INDEXING AND RETRIEVAL SYSTEMS ON THE WEB FOR DESIGN AND DEVELOPMENT OF A LOW COST DIGITAL LIBRARY

by

Satish K. M*
Jayashree S**

ABSTRACT

The world wide web has become an undisputed and powerful medium to integrate multiple information sources and services on the part of the libraries, leading to continual development of new applications and services. Efficient implementation of existing services and an opportunity for delivering information to the desktop of remote and local users has become an immediate priority on the part of the libraries. Locally owned resources like the OPAC, locally hosted electronic journals and databases, alerting services like new additions and content pages, and integrating host of other services call for an unpretentious approach for successful information services. In this paper an effort has been made to study the functions of various indexing and retrieval tools available on the web for their understanding and experience with standard indexing practices facilitating accurate and efficient data retrieval leading to the abstraction of digital library.

Keywords: HTML, SGML, XML, Content Management – Document Formation

*** Assistant Librarian, Goa University, Taleigao Plateau, Panaji – 403 206**
**** Technical Assistant, National Aerospace Laboratories, Bangalore – 560 017**

0 Introduction

Several free and commercial indexing and retrieval software have been developed that address the unique search and retrieval need of user communities. They are basically designed to crawl and index web servers or portions of these servers to create custom and searchable indexes of the documents and data housed on the servers. They have features that are common with Internet search engines, but also contain some features that are unique, viz., provide indexing for other document formats like the PDF, word processing. Spread sheets, databases, graphics and others contained in an intranet web site and are usually designed to provide more precise data filtering and retrieval limiting the quantum of information the user is required to sift through. Information professionals familiar with the indexing and subsequent searching process can lend a lot to the evaluation and implementation of these indexing and retrieval tools within the libraries. The need is to familiarise with the products available and the issues surrounding their selection, implementation and use. In-depth knowledge of searching techniques, together with the use of controlled vocabulary, Boolean operators, proximity operators and relevance ranking is necessary for evaluation. An understanding and experience with standard indexing practices and parameters can also ensure that the data contained in the various indexes built using these tools will facilitate accurate and efficient data retrieval. Although there are many players in this area, we shall limit and discuss the features and functionality of the following free indexing and retrieval systems for bibliographic and fulltext data: freeWAIS-sf, Isite/Isearch, MPS, Yaz/Zebra. We also look into the tools for indexing web sites and HTML files: Harvest, SWISH, Ht://Dig and WebGlimpse.

As the size of information systems increase so does the necessity of providing searchable interfaces to the underlying data. Indexing content and implementing an HTML form to search the index is one way to accomplish this goal, but all indexing and retrieval tools are not same. This case study enumerates the pros and cons of the above mentioned toolkits currently available and makes recommendations on which to use and for what purposes. The scope is limited to the indexing of ASCII format. Finally, this case study will make readers aware that good search interface alone does not make for good information systems. Good information systems also require consistently applied subject analysis and well-structured data.

1 Bibliographic and Fulltext data

freeWAIS-sf

Of the indexing and retrieval tools analysed here, free WAIS-sf is by far the oldest, and the predecessor Isite/Isearch, SWISH, and MPS. Yet, freeWAIS-sf is not really the oldest indexer because it owes its existence to WAIS originally developed by Brewster Kahle of Thinking Machines, Inc., in 1991.

FreeWAIS-sf supports a bevy of indexing types. For example, it can easily index Unix mbox files, text files where records are delimited by blank lines, HTML files, as well as others. Sections of these text files can be associated with fields for field searching through the creation "format files" -- configuration files made up of regular expressions. After data has been indexed, it can be made accessible through a CGI interface called SFgate, but the interface relies on a Perl module, WAIS.pm, which is difficult to compile as it uses some of the shared objects evolved after compiling the freeWAIS-sf . The interface supports lots of search features including field searching, nested queries, right-hand truncation, thesauri, multiple-database searching, and Boolean logic. This indexer represents aging code. Not because it doesn't work, but because as new versions of operating systems evolve freeWAIS-sf get harder and harder to install. After many trials and tribulations, it has been possible to compile and install on RedHat Linux, and has been found most useful for indexing two types of data: archived email and public domain electronic texts. For example, by indexing archived email, one can do free text searches against the archives and return names, subject lines, and ultimately the email messages (plus any attachments). Using the "para" indexing type it is possible to index a small collection of public domain literature and provide a mechanism to search one or more of these texts simultaneously for keywords like "dynasty" to identify paragraphs from the collection. It is also amenable to index the bibliographic records separated by any delimiter. Conspicuous limitation here is the generation of a large size of the index files when compared to its contemporaries.

