You Too Can Build a Better Search Tool

IT'S 2008, AND IT'S NEVER BEEN EASIER TO MEET YOUR USERS' NEEDS WITH A REALLY GREAT SEARCH TOOL.

by DANIEL CHUDNOV

I've written four search interfaces of note during my career. One, the "jake" journal product database, is now defunct, but it was successful despite its ultimate demise. Search interfaces for the unalog bookmarking site (http://unalog.com) and for the Canary Database (http://canarydatabase.org) are still alive and kicking and good enough to suit their purposes. The fourth, for a project still incubating at work, is more complicated than any of the others, but it isn't even intended to be used much in the typical "people type words into the box" sort of way. It meets its project's requirements well enough. If you told me in 1995, when Joe Janes was teaching us at the University of Michigan how to search Dialog, that during the next 13 years I'd write four custom, somewhat-successful search interfaces myself, I might not have believed you. I certainly wouldn't have known where to start. So let me be the first to tell you—you too can build suitable search interfaces yourself, and you probably will.

What Are You Searching For?

It might sound silly, but this is the most important question to answer if you need your own search interface. In 1998, I was working on a medical library reference desk, and the question we were asked most often was, "Where can I get this journal article?" Sometimes the people were working off a list of references, sometimes just a jotted-down title abbreviation with a volume, issue, and page. Back then, believe it or not, lots of people would still come into the library a few times a week on the days when they knew the new issues of their preferred titles would arrive. If somebody else was reading the latest issue off the shelf already, they'd want to know if it was online too. (Back then only some of the journals were online.) They'd approach the desk with those references, and the first thing we had to do was to chop the references up. What title did they mean? Did we have it? Was it online? Was the paper copy bound, and was the volume on the shelf, or maybe at the bindery? All of these questions were incidental to the users' question. What we really needed was a search interface that let people ask the question directly: "Where can I get this journal article, with this title, this volume, this issue, and this page?" Even though the main success of the jake database was in helping people understand what they would need from link resolvers, its usefulness to normal people was apparent from the start. To this day people tell me they miss that simple interface. It was still useful even after the data was long out-of-date. The database design was simplistic and the user interface was far from fancy, but it was exactly suited to the purpose it served. That made it work for people all over.

The search interface for unalog, the bookmarking site, was an early feature that preceded user tagging. When these applications were new, I thought that indexing page title words, URL or domain name fragments, and information about the date I saved it (e.g., "last month") would suffice, but after a while it was clear that people wanted to be able to ask the system questions such as, "It was by somebody named Joan and it was about knitting." In that case you want to search for **joan knitting** and find just the link you meant. But if the words "Joan" and "knitting" weren't in the title or the link itself, you were out of luck. Once again, the answer might sound simple, but by just grabbing the content of the page itself, and indexing that too (after stripping out HTML tags and script content), we got the parts that said "Joan" and "knitting" into the index. That query would return the right hits. After a while, users wanted tagging, so we added that. For me, the page indexing has always been more useful than the tagging because I know I can type just what I mean into the box and find what I want—tags and dates then become a useful way to filter after the first search result. Tags and searches both are also great ways to project feeds out to other sites, which unalog does with OpenSearch, a "feed for search results" that I wrote about last year in the standards issue.

The search interface for the Canary Database was more difficult to get "right." The site uses MEDLINE indexing data from PubMed, along with additional study classification and indexing performed by project curators. At first I thought that by just "putting the indexing out there," literally just indexing all the fielded data we had available in the records—the regular citation data such as author, journal title, article title, abstracts, and subject headings, along with our curated values such as exposures, outcomes, species, and locations—it would all "just work." We even cross-referenced alternate species names and subject headings so **dog**, **dogs**, and **canis familiaris**, for example, would all work. But our initial usability testing proved that it didn't "just work" for users. I exposed all those values, but much like with older library systems, I made a user "learn how" to ask the system for them. For example, you couldn't just say **dogs** if you wanted to find studies about dogs. Instead you had to ask for **species:dogs**. Also, people had already grown used to the implicit AND function from using popular search systems, but I used an implicit OR, which meant that a search for **species:dogs exposure:smoke** would return a lot of hits about smoke or dogs but not the combination. Our usability results also indicated the following:

