Optimization for Keyword-Based Queries by using Web Search Engine

Nripendra Narayan Das, Monika Gupta

Dept. of CSE & IT, ITM University, Gurgaon, India

Abstract
Internet search engines have popularized the keyword based search technique. While traditional database management systems offer powerful query languages, they do not allow keyword-based search. What type of keyword which has to be entered for maximum result. How search engine work for processing that query.

Keywords
Web Search Engine, Crawler, Indexing, Hypertext, Boolean

I. Introduction
Web search engines are very successful for searching textual documents, images, and videos. On the other hand, there are also vast collections of structured and semi-structured data both on the Web and in enterprises, referred to as “databases” in this paper, such as relational databases, XML, data extracted from text documents, workflows, etc. Traditionally, to access these resources, users have to learn structured query languages, such as SQL or XQuery. Besides, users also need to know the schemas of the data, which are most likely complex, fast-evolving, or even unavailable in Web applications. The prevalence of web search engines, a natural question to ask is whether we can empower users to effectively search and explore databases using keyword queries. There are several immediate advantages of this keyword query-based approach to search and explore the databases. First, it can relieve casual users of studying structured query languages and data schemas for accessing structured data. Second, it allows users to easily access heterogeneous data from websites with database backends, this approach provides a more flexible search method. Third, ideally the result of a keyword search over databases will automatically assemble relevant pieces of data that are in different locations but are inter-connected and collectively relevant to the query. Thus unlike the results to a structured query, the results to a keyword query may reveal interesting or unexpected.

II. Search Strategies for Keyword
Suppose a typical user of a Web search engine with an information need for which she can come up with a set of potential keywords. The user will submit a query containing some of these words and expects a reply in the form of a ranked result list along with an estimation of the total number of results. Experience shows that very long queries return few or even no hits while rather short Web queries are likely to return millions of documents. Of course, such short queries are often answered reasonably well, i.e., among the top-ranked results the user has a good chance to find a matching item. However, these “good-natured” queries are not the focus of this paper; we address scenarios where the user is not satisfied by the results of her initial query. This case is not uncommon, and search engines provide different means of supporting users in such a situation. Examples include query expansion for queries returning lots of hits or spelling correction for queries returning no hits due to typos.

III. Approximate Keyword Search
Most search engines assume an exact matching model i.e. The queried keyword has to be the same as the indexed keyword such that the desired web pages can be matched. So far as we know no popular search engines have provided approximate matching capability that can find similar matches of the queried keyword. For example, if the query keyword is “weebserch” while the desired page uses “websearch” for the composer’s name, that page will not be found. There are circumstances where the approximate matching is needed. Some user might want to find a web site that has a name similar to a well known name. For example, if we forget the name of a website whose name is www.bozart.com, but remember that its name is similar to Mozart, with the approximate matching capability, we may be able to find the web site by querying keyword “Mozart” . In other conditions, the user might have spelling errors of the keywords or the desired web pages have misspellings or spelling variations of the queried keyword. In such cases, the approximate matching capability is needed to retrieve the desired pages.

Most search engines now provide spelling error corrections [1-3] for the user’s queries to provide partial solution to the problem. For example, if the query is “Mosart”, then search engines will prompt a corrected spelling “Mozart” for the user to issue a corrected query.

However, for the cases where the user does not have spelling problem but want to find similar matches, or in the cases there are misspellings or spelling variations in the wanted web pages, the spelling correction in the query is not helpful. Up to now, we have seen very few research results on approximate matching for search engines.

IV. How Web Search Engine Work?
A web search engine is designed to search for information on the World Wide Web (The World Wide Web or the proper World-Wide Web; abbreviated as WWW or W3 and commonly known as the Web) is a system of interconnected hypertext documents accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks.) and FTP servers.

The search results are generally presented in a list of results often referred to as SERPS, or “search engine results pages”. The information may consist of web pages, images, video, and other types of files.

A. All Search Engine Perform Three Basic Tasks
1. The searcher types a query into a search engine.
2. Search engine software quickly sorts through literally millions of pages in its database to find matches to this query.
3. The search engine’s results are ranked in order of relevancy.

