The application of index entries to search and retrieval of books and book content

Michael Stelmach

The use of index entries for search and retrieval of books in terms of their content presents the opportunity to bring printed books into the mainstream of electronically accessible information. Instead of having only a few entry points to find a book, library users and bookstore customers will be able to search on hundreds or thousands of weighted terms.

Introduction

Until now there has been very limited ability to search for books. Most access points such as Author, Title, International Standard Book Number (ISBN), and Library of Congress catalog card number (LCCN) require that patrons and customers already have very specific information about what they are looking for. These searches are generally used simply to confirm that a known book is available from a given source. Typically, this has meant a labor-intensive manual search through a library catalog to locate a book in the library's collection.

With the advent of uniform subject cataloging, using Library of Congress subject access terms, users were given a tool to do fairly broad subject searches to locate books without needing to know the specific work. The online public access catalog (OPAC) allowed users to search for books more quickly, but no more thoroughly. The OPAC uses the same search fields as the traditional card catalog. The author/ title/ subject search model for books remains largely unchanged from what probably existed in the 3rd century BC.

Not only researchers, but the average book consumer can find searching for the right book a trying experience. In the library community, organizations such as the Carl Corporation are allowing users to search hundreds of library collections. Among others, these include the Library of Congress, Yale Library, and the combined libraries of the University of California system. Up to six library collections can be searched simultaneously. On the commercial side, bookstores on the Web are advertising millions of titles. With all these resources, users are still conducting the same basic searches used for centuries: author, title, subject.

Searches are run on remote libraries and bookstores that are hundreds or thousands of miles from the user. The result is that, unlike past generations of book users, current customers or patrons often do not have physical access to the books.

Clearly there is a need for library patrons and bookstore customers to be able to conduct effective searches and to examine the content of books without the books being physically present.

Index-based searching

Centuries before the advent of computers, the index served as a physical database providing search and retrieval capability complete with a passive thesaurus (see and see also references), relevance ranking (page counts), and a hierarchical structure.

Furthermore, the index is much more than an alphabetic arrangement of terms appearing in the body of the book. An index is an evaluation of the book rather than a simple summary. Words appearing numerous times in a book may never make it into an index because they are incidental and are not used to provide information on that topic. Another word only mentioned once may appear as a index entry heading with subordinate subheadings. This is the value of having an index prepared by an experienced indexer analyzing a book. There is no comparable analytical tool that can be used to achieve a similar result, including those used by current full-text search engines. (These can include natural language queries, fuzzy searching, phonetic searching, inferential searching, and other complex searching algorithms.)

Quantitative measure of relevance

The page locators that follow index entries are one of the greatest strengths of a book index. The locators provide not only the location, but the number of pages on which an index term or concept appears. Even though they do not indicate how much text is devoted to the concept, they provide an invaluable measure of relevance which is unavailable with subject searches, and in many ways exceeds the relevance potential of full-text searches. Index locators have the advantage over full-text searches in that they refer to the substantive treatment of a concept rather than the simple occurrence of a word or combination of words.

For example, a full-text search on the phrase “heart attack” may result in fifty hits even though the book may not have any substantive discussion on the topic. If an index search results in fifty pages on the phrase you can be certain that there is substantive treatment of the topic.

Making indexes searchable across titles

With the great variability across indexes, the content and structure of each index must be evaluated, the associated metadata stored along with the data, and the indexes then reconstructed for display. There are four major tasks that must be accomplished by an application in order to allow search and retrieval capability across multiple book indexes.

1. Index parsing
2. Database loading
3. Database searching
4. Reconstructed index display
The database allows a great number of search options. Searches reside on the database. Any records that fail to match the combined with bibliographic data based on ISBN, LCCN, or any cataloging data for the books. Finally, the parsed data can be page totals, cross-references and targets, and character properties such as bold, italic, underlined text, and diacritical marks. In effect, the parser identifies and tags all the index information necessary to populate a searchable database and to reconstruct an HTML version of the index.

Once the parser has completed processing an index, it creates a file that can be loaded into a relational database. This file will allow the database to create tables for various search and display options for the index.

2. Database loading
The database loader takes all of the data from the parser and puts it into the appropriate tables of a relational database. Then, it links the index data with files containing bibliographic and cataloging data for the books. Finally, the parsed data can be combined with bibliographic data based on ISBN, LCCN, or any unique identifier.

The loader also checks index and bibliographic data against a number of validation rules to insure consistency of the data that reside on the database. Any records that fail to match the validation criteria are sent to a log file to be checked at a later time.

3. Database searching
The database allows a great number of search options. Searches can be conducted against index terms (one or many), subject terms, author, title, date, publisher, or any desired combination of fields.

