantage of machine-readable metadata such as semantic mark-up is that the search engines can use it to infer additional semantic relations; then, they can apply a so-called *semantic search*. Semantic search aims to extend and improve traditional search processes based on IR technology.

A new generation of intelligent search engines incorporates Web semantics and uses more advanced search techniques based on concepts such as machine learning. These approaches enable intelligent Web information services, personalized Web sites, and semantically empowered search engines. Figure 2 shows a possible mediated architecture of Web searching with ontologies.

An additional branch of Web searching uses the *XML Web*: a subset of the Web containing XML documents only. A recent paper[13] reports the first results of an analysis of roughly 200,000 XML documents publicly available on the Web.[14] Searching such data can take into account mark-up of XML documents and the mark-up's structure.[15] Other approaches use keyword processing,[16] a form of approximate querying,[6] contexts and weights (as is usual in IR[17,18]), or a text similarity.[19] Unfortunately, most of these techniques are now used in XML native databases, such as XYZfind,[20] rather than in a full Web context.

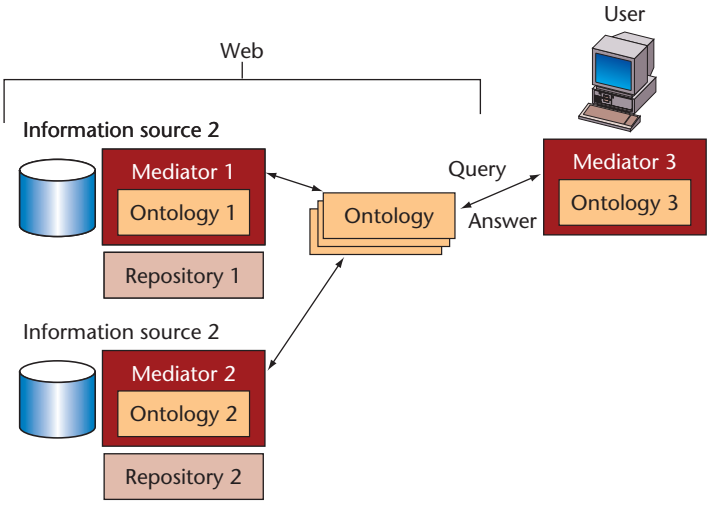

Figure 2. Architecture of a mediated Web search. The user formulates a query in Ontology 3; a mediator then transforms this query to queries based on other ontologies.

Many people think of the Web as a (digital) library. In his well-known test to prove the contrary,[21] José-Marie Griffiths pointed out that

- the Web does not offer access to all information,
- the Web lacks authority and quality control,
- the Web is inadequately catalogued, and
- Web search interfaces and other tools are ineffective and simplistic.

In other words, searching is not enough. Although the Web is not a digital library from a librarian's viewpoint, material for library collections can be found on the Web. Steps toward a Semantic Web are also steps toward intelligent searching: information and annotated information sources and their content support the vision of the next-generation Web as a digital library. Let's not forget an important truth: the only way intelligence gets into a computer is as a result of humans putting it there.

Acknowledgments

Grant GACR 201/03/0912 and project RC-3-42 in the Greece–Czech Cooperation Program partially supported this research.

References

1. M.K. Bergman, "The Deep Web: Surfacing Hidden Value," *J. Electronic Publishing*, vol. 7, no. 1, 2001, p. 6830.
2. W. Koehler, "Digital Libraries and World Wide Web Sites and Page Persistence," *Information Research*, vol. 4, no. 4, 1999, http://informationr.net/ir/4-4/paper60.html.

3. A. Arasu et al., "Searching the Web," *ACM Trans. Internet Technology*, vol. 1, no. 1, 2001, pp. 2–43.
4. M. Agosti and M. Melucci, "Information Retrieval on the Web," *Lectures on Information Retrieval: Third European Summer School* (ESSIR 2000), M. Agosti, F. Crestani, and G. Pasi, eds., Springer, 2001, pp. 242–285.
5. P.G. Ipeirotis and L. Gravano, "Distributed Search over the Hidden Web: Hierarchical Database Sampling and Selection," *Proc. Conf. Very Large Databases*, Morgan Kaufmann, 2002, pp. 394–405.
6. A. Singh et al., "Apoidea: A Decentralized Peer-to-Peer Architecture for Crawling the World Wide Web," *Proc. SIGIR 2003 Workshop on Distributed Information Retrieval*, Springer Verlag, 2003, pp. 126–142.
7. K. Aberer and J. Wu, "A Framework for Decentralized Ranking in Web Information Retrieval," *Proc. Asia Pacific Web Conf.*, Springer Verlag, 2003, pp. 213–226.
8. L. Page et al., "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, vol. 30, nos. 1–7, 1998, pp. 107–117.
9. J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *J. ACM*, vol. 46, no. 5, 1999, pp. 604–632.
10. T.H. Haveliwala et al., "Evaluating Strategies for Similarity Search on the Web," *Proc. WWW*, ACM Press, 2002, pp. 432–442.
11. M.R. Henzinger, R. Motwani, and C. Silverstein, "Challenges in Web Search Engines," *Proc. ACM Special Interest Group on Information Retrieval Forum*, ACM Press, 2002, pp. 11–22.
12. S. Chakrabarti, "Enhanced Topic Distillation Using Text, Markup Tags, and Hyperlinks," *Proc. ACM SIGIR Conf. on Research and Development on IR*, ACM Press, 2001, pp. 208–216.
13. L. Mignet, L. Barbosa, and V. Pierangelo, "The XML Web: A First Study," *Proc. WWW*, ACM Press, 2003, pp. 500–510.
14. L. Xyleme, "A Dynamic Warehouse for XML Data of the Web," *IEEE Data Eng. Bulletin*, vol. 24, no. 2, 2001, pp. 40–47.
15. R. Luk et al., "A Survey of Search Engines for XML Documents," *Proc. ACM SIGIR 2000 Workshop on XML and Information Retrieval*, ACM Press, 2000, pp. 1–9.
16. D. Florescu, D. Kossmann, and I. Manolescu, "Integrating Keyword Search into XML Query Processing," *Computer Networks*, vol. 33, nos. 1–6, 2000, pp. 119–135.
17. N. Fuhr and K. Grobjohann, "XIRQL: An Extension of XQL for Information Retrieval," *Proc. ACM SIGIR 2000 Workshop on XML and Information Retrieval*, ACM Press, 2000, pp. 172–180.
18. A. Theobald and G. Weikum, "Adding Relevance to XML," *Proc. WebDB 2000*, ACM Press, 2000, pp. 105–124.
19. T. Chinenyanga and N. Kushmerick, "Expressive Retrieval from XML Documents," *Proc. ACM 24th Int'l Conf. Research and Development in Information Retrieval* (SIGIR'01), ACM Press, 2001, pp. 163–171.
20. D. Egnor and R. Lord, "XYZfind: Searching in Context with XML," *Proc. ACM SIGIR 2000 Workshop on XML and Information Retrieval*, ACM Press, 2000, pp. 69–78.
21. J.-M. Griffiths, "Why the Web Is Not a Library," *The Mirage of Continuity: Reconfiguring Academic Information Resources for the Twenty-First Century*, B.L. Hawkins and P. Battin, eds., Council on Library and Information Resources, 1998.

Jaroslav Pokorný is a full professor of computer science at Charles University in Prague and the head of its Department of Software Engineering. His research interests include database systems, text information systems, and XML. He has published more than 200 papers and four books. Pokorný is a member of the ACM and the IEEE. Contact him at pokorny@ksi.ms.mff.cuni.cz.