OpenSearch and SRU: A Continuum of Searching

Ralph LeVan

Not all library content can be exposed as HTML pages for harvesting by search engines such as Google and Yahoo!. If a library instead exposes its content through a local search interface, that content can then be found by users of metasearch engines such as A9 and Vivísimo. The functionality provided by the local search engine will affect the functionality of the metasearch engine and the findability of the library's content. This paper describes that situation and some emerging standards in the metasearch arena that choose different balance points between functionality and ease of implementation.

Editor's Note: This article was submitted in honor of the fortieth anniversaries of LITA and *ITAL*.

The content provider's dilemma

Consider the increasingly common situation in which a library wants to expose its digital content to its users. Suppose it knows that its users prefer search engines that search the contents of many sites simultaneously, rather than site-specific engines such as the one on the library's Web site. In order to support the preferences of its users, this library must make its contents accessible to search engines of the first type.

The easiest way to do this is for the library to convert its contents to HTML pages and let the harvesting search engines such as Google and Yahoo! collect those pages and provide searching on them. However, a serious problem with harvesting search engines is that they place limits on how much data they will collect from any one site. Google and Yahoo! will not harvest a 3-million-record book catalog, even if the library can figure out how to turn the catalog entries into individual Web pages.

An alternative to exposing library content to harvesting search engines as HTML pages is to provide a local search interface and let a metasearch engine combine the results of searching the library's site with the results from searching many other sites simultaneously. Users of metasearch engines get the same advantage that users of harvesting search engines get (i.e., the ability to search the contents of many sites simultaneously) plus those users get access to data that the harvesting search engines do not have. The issue for the library is determining how much functionality it must provide in its local search engine so that the metasearch engine can, in turn, provide acceptable functionality to its users. The amount of functionality that the library provides will determine which metasearch engines will be able to access the library's content.

Metasearch engines, such as A9 and Vivísimo, are search engines that take a user's query, send it to other search engines, and integrate the responses.[1] The level of integration usually depends on the metasearch engine's ability to understand the responses it receives from the various search engines it has queried. If the response is HTML intended for display on a browser, then the metasearch engine developers have to write code to parse through the HTML looking for the content. In such a case, the perceived value of the content determines the level of effort that the metasearch engine developers put into the parsing task; low-value content will have a low priority for developer time and will either suffer from poor integration or be excluded.

For metasearch engines to work, they need to know how to send a search to the local search engine and how to interpret the results. Metasearch engines such as Vivísimo and A9 have staffs of programmers who write code to translate the queries they get from users into queries that the local search engines can accept. Metasearch engines also have to develop code to convert all the responses returned by the local search engines into some common format so that those results can be combined and displayed to the user. This is tedious work that is prone to breaking when a local search engine changes how it searches or how it returns its response. The job of the metasearch engine is made much simpler if the local search engine supports a standard search interface such as SRU (Search and Retrieve URL) or OpenSearch.

What does a metasearch engine need in order to use a local search engine?

The search process consists of two basic steps. First, the search is performed. Second, records are retrieved.

To do a search, the metasearch engine needs to know:

1. The location of the local search engine
2. The form of the queries that the local search engine expects
3. How to send the query to the local search engine

To retrieve records, the metasearch engine needs to know:

4. How to find the records in the response
5. How to parse the records

Ralph LeVan (levan@oclc.org) is a Research Scientist at OCLC Online Computer Library Center in Dublin, Ohio.

Four protocols

This paper will discuss four search protocols: OpenSearch, OpenSearch 1.1, SRU, and the MetaSearch XML Gateway (MXG).[2]

OpenSearch was initially developed for the A9 metasearch engine. It provides a mechanism for content providers to notify A9 of their content. It also allows RSS (Really Simple Syndication) browsers to display the results of a search.[3]

OpenSearch 1.1 has just been released. It extends the original specification based on input from a number of organizations, Microsoft being prominent among them.

SRU was developed by the Z39.50 community.[4] Recognizing that their standard (now eighteen years old) needed updating, they simplified it and created a new Web service based on an XML encoding carried over HTTP.

The MXG protocol is the product of the NISO MetaSearch Initiative, a committee of metasearch engine developers, content providers, and users.[5] MXG uses SRU as a starting place, but eases the requirement for support of a standard query grammar.

Functionality versus ease of implementation

A library rarely has software developers. The library's area of expertise is, first of all, the management of content and, secondarily, content creation. Librarians use tools developed by other organizations to provide access to their content. These tools include the library's OPAC, the software provided to search any licensed content, and the software necessary to build, maintain, and access local digital repositories.

For a library, ease of adoption of a new search protocol is essential. If support for the search protocol is built into the library's tools, then the library will use it. If a small piece of code can be written to convert the library's existing tools to support the new protocol, the library may do that. Similarly, the developers of the library's tools will want to expend the minimum effort to support a new search protocol.

The tool developer's choice of search protocol to support will depend on the tension between the functionality needed and the level of effort that must be expended to provide and maintain it. If low functionality is acceptable, then a small development effort may be acceptable. High functionality will require a greater level of effort.

The developers of the search protocols examined here recognize this tension and are modifying their protocols to make them easier to implement. The new OpenSearch 1.1 will make it easier for some local search-engine providers to implement by easing some of the functionality requirements of version 1.0. Similarly, the NISO Metasearch Committee has defined MXG, a variant of SRU that eases some of the requirements of SRU.[6]

Search protocol basics

Once again, the five basic pieces of information that a metasearch engine needs in order to communicate effectively with a local search engine are: (1) local search engine location, (2) the query-grammar expected, (3) the request encoding, (4) the response encoding, and (5) the record encoding. The four protocols provide these pieces of information to one degree or another (see table 1).