Index searches and relevance ranking
Searching on index terms provides the greatest strength to the database. The searches are run against modified representations of index entries. The information derived from the parser allows the index headings and subheadings to be concatenated and the page counts to be modified to give an accurate representation of index term relevance. The examples provided below illustrates a small part of how the database stores the index entries, particularly the page count for relevance ranking.

Example 1: Original index format
data, 34-35
dynamic, 3, 41, 53
defined, 3
redundant, 6-7, 188-189, 280
defined, 6
static, 4, 34-35, 41
defined, 4

Example 2: Database index format

<table>
<thead>
<tr>
<th>ENTRY</th>
<th>HEADING</th>
<th>SUB HEADING</th>
<th>SUB-SUB HEADING</th>
<th>LEVEL</th>
<th>LOCATORS</th>
<th>PAGE COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>dynamic</td>
<td>1</td>
<td>xii, 34-35</td>
<td>2</td>
<td>3, 41, 53</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>defined</td>
<td>2</td>
<td>3</td>
<td></td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>redundant</td>
<td>2</td>
<td>6-7, 188-189, 280</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>defined</td>
<td>3</td>
<td>6</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>static</td>
<td>2</td>
<td>4, 34-35, 41</td>
<td>4</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>defined</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>SUM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14</td>
</tr>
</tbody>
</table>

The total page count is calculated from the locators and is used to inform the user how relevant each book in a result list is to the search criteria. In the example above, a search on “data” and “static” would have resulted in a relevance ranking of four.

Relevance ranking is useful both for determining the value of an individual book related to a user’s information need as well as comparing the relative value of many books that all match a user’s needs to some extent.

Page count redundancy
Notice that the page count in Example 2 for the entry as a whole does not add up to 19, which would be the case if the numbers were summed normally. The missing five pages are accounted for in the repetition of pages 34-35, 3, 6, and 4. Repeating the page count would give a false indication of the total number of pages where “Data” was discussed. In this case, a search on the index term “data” would provide a relevance ranking of 14 because all of the subheadings also deal with “data”. Sub-sums are also calculated for subheadings containing sub-subheadings using the same logic.

Page redundancy is not only accounted for within an index entry, but across index entries as well. For example, the word “helicopter” may occur as a subheading under the main headings of “Aircraft,” “Defense Contractors,” “Ground Support,” and “Sikorski”. It is very likely that there will be overlap in the pages for “helicopter” as a subheading. These duplicate pages are treated in the same manner as duplicate pages within an index.

4. Reconstructed HTML index display
Once the users conduct a search, they are provided with the option to view the bibliographic record or the index (many other options are available as well). If the user chooses to view the index, the display brings up an index reconstructed from the information derived from the parser. The reconstructed index is displayed using the viewer, and includes all the original character properties such as bold, underline, and italics. Diacritical marks also appear as in the original index.

Reconstructing the index provides many advantages over simply displaying an image of the original index. By rebuilding the index from information stored in the database, we can make the index display uniform and embed tools for navigation. It is also possible to customize the display to fit special requirements.

Uniform display
By reconstructing an index we are able to provide a uniform display regardless of the original index format (indented, run-in, hybrid). All indexes are displayed as HTML indented style indexes. Type size, indents, cross-references, and locator format are all standardized. This makes it much easier for the user to compare a multitude of indexes that can result from a typical search.
Navigation tools — alphabetic

The user interface provided for the display of the reconstructed indexes features several navigation tools. If the user had run an index search and then chosen to display the index, the index would be displayed at the first occurrence of their search term. A "next hit" button would allow them to see all of their search terms in context. An alphabetic range location tool would also allow the user quickly to go to any alphabetic placement in the index. These tools are illustrated below.

Cross-reference links

The index display also contains embedded links for all see and see also references with valid targets (single or multiple). Simply clicking on a highlighted cross-reference would bring the user to the index term being referenced. Cross-references with multiple targets would allow the users to select individually which of the multiple terms they would like to go to. Below are two screen shots which illustrate the use of cross-reference links.

Searching within the displayed index

A simple text entry box would allow users to search within an individual index for specific words or phrases (text). Although the simple alphabetic navigation tools are sufficient for finding main heading text, the user may wish to locate text that is subordinate to the main heading. It is also quite possible that the desired text resides in several locations within an index, both in main headings and subordinate headings.
Expanded capabilities

Although this paper has concentrated on index searching, the database described here provides the user with the ability to search any combination of index, subject, and bibliographic search terms. The search result display can be sorted by date, author, title, price, or index ranking (if an index term was included in the search).

The database also has the capability to store and display the book table of contents, book cover art, publisher notes on the book, and author biography. While concentrating on the search strength of book indexes, the database has been designed to accommodate any other relevant search or display information related to a book.

