Internet Searching

Peter Norvig, Google Inc.

In 300 B.C., the Library of Alexandria held some half million scrolls and served as a cultural center for the world, uniting the thoughts of Greek, Egyptian, Macedonian and Babylonian culture. According to the *Letter of Aristeas*, the library had "a large budget in order to collect, if possible, all the books in the world." Whole fields of study such as geometry and grammar grew from the scholars who had the fortune to travel to and study at Alexandria.

In 2003 A.D., the Internet forms a distributed library of billions of pages, one that is accessible to anyone, anywhere in the world, at the click of a mouse. Every day hundreds of millions of "trips" to the library start with a query to an Internet search engine. If I want to research a topic such as "Library of Alexandria," I can type those words to a search engine and in less than a second have access to a list of 14,000 pages, sorted by their usefulness to the query. This unprecedented ease of access to information has revolutionized the way research is done by students, scientists, journalists, shoppers, and others. It opens up an online marketplace of products, services, and ideas that benefits both information providers and seekers; sellers and buyers; consumers and advertisers.

How is it that an Internet Search engine can find the answers to a query so quickly? It is a four-step process:

1. *Crawling* the Web, following links to find pages.
2. *Indexing* the pages to create an index from every word to every place it occurs.
3. *Ranking* the pages so the best ones show up first.
4. *Displaying* the results in a way that is easy for the user to understand.

Crawling is conceptually quite simple: starting at some well-known sites on the web, recursively follow every hypertext link, recording the pages encountered along the way. In computer science this is called the transitive closure of the *link* relation. However, the conceptual simplicity hides a large number of practical complications: sites may be busy or down at one point, and come back to life later; pages may be duplicated at multiple sites (or with different URLs at the same site) and must be dealt with accordingly; many pages have text that does not conform to the standards for HTML, HTTP redirection, robot exclusion, or other protocols; some information is hard to access because it is hidden behind a form, Flash animation or Javascript program. Finally, the necessity of crawling 100 million pages a day means that building a crawler is an exercise in distributed computing, requiring many computers that must work together and schedule their actions so as to get to all the pages without overwhelming any one site with too many requests at once.

A search engine's *index* is similar to the index in the back of a book: it is used to find the pages on which a word occurs. There are two main differences: the search engine's index lists every occurrence of every word, not just the important concepts, and the number of pages is in the billions, not hundreds. Various techniques of compression and clever representation are used to keep the index "small," but it is still measured in terabytes (millions of megabytes), which again means that distributed computing is required. Most modern search engines index *link* data as well as word data. It is useful to know how many pages link to a given page, and what are the quality of those pages. This kind of analysis is similar to citation analysis in bibliographic work, and helps establish which pages are authoritative. Algorithms such as PageRank and HITS are used to assign a numeric measure of authority to each page. For example, the PageRank algorithm says that the rank of a page is a function of the sum of the ranks of the pages that link to the page. If we let PR(p) be the PageRank of page p, Out(p) be the number of outgoing links from page p, Links(p) be the set of pages that link to page p and N be the total number of pages in the index, then we can define PageRank by

$$PR(p) = r/N + (1 - r) \sum_{i \subset \mathrm{Links}(p)} PR(i)/Out(i)$$

where r is a parameter that indicates the probability that a user will choose not to follow a link, but will instead restart at some other page. The r/N term means that each of the N pages is equally likely to be the restart point, although it is also possible to use a smaller subset of well-known pages as the restart candidates. Note that the formula for PageRank is recursive – PR appears on both the right- and left-hand sides of the equation. The equation can be solved by iterating several times, or by standard linear algebra techniques for computing the eigenvalues of a (3-billion-by-3-billion) matrix.

The two steps above are *query independent*—they do not depend on the user's query, and thus can be done before a query is issued with the cost shared among all users. This is why a search takes a second or less, rather than the days it would take if a search engine had to crawl the web anew for each query. We now consider what happens when a user types a query.

Consider the query ["National Academies" computer science], where the square brackets denote the beginning and end of the query, and the quotation marks indicate that the enclosed words must be found as an exact phrase match. The first step in responding to this query is to look in the index for the *hit lists* corresponding to each of the four words “National,” “Academies,” “computer” and “science.” These four lists are then intersected to yield the set of pages that mention all four words. Because “National Academies” was entered as a phrase, only hits where these two words appear adjacent and in that order are counted. The result is a list of 19,000 or so pages.

The next step is *ranking* these 19,000 pages to decide which ones are most relevant. In traditional information retrieval this is done by counting the number of occurrences of each word, weighing rare words more heavily than frequent words, and normalizing for the length of the page. A number of refinements on this scheme have been developed, so it is common to give more credit for pages where the words occur near each other, where the words are in bold or large font, or in a title, or where the words occur in the anchor text of a link that points to the page. In addition the query-independent authority of each page is factored in. The result is a numeric score for each page that can be used to sort them best-first. For our four-word query, most search engines agree that the Computer Science and Telecommunications Board home page at www7.nationalacademies.org/cstb/ is the best result, although one preferred the National Academies news page at www.nas.edu/topnews/ and one inexplicably chose a year-old news story that mentioned the Academies.

The final step is *displaying* the results. Traditionally this is done by listing a short description of each result in rank-sorted order. The description will include the title of the page and may include additional information such as a short abstract or excerpt from the page. Some search engines generate query-independent abstracts while others customize each excerpt to show at least some of the words from the query. Displaying this kind of query-dependent excerpt means that the search engine must keep a copy of the full text of the pages (in addition to the index) at a cost of several more terabytes. Some search engines attempt to cluster the result pages into coherent categories or folders, although this technology is not yet mature.

Studies have shown that the most popular uses of computers are email, word processing and Internet searching. Of the three, Internet searching is by far the most sophisticated example of computer science technology. Building a high-quality search engine requires extensive knowledge and experience in information retrieval, data structure design, user interfaces, and distributed systems implementation.

Future advances in searching will increasingly depend on statistical natural language processing [Lee] and machine learning [Mitchell] techniques. With so much data—billions of pages, tens of billions of links, and hundreds of millions of queries per day—it makes sense to use data mining approaches to automatically improve the system. For example, several search engines now do spelling correction of user queries. It turns out that the vast amount of correctly and incorrectly spelled text available to a search engine makes it easier to create a good spelling corrector than traditional techniques based on dictionaries. It is likely that there will be other examples of text understanding that can be done better with a data-oriented approach; this is an area that search engines are just beginning to explore.

Recommended Reading

S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins. "Hypersearching the Web." *Scientific American*, June 1999. (pp. 54-60.)