The four protocols expose a site's searching functionality and return responses in a standard format. All of these protocols have some common properties. They expect that the content provider will have a description record that describes the search service. All of these services send searches via HTTP as simple URLs, and the responses are sent back as structured XML. To ease implementation, OpenSearch 1.1 allows the content provider to return HTML instead of XML.

All four protocols use a description record to describe the local search engine. The OpenSearch protocols define what a description record looks like, but not how it is retrieved. The location of the description record is discovered by some means outside the protocol (a priori knowledge). The description record specifies the location of the local search engine. The SRU protocols define what a description record looks like and specifies that it can be obtained from the local search engine. The location of the local search engine is provided by a means outside the protocol (a priori knowledge again).

Each protocol defines how to formulate the search URL. OpenSearch does this by having the local search-engine provider supply a template of the URL in the description record. SRU does this by defining the URL.

OpenSearch and MXG do not define how to formulate the query. The metasearch engine can either pass the user's query along to the local search engine unchanged or reformulate the query based on information about the local search engine's query language that it has gotten by outside means (more a priori knowledge). In the first case, the metasearch engine has to hope that some magic will happen and the local search engine will do something useful with the query. In the latter case, the metasearch engine's staff has to develop a query translator.

SRU specifies a standard query grammar: CQL (Common Query Language).[7] This means that the metasearch engine only has to write one translator for all the SRU local search engines in the world. But it also means that all the SRU local search engines have to support the CQL query grammar. Since there are no local search engines that support CQL as their native query grammar, the content provider is left with the task of translating CQL queries into their native query grammar. The query translation task has moved from the metasearch engine to the content provider.

OpenSearch 1.0, MXG, and SRU define the structure of the query response. In the case of OpenSearch, the response is returned as an RSS message, with a couple of extra elements added. MXG and SRU define an XML schema for their responses.

OpenSearch 1.1 allows the local search engine to return the response as unstructured HTML. This moves the requirement of creating a standard response from the content provider and leaves the metasearch engine with the much tougher task of finding the content embedded in HTML. If the metasearch engine doesn't write code to parse the response, then all it can do is display the response. It will not be able to combine the response from the local search engine with the responses from other engines.

SRU and MXG require that records be returned in XML and that the local search engine must specify the schema for those records in the response. This leaves the content provider with the task of formatting the records according to the schema of their choice, a task that the content provider is probably best able to do. In turn, the metasearch engine can convert the returned records into some common format so that the records from multiple local search engines can be combined into a single response. Because the records are encoded in XML, it is assumed that standard XML formatting tools can be used for the conversion.

OpenSearch does not define how records should be structured. The OpenSearch response has a place for the title of the record and a URL that points to the record. The structure of the record is undefined. This leaves the metasearch engine with the task of parsing the record that is returned. Again, the effort moves from the content provider to the metasearch engine. If the metasearch engine does not or cannot parse the records, then it can at least display the records in some context, but it cannot combine them with the records from another local search engine.

Table 1. Comparison of requirements of four metasearch protocols for effective communication with local search engines

Protocol Feature	OpenSearch 1.1	OpenSearch 1.0	MXG	SRU
Local search engine location	A priori	A priori	A priori	A priori
Request encoding	Defined	Defined	Defined	Defined
Response encoding	None	RSS	XML	XML
Record encoding	None	None	XML	XML
Query grammar	None	None	None	CQL

Conclusion

These protocols sit on a spectrum of complexity, trading the content provider's complexity for that of the search engine. However, with lessened complexity for the metasearch engine comes increased functionality for the user. Metasearch engines have to choose what content providers they will search. Those that provide a high level of functionality can be easily combined with their existing local search engines. Content providers with a lower level of functionality will either need additional development by the metasearch engine or will not be searched. Not all metasearch engines require the same level of functionality, nor will they be prepared to accept content with a low level of functionality. Content providers, such as digital libraries and institutional repositories, will have to choose the functionality they need to support to reach the metasearch engines they desire.

References and notes

1. Joe Barker, "Meta-Search Engines," in *Finding Information on the Internet: A Tutorial* (U.C. Berkeley: Teaching Library Internet Workshops, Aug. 23, 2005 [last update]), www.lib.berkeley.edu/TeachingLib/Guides/Internet/MetaSearch.html (accessed May 8, 2006).

2. A9.com, "OpenSearch Specification," http://opensearch.a9.com/spec/ (accessed May 8, 2006); A9.com, "OpenSearch 1.1," http://opensearch.a9.com/spec/1.1/ (accessed May 8, 2006).

3. Mark Pilgrim, "What is RSS?" O'Reilly XML.com, Dec. 18, 2002, www.xml.com/pub/a/2002/12/18/dive-into-xml.html (accessed May 8, 2006).

4. The Library of Congress Network Development and MARC Standards Office, "Z39.50 Maintenance Agency Page," www.loc.gov/z3950/agency/ (accessed May 8, 2006).

5. National Information Standards Organization, "NISO MetaSearch Initiative," www.niso.org/committees/MS_initiative.html (accessed May 8, 2006).

6. NISO Metasearch Initiative Task Group 3, "NISO MetaSearch XML Gateway Implementors Guide, Version 0.2," May 16, 2005, [Microsoft Word Document] www.lib.ncsu.edu/niso-mi/images/0/06/NISO_Metasearch_Initiative_XML _Gateway _Implementors_Guide.doc (accessed May 8, 2006); The Library of Congress, "SRU: Search and Retrieve via URL; SRU Version 1.1 13 February 2004," www.loc.gov/standards/sru/index.html (accessed May 8, 2006).

7. The Library of Congress, "Common Query Language; CQL Version 1.1 13th February 2004." [Web page] www.loc.gov/standards/sru/cql/index.html (accessed May 8, 2006).