B. A Search Engine Operates in the Following Order:
1. Web crawling
2. Indexing
3. Searching
A web crawler is a relatively simple automated program, or script, that methodically scans or “crawls” through Internet pages to create an index of the data it’s looking for; these programs are usually made to be used only once, but they can be programmed for long-term usage as well. There are several uses for the program, perhaps the most popular being search engines using it to provide web surfers with relevant websites. Other users include linguists and market researchers, or anyone trying to search information from the Internet in an organized manner. Alternative names for a web crawler include web spider, web robot, bot, crawler, and automatic indexer. Crawler programs can be purchased on the Internet, or from many companies that sell computer software, and the programs can be downloaded to most computers. Search engines frequently use web crawlers to collect information about what is available on public web pages. Their primary purpose is to collect data so that when Internet surfers enter a search term on their site, they can quickly provide the surfer with relevant web sites.

**V. Web Crawling**

Web crawling is an important method for collecting data on, and keeping up with, the rapidly expanding Internet. A vast number of web pages are continually being added every day, and information is constantly changing. A web crawler is a way for the search engines and other users to regularly ensure that their databases are up-to-date. There are numerous illegal uses of web crawlers as well such as hacking a server for more information than is freely given.

**A. How Web Crawler Works?**

When a search engine’s web crawler visits a web page, it “reads” the visible text, the hyperlinks, and the content of the various tags used in the site, such as keyword rich meta tags. Using the information gathered from the crawler, a search engine will then determine what the site is about and index the information. The website is then included in the search engine’s database and its page ranking process. Hopefully after the technical issues have subsided. Examples Web crawlers may operate one time only, say for a particular one-time project. If its purpose is for something long-term, as is the case with search engines, web crawlers may be programmed to comb through the Internet periodically to determine whether there has been any significant changes. If a site is experiencing heavy traffic or technical difficulties, the spider may be programmed to note that and revisit the site again.

Example:

Yahoo! Slurp is the name of the Yahoo Search crawler. Bingbot is the name of Microsoft’s Bing webcrawler. It replaced Msnbot.

**C. Web Crawler**

A web crawler is a relatively simple automated program, or script, that methodically scans or “crawls” through Internet pages to create an index of the data it’s looking for; these programs are usually made to be used only once, but they can be programmed for long-term usage as well. There are several uses for the program, perhaps the most popular being search engines using it to provide web surfers with relevant websites. Other users include linguists and market researchers, or anyone trying to search information from the Internet in an organized manner. Alternative names for a web crawler include web spider, web robot, bot, crawler, and automatic indexer. Crawler programs can be purchased on the Internet, or from many companies that sell computer software, and the programs can be downloaded to most computers. Search engines frequently use web crawlers to collect information about what is available on public web pages. Their primary purpose is to collect data so that when Internet surfers enter a search term on their site, they can quickly provide the surfer with relevant web sites.

**B. Indexing**

Once the spiders have completed the task of finding information on Web pages (and we should note that this is a task that is never actually completed -- the constantly changing nature of the Web means that the spiders are always crawling), the search engine must store the information in a way that makes it useful. There are two key components involved in making the gathered data accessible to users:

- The information stored with the data
- The method by which the information is indexed

In the simplest case, a search engine could just store the word and the URL where it was found. In reality, this would make for an engine of limited use, since there would be no way of telling whether a word was used in an important or a trivial way on the page, whether the word was used once or many times or whether the page contained links to other pages containing the word. In other words, there would be no way of building the ranking list that tries to present the most useful pages at the top of the list of search results. To make for more useful results, most search engines store more than just the word and URL. An engine might store the number of times that the word appears on a page. The engine might assign a weight to each entry, with increasing values assigned to words as they appear near the top of the document, in sub-headings, in links, in the meta tags or in the title of the page. Each commercial search engine has a different formula for assigning weight to the words in its index. This is one of the reasons that a search for the same word on different search engines will produce different lists, with the pages presented in different orders. Regardless of the precise combination of additional pieces of information stored by a search engine, the data will be encoded to save storage space. For example, the original Google paper describes using 2 bytes, of 8 bits each, to store information on weighting — whether the word was capitalized, its font size, position, and other information to help in ranking the hit. Each factor might take up 2 or 3 bits within the 2-byte grouping (8 bits = 1 byte). As a result, a great deal of information can be stored in a very compact form. After the information is compacted, it’s ready for indexing. An index has a single purpose: It allows information to be found as quickly as possible. There are quite a few ways for an index to be built, but one of the most effective ways is to build a hash table. In hashing,
a formula is applied to attach a numerical value to each word. The formula is designed to evenly distribute the entries across a predetermined number of divisions. This numerical distribution is different from the distribution of words across the alphabet, and that is the key to a hash table’s effectiveness.

