Comparing Open Source Indexers

Abstract

This text compares and contrasts the features and functionality of various open source indexers: freeWAIS-sf , Harvest , Ht://Dig , Isite/Isearch , MPS , SWISH , WebGlimpse , and Yaz/Zebra . 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 indexers are not created equal. This case study enumerates the pluses and minuses of various open source indexers currently available and makes recommendations on which indexer to use for what purposes. Finally, this case study will make readers aware that good search interfaces alone to not make for good information systems. Good information systems also require consistently applied subject analysis and well structured data.

Indexers

Below are a few paragraphs about each of the indexers reviewed here. They are listed in alphabetical order.

freeWAIS-sf

Of the indexes reviewed here, freeWAIS-sf is by far the grand daddy of the crowd, 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. as long ago as 1991 or 1992.

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 very difficult to compile. The interface supports lots o' 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 incarnations of operating systems evolve freeWAIS-sf get harder and harder to install. After many trials and tribulations, I have been able to get it to compile and install on RedHat Linux, and I have found it most useful for indexing two types of data: archived email and public domain electronic texts. For example, by indexing my archived email I can do free text searches against the archives and return names, subject lines, and ultimately the email messages (plus any attachments). This has been very helpful in my personal work. Using the "para" indexing type I have been able to index a small collection of public domain literature and provide a mechanism to search one or more of these texts simultaneously for things like "slave" to identify paragraphs from the collection.

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 your content does not change very often it is more a nuisance than a benefit.

Harvest is not very difficult to compile and install. It comes with a decent shell script allowing you to set 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 your 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 your 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. If everybody were to index their content and make it available via a gatherer, then a few brokers can be created collecting the content of the gatherers to produce subject- or population-specific indexes, but alas, this was a dream that came to fruition.

Ht://Dig

This is nice little indexer, but just doesn't have the features of some of the other available distributions. Configuring the application for compilation is not too tricky, but unless you set paths correctly you may create a few broken links. Like SWISH, to index your data you feed the application a 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. Not too hard.

The indexer supports Boolean queries, but not phrase searching. It can apply an automatic stemming algorithm, but upon doing so you might give the unsuspecting user information overload. 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 nice thing Ht://Dig 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 reviewed here, I might rank this one as #2 after SWISH.

Isite/Isearch

Isite/Isearch is one of the very first implementations based on the WAIS code. Like Yaz/Zebra, it is intended to support the Z39.50 information retrieval protocol. Like freeWAIS (and unlike Yaz/Zebra) 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. If you require Z39.50 compliance and for some reason Yaz/Zebra does not work for you, then give Isite/Isearch a whirl.

MPS

MPS seems to be the zippiest of the indexers reviewed here. 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 your favorite database application and have the content of your database indexed and searchable by MPS. You are 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 a 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.

SWISH

SWISH is currently my favorite indexer. Originally written by Kevin Hughes (who is also the original author of hypermail), this software is a model of simplicity. To get it to work for you all that needs to be done is to download, unpack, configure, compile, edit the configuration file, and feed the file to the application. A single binary and a single configuration file is 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 modules have been developed allowing the developer to avoid this problem, but the modules are considered beta software.

Like Harvest, SWISH can "automagically" extract the content of HTML META tags and make this content field searchable. Assume you have a META tag in the header of your HTML document such as this:

```
<META NAME="subject" CONTENT="adaptive technologies">
```

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)". You could then submit a query to SWISH such 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, other things can be done with the SWISH results; SWISH results can easily be sorted by a specific field, or more importantly, SWISH results can be marked up before they are returned. For example, if your 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."

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 index(es) 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.

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 computing before client/server computing was popularized:

> 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 are 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. -- http://lcweb.loc.gov/z3950/agency/markup/01.html

Put another way, 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 you to create Z39.50 clients and servers. Zebra is an indexer with a Z39.50 front-end. To make these tools work for you the first thing to be done is to download and compile the Yaz toolkit. Once installed you can feed documents 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, you there are at least a couple of Perl modules available from CPAN providing Z39.50 interfaces. There is also a module called ZAP! (http://www.indexdata.dk/zap/) allowing you to embed a Z39.50 client into Apache.

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 you must maintain and the formats of the files available for indexing are rather obtuse. If you require Z39.50, then this is the tool for you. If not, then something else might be better suited to your needs.

Local examples

A number of local implementations of the various indexers reviewed here have been created. Use these links to play and see how well they work:

- freeWAIS-sf (plain text files where each "record" is delimited by a blank line)
- Harvest (plain text and HTML files across the Internet)
- Ht://Dig (HTML pages containing HTML META tags)
- Isite/Isearch (HTML pages containing HTML META tags)
- MPS (plain text files on the local file system)
- SWISH (HTML pages containing HTML META tags)
- WebGlimpse (HTML pages containing HTML META tags)

Summary and information systems

Indexers provide one means for "finding a needle in a haystack" but don't rely on it to satisfy people's 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. Readability is another word for usability. 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. The browsable categories of Yahoo! are a good example. 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 a person will not find that information and may need help. Interactive assistance is that help mechanism.

By creating well-structured data you can supplement the searchability aspects of your information system. For example, if the data you have indexed is HTML, then insert META tags into your documents and use a controlled vocabulary -- a thesaurus -- to describe those documents. If you do this then you can use SWISH or Harvest to extract these tags and provide canned field searching access to your documents; freetext searches rely too much on statistical analysis and can not return as high precision/recall ratios as field searches. If your content is saved in a database, then it is an easy process to create your HTML and include META tags. Such a process is described in more detail in "Creating 'Smart' HTML pages with PHP" (http://www.infomotions.com/musings/smart-pages/).

The indexers reviewed here have different strengths and weaknesses. If your content is primarily HTML pages, then SWISH is most likely the application you would want to use. It is fast, easy to install, and since it comes with no user interface you can create your own with just about any scripting language.

If your content is not necessarily HTML files, but structured text files such database dumps, then MPS or the Yaz/Zebra combination may be something more of what you need. Both of these applications support a wide variety of file formats for indexing as well as the incorporation of standards.

Links

Here is a list of URL's pointing to the indexers reviewed in this text.

1. freeWAIS-sf - http://ls6-www.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf/
2. Harvest - http://harvest.sourceforge.net/
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://www.swish-e.org/
7. WebGlimpse - http://webglimpse.net/
8. Yaz/Zebra - http://indexdata.dk/zebra/

Author: Eric Lease Morgan <eric_morgan@infomotions.com>
Source: This article was originally written for a presentation at the O'Reilly Open Source Software conference of 2001 in San Diego, CA
Date created: 2001-05-28
Date updated: 2004-11-19
Subject(s): indexing; open source software;
XML source: http://infomotions.com/musings/opensource-indexers/indexers.xml