Isite/Isearch

Isearch is a software system for searching though large amounts of text. Developed by the Clearinghouse for Networked Information Discovery and Retrieval (CNIDR) in 1994. The system allows a user to very quickly find out what documents are available that contain certain words. Unlike older search systems, Isearch does not use a list of keywords or an abstract; every word of every document can be checked. This allows greatly improved chances of discovering new information in old collections. Handles very large collections: over 1-gigabyte collections can be handled on modest servers. Essentially unlimited textbases can be searched with careful layout and planning. Very sophisticated result sorting: The documents most likely to be useful are returned first. Ranking is based on statistical analysis of word frequencies and is generalized for a wide variety of subjects and user skill levels. Works well with OCR document storage and retrieval systems: no need for people to classify documents, and the statistical ranking method is forgiving of OCR errors. Easy to customize: The modular, object-oriented structure of Isearch means that new features can be added independently of the Isearch core. Third party extension is facilitated by using well-defined Application Programming Interfaces (APIs) implemented in C++. Integrates smoothly with World Wide Web (WWW) and ANSI Z39.50 servers: Anyone can search an Isearch textbase using web browser. When used with Isite package, Isearch can be used through a Z39.50 session to interoperate with library automation software. Isearch and Isite together form three-tier client-server architecture to allow essentially unlimited capacity growth.

Isite/Isearch is one of the very first implementations based on the WAIS code. It is intended to support the Z39.50 information retrieval protocol. Like freeWAIS it supports a number of file formats for indexing. Unfortunately, Isite/Isearch no longer seems to be supported and the documentation is weak. While it comes with a CGI interface and is easily installed, the user interface is difficult to understand and needs a lot of tweaking before it can be called usable by today's standards.

MPS

MPS seems to be the fastest of the indexers analysed. It can create more data in a shorter period of time than all of the other indexers. Unlike the other indexers MPS divides the indexing process into two parts: parser and indexer. The indexer accepts what is called a "structured index stream", a specialized format for indexing. By structuring the input, the indexer expects it is possible to write output files from the database application and have the content of database indexed and searchable by MPS. It is not limited to indexing the content of databases with MPS. Since it too was originally based on the WAIS code, it indexes many other data types such as mbox files, files where records are delimited by blank lines (paragraphs), as well as a number of MIME types (RTF, TIFF, PDF, HTML, SOIF, etc.). Like many of the WAIS derivatives, it can search multiple indexes simultaneously, supports a variant of the Z39.50 protocol, and a wide range of search syntax.

MPS also comes with a Perl API and an example CGI interface. The Perl API comes with the barest of documentation, but the CGI script is quite extensive. One of the neatest features of the example CGI interface is its ability to allow users to save and delete searches against the indexes for processing later. For example, if this feature is turned on, then a user first logs into the system. As the user searches the system their queries are stored to the local file system. The user then has the option of deleting one or more of these queries. Later, when the user returns to the system they have the option of executing one or more of the saved searches. These searches can even be designed to run on a regular basis and the results sent via email to the user. This feature is good for data that changes regularly over time such news feeds, mailing list archives, etc. MPS has a lot going for it. If it were able to extract and index the META tags of HTML documents, and if the structured index stream as well as the Perl API were better documented, then this indexer/search engine would ranking higher on the list.