Applications for index-based search technology

There are myriad applications for index-based search and retrieval. Some of the most obvious include libraries, bookstores, and publishers. There are other less obvious uses, such as automated thesaurus construction based on index cross-references. Thesauri could be specific to a collection, subject area, or publisher. Indexes and the associated retrieval tool could be made available on the Web, CD-ROM, or local area networks.

Example of implementation to an existing web bookstore

Since a discussion on each possible application of index-based searching is beyond the scope of a journal article, this section is limited to providing a theoretical model of how application of index search and retrieval software to a Web-based bookstore (Amazon.com in this case) would increase book sales and customer satisfaction.

The tools developed by IndexLogic would provide several advantages to the current Amazon.com site. These include:

- Greater opportunities for the customer to find books covering a specific or specialized topic
- Ability to present the customer with a greater selection of books on a given subject
- Ability for the customer to choose the “best” book on a given subject
- Opportunity for the customer to review the contents of a book before purchasing it
- Ability to present the customer with alternate search terms, resulting in more choices

These can be generalized to two basic tools for the customer. First is the ability to find books using terms that are not included in either the subject or title of a book. Second is the ability to narrow the selection of books when the general subject or keyword search results in a large number of titles in the result list.

Example of a typical book search

In this example the customer is looking for books on the Perl scripting language. The customer has need for a general text, but has special interest in strings and filehandles.

Your Book Search Results
for: the subject words include "perl"

Top matches for this search:

- Programming Perl (Nutshell Handbook) - Larry Wall (Editor), et al, Paperback
- Effective Perl Programming: Writing Better Programs With Perl - Joseph N. Hall, Paperback

Full Results: The first 100 of 107 are shown below. To see more results scroll down and click the "more" button.
The application of index entries to search and retrieval of books and book content

The screen above shows the results from using "perl" as a subject search.

Of the 268 book titles in the result set, there is no indication of which books would best fit the customer's need for comprehensive treatment of strings and filehandles. Adding either “strings” or “filehandles” in the search does not yield any results.

The text boxes give examples from two books on Perl showing what information can be provided to the customer when the index data is made available. In these examples only the entries under the main headings for “strings” and “filehandles” has been included.

Example 3


**filehandles — 7 pages**

- filehandle
  - localizing with typeglob, 238
  - passing, 101
- passing by reference to typeglob, 160
- passing with FileHandle, 102
- passing with IO file, 102
- passing with typeglob, 102-103
- recognized in context, 10
- tied, 213

**strings — 3 pages**

- string comparison operators
  - instead of regular expressions, 74
- look like words, 17
- string context, 19


**filehandles — 17 pages**

- caching open filehandles, 258-259
- copying, 262-263
- non-blocking I/O, 252
- printing to multiple simultaneously, 259-260
- reading from many, 250-252
- reporting filenames in errors, 232
- storing as variables, 222, 255-258
- tied, 481-482, 487-488

**strings — 68 pages**

- strings, 1-41
- accessing substrings, 3-6
- checking if valid numbers, 44-45
- converting ASCII and values, 9-11
- converting ASCII to/from HTML, 714-715
- converting case, 19-21
- copying and substituting, 164-165
- default values for, 6-8
- duplicate words, finding, 194
- escaping characters, 28-30
- hash element indices, 129
- HTML text substitutions, 731-733
- interpolating functions/expression within, 21-22
- matching letters, 165-166
- matching words, 167
- numeric operators with, 477-478
- parsing datetime information in, 81-82
- plurals based on numbers, 65-67
- processing characters individually, 11-13
- psgrep program (example), 37-41
- reading from binary files, 296-297
- reformating paragraphs, 26-28
- removing leading/trailing spaces, 30-31
- removing/extracting HTML tags, 716-718
- substituting specific words, 34-37
- text color, changing, 522-52

Between books one and two we see a 243% difference in the number of page references for the term “filehandles” and a 2,267% difference in the number of page references for the term “strings”. Not only are there huge quantitative differences, but if a potential customer was interested in specific string topics such as extracting HTML tags or changing text color, a quick look at the index would let him or her know that book 2 does indeed cover these topics.

Similar results are realized in trying to find specific topical treatment in books on SQL, HTML, C++, or other software topics using the standard bibliographic searches offered on bookstore sites. Generally, entering the general area as a subject search produces over 100 titles in the result set, but narrowing the search to a particular topic within the field produces no results.

Observing people in a traditional bookstore, one will notice customers in the aisles dedicated to computer books glancing at the table of contents and poring over the indexes. Making the index available for search and display would actually give the customers an advantage over the browsing the shelves because they can search across the books rather than search through each individually.

Michael Stelmach is President of IndexLogic. He has previously worked with library collection automation, information management system design, and software project management. Email: mstelmach@indexlogic.com.