In English, there are some letters that begin many words, while others begin fewer. You’ll find, for example, that the “M” section of the dictionary is much thicker than the “X” section. This inequity means that finding a word beginning with a very “popular” letter could take much longer than finding a word that begins with a less popular one. Hashing evens out the difference, and reduces the average time it takes to find an entry. It also separates the index from the actual entry. The hash table contains the hashed number along with a pointer to the actual data, which can be sorted in whichever way allows it to be stored most efficiently. The combination of efficient indexing and effective storage makes it possible to get results quickly, even when the user creates a complicated search.

C. Searching
A web search query is a query that a user enters into web search engine to satisfy his or her information needs. Web search queries are distinctive in that they are unstructured and often ambiguous; they vary greatly from standard query languages which are governed by strict syntax rules.

1. Building a Search
Searching through an index involves a user building a query and submitting it through the search engine. The query can be quite simple, a single word at minimum. Building a more complex query requires the use of Boolean operators that allow you to refine and extend the terms of the search.

The Boolean operators most often seen are:
- **AND** - All the terms joined by “AND” must appear in the pages or documents. Some search engines substitute the operator “+” for the word AND.
- **OR** - At least one of the terms joined by “OR” must appear in the pages or documents.
- **NOT** - The term or terms following “NOT” must not appear in the pages or documents. Some search engines substitute the operator “-” for the word NOT.
- **FOLLOWED BY** - One of the terms must be directly followed by the other.
- **NEAR** - One of the terms must be within a specified number of words of the other.
- **Quotation Marks** - The words between the quotation marks are treated as a phrase, and that phrase must be found within the document or file.

2. Future Search
The searches defined by Boolean operators are literal searches the engine looks for the words or phrases exactly as they are entered. This can be a problem when the entered words have multiple meanings. “Bed,” for example, can be a place to sleep, a place where flowers are planted, the storage space of a truck or a place where fish lay their eggs. If you’re interested in only one of these meanings, you might not want to see pages featuring all of the others. You can build a literal search that tries to eliminate unwanted meanings, but it’s nice if the search engine itself can help out.

One of the areas of search engine research is concept-based searching. Some of this research involves using statistical analysis on pages containing the words or phrases you search for, in order to find other pages you might be interested in. Obviously, the information stored about each page is greater for a concept-based search engine, and far more processing is required for each search. Still, many groups are working to improve both results and performance of this type of search engine. Others have moved on to another area of research, called natural-language queries. The idea behind natural-language queries is that you can type a question in the same way you would ask it to a human sitting beside you no need to keep track of Boolean operators or complex query structures.

VI. Working of Web Search Engine

- **Fig. 3: Working of Web Search Engine**

A. Market Analysis of Previous Years for Searching Data by using Search Engine

- **Fig. 4: Market Analysis of 2010 and 2011 for Searching Data using Search Engine**
References


Nripendra Narayan Das received the M.Tech. degree in Information Technology from Guru Gobind Singh Indraprastha University in 2007. Currently He is pursuing Ph.D. in Computer Science from Gautam Budha University. His research interests include Web Data extraction and Deep Web data integration.

Monika Gupta received the B.Tech. degree in Information Technology from Maharshi Dayanand University. Currently She is pursuing M.Tech, in Computer Science from ITM University. Her research interests include Web Data extraction and Deep Web data integration.