Yaz/Zebra

The Yaz/Zebra combination is probably the best indexer/search engine solution for librarians who want to implement an open source Z39.50 interface. Z39.50 is an ANSI/NISO standard for information retrieval based on the idea of client/server. According to Library of Congress web site "It specifies procedures and structures for a client to search a database provided by a server, retrieve database records identified by a search, scan a term list, and sort a result set. Access control, resource control, extended services, and a help facility is also supported. The protocol addresses communication between corresponding information retrieval applications, the client and server (which may reside on different computers); it does not address interaction between the client and the end-user." In another words, Z39.50 tries to facilitate a "query once, search many" interface to indexes in a truly standard way, and the Yaz/Zebra combination is probably the best open source solution to this problem.

Yaz is a toolkit allowing creating Z39.50 clients and servers. Zebra is an indexer with a Z39.50 front-end. To make these tools work, the first thing to be done is to download and compile the Yaz toolkit. Once installed documents are fed to the Zebra indexer (it requires a few Yaz libraries) and make the documents available through the server. While the Yaz/Zebra combination does not come with a Perl API, there are at least a couple of Perl modules available from CPAN providing Z39.50 interface. There is also a module called ZAP (http://www.indexdata.dk/zap/) allowing embedding a Z39.50 client into Apache web server.

There is absolutely nothing wrong with the Yaz/Zebra combination. It is well documented, standards-based, as well as easy to compile and install. The difficulty with this solution is the protocol, Z39.50. It is considered overly complicated and therefore the configuration files need to be maintained and the formats of the files available for indexing are rather obtuse.

2 Indexing Websites and HTML Files

Harvest

Harvest was originally funded by a federal grant in 1995 at the University of Arizona. It is essentially made up of two components: gatherers and brokers. Given sets of one or more URLs, gatherers crawl local and/or remote file systems for content and create surrogate files in a format called SOIF. After one or more of the SOIF collections have been created they can be federated by a broker, an application indexing them and makes them available though a web interface.

The Harvest system assumes the data being indexed is ephemeral. Consequently, index items become "stale", are automatically removed from retrieval, and need to be refreshed on a regular basis. This is considered a feature, but if the content does not change very often it is more a hindrance than a benefit. Harvest is not very difficult to compile and install. It comes with a decent shell script allowing setting up rudimentary gatherers and brokers. Configuration is done through the editing of various text files outlining how output is to be displayed. The system comes with a web interface for administrating the brokers. If the indexed content is consistently structured and includes META tags, then it is possible to output very meaningful search results that include abstracts, subject headings, or just about any other fields defined in the META tags of the HTML documents. The real strength of the Harvest system lies in its gathering functions. Ideally system administrators are intended to create multiple gatherers. These gatherers are designed to be federated by one or more brokers.

SWISH

Kevin Hughes originally wrote SWISH. This software is a model of simplicity. To get it to work one needs to downloaded, unpack, configure, compile, edit the configuration file, and feed the file to the application. A single binary and a single configuration file are used for both indexing and searching. The indexer supports web crawling. The resulting indexes are portable among hosts. The search engine supports phrase searching, relevance ranking, stemming, Boolean logic, and field searches.

The hard part about SWISH is the CGI interface. Many SWISH CGI implementations pipe the search query to the SWISH binary, capture the results, parse them, and return them accordingly. Recently a Perl as well as PHP module have been developed allowing the developer to avoid this problem, but the modules are considered beta software. Like Harvest, SWISH can automatically extract the content of HTML META tags and make this content field searchable. Assume a META tag in the header of the HTML document such as this:

<META NAME="subject" CONTENT="adaptive technologies; CIL (Computers In Libraries);">

The SWISH indexer would create a column in its underlying database named "subject" and insert into this column the values "adaptive technologies" and "CIL (Computers In Libraries)". Then a query can be submitted to SWISH as this:

subject = "adaptive technologies"

This query would then find all the HTML documents in the index whose subject META tag contained this value resulting in a higher precision/recall ratio. This same technique works in Harvest as well, but since the results of a SWISH query are more easily malleable before they are returned to the web browser,. A specific field can easily sort SWISH results, or more importantly, SWISH results can be marked up before they are returned. For example, if CGI interface supports the GET HTTP method, then the content of META tags can be marked up as hyperlinks allowing the user to easily address the perennial problem of "Find me more like this one.” thus supporting the query by example search type.

Ht://Dig

This is simple web site indexer, but does not have the features of some of the other available distributions. Configuring the application for compilation is easy, but unless the paths are set correctly. Like SWISH, to index the data needs to be feed via the application configuration file and it then creates gobs of data. Many indexes can be created and they then have to be combined into a single database for searching.

The indexer supports Boolean queries, but not phrases searching. It can apply an automatic stemming algorithm. The search engine does not support field searching, and a rather annoying thing is that the indexer does not remove duplicates. Consequently, index.html files almost always appear twice in search results. On the other hand, one notable feature is it does do that the other engines don't do (except WebGlimpse) is highlight query terms in a short blurb (a pseudo-abstract) of the search results. Ht://Dig is a simple tool. Considering the complexity of some of the other tools covered here.

WebGlimpse

WebGlimpse is a newer incarnation of the original Harvest software. Like Harvest, WebGlimpse relies on Glimpse to provide an indexing mechanism, but unlike Harvest, WebGlimpse does not provide a means to federate indexes through a broker. Compilation and installation is rather harmless, and the key to using this application effectively is the ability to edit a small configuration file that is used by the indexer (archive.cfg). Once edited correctly, another binary reads this file, crawls a local or remote file system, and indexes the content. The indexes are then available through a simple CGI interface. Unfortunately, the output of the interface is not configurable unless the commercial version of the software is purchased. This is a real limitation, but on the other hand, the use of WebGlimpse does not require a separate pair of servers (a broker and/or a gatherer) running in order to operate. WebGlimpse reads Glimpse indexes directly.

3 Conclusion

Indexers provide one means for "finding a needle in a haystack" but not necessarily depend on it to satisfy user information needs; information systems require well-structured data and consistently applied vocabularies in order to be truly useful. Information systems can be defined as organized collections of information. In order to be accessed, they require elements of readability, browsability, searchability, and finally interactive assistance. It connotes meaningful navigation, a sense of order, and a systematic layout. As the size of an information system increases, it requires browsability -- an obvious organization of information that is usually embodied through the use of a controlled vocabulary. Searchability is necessary when a user seeks specific information and when the user can articulate their information need. Searchability flattens browsable collections. Finally, interactive assistance is necessary when an information system becomes very large or complex. Even though a particular piece of information exists in a system, it is quite likely that a user will not find that information and may need help. Interactive assistance is that help mechanism.

By creating well-structured data one can supplement the searchability aspects of the information system. For example, if the data is indexed is HTML, then insertion of META tags into the documents is useful and can be used as a controlled vocabulary -- a thesaurus – to describe those documents. If this is used then SWISH or Harvest can be used to extract these tags and provide field-searching access to the documents. Free text searches rely too much on statistical analysis and can not return as high precision/recall ratios as field searches.

The indexing and retrieval tools discussed here have different strengths and weaknesses. If the content is primarily HTML pages, then SWISH is most likely the application one would want to use. It is fast, easy to install, and since it comes wih no user interface one can create with just about any scripting language. If content is not necessarily HTML files, but structured text files then MPS or the Yaz/Zebra combination may be preferred. Both of these applications support a wide variety of file formats for indexing as well as the incorporation of standards.

4 References

1. freeWAIS-sf
   http://ls6-www.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf/
2. Harvest
   http://www.tardis.ed.ac.uk/harvest/
3. Ht://Dig
   http://www.htdig.org/
4. Isite/Isearch
   http://www.etymon.com/Isearch/
5. MPS
   http://www.fsconsult.com/products/mps-server.html
6. SWISH
   http://sunsite.berkeley.edu/SWISH-E/
7. WebGlimpse
   http://webglimpse.net/
8. Yaz/Zebra
   http://indexdata.dk/zebra/
9. http://lcweb.loc.gov/z3950/agency/markup/01.html