- Nobody would search **species:dogs AND exposure:smoke**. They would just search for **dogs smoke** again and again even if you told them it wouldn't work!
- These were medical researchers who knew PubMed, so they didn't know our field syntax, but they did know **smith [auth]**.
- We also tested with librarians who wanted to use **smith.au.** as their search term.
- Even though the system could handle complex Boolean searches and users understood they needed to carefully compose complex searches, sometimes it was too much work to do, and nobody did it.
- Ultimately, we had a lot of value-added data that users were never finding.

One response to these results might have been "Users are stupid!" Instead, the response we chose was "Our search interface isn't smart enough!" We knew our data were good, so we decided to work harder to make the interface better. Here are some of the changes we made:

- Switched to implicit AND
- Added both PubMed style and BRS style fielded search support (I had to write a parser for each, but it wasn't too hard. I'd read an article about somebody else doing something similar so I knew it was possible—write me for details if you want to know more.)
- Boosted nonfielded keyword matches on fielded term values (This made a keyword hit for **dogs** more important if **dogs** was indexed as a species than if it was just a term in an abstract. Most people don't use the fields at first, so this was very helpful.)
- Added guided navigation (facets with AND/OR links) for the most important fields for our users and our data: exposures, outcomes, species, and author names

As a result of these changes, our next round of usability testing went much more smoothly. Searches for **dogs smoke** found the articles they hadn't before about dogs exposed to smoke, and not a lot of articles about just dogs or just smoke. The faceted combination links didn't always make sense to the users at first, but as soon as they tried clicking these links, they got it and saw that they could build up complex queries quickly without having to compose the syntax by hand. We found other problems with the site to work on, but the search worked much better, and we got our users to our data the way they wanted to get to it.

Before I summarize a few things I've learned along the way, I'd invite you to try out the search functions on http://unalog.com and http://canary database.org. Let me know how well they work for you!

»

AFTER A WHILE, USERS WANTED TAGGING, SO WE ADDED THAT.

Basic Lessons

When I started building my first search interface, I didn't know software very well. I could put a small database together and write a little PHP and SQL to query it, and that was about it. Working with my colleagues, we figured out what the interface needed to do, and I futzed with the data and the code until it got closer and closer to something we could use at the reference desk. It had its flaws, but after a while we were comfortable enough with it to tell our users and then other libraries about it. It never did all the things we wanted it to (and to this day resolver databases aren't used for half of the functions we had in mind), but it helped us to solve the biggest problem first. This is the first lesson I'd emphasize: If you have a clear understanding of the most important questions your users need to ask of your data, you'll know what work to do so they can use your system to answer their questions.

Fast-forward from 1998 to 2003, and the Free Software (aka FLOSS) movement I wrote about a few months ago had taken off. The Lucene information retrieval software library had taken off and made it easier for someone like me to implement a sophisticated search interface, and that's what I used for unalog and the Canary Database. There are many other retrieval toolkits available to choose from now, but what made Lucene so compelling then (and still today) is its simple internal model. In Lucene, every item you want your users to find is a "Document," and all of the data associated with that item that you might want to index to help people find the item is indexed by assigning it to one or more "Fields." There's more to it, of course, but it really is mainly about figuring out how you're going to define your documents and their fields. You might split metadata values into one Field per metadata field, and you might chunk up free text into another Field, or you might do both, but the flexibility is yours to use to choose how best to serve your users' needs.

This leads me to the next lesson, one I learned from Cindy Cunningham, who spent years supporting search systems for Amazon.com and Corbis and recently joined OCLC to work on WorldCat. It used to be (in the 1970s, 1980s, and early 1990s) that we could put any kind of interface up over our data and expect our users to learn how to use it, however obtuse. We haven't been able to get away with that for 10 years, though. Now it's up to us to build everything we know into our access systems so users can find all the good stuff without jumping through hoops to get there. If you've got useful metadata that's not being indexed, get it indexed. If you've got synonyms that could aid with keyword matches, cross-reference them. If these aren't being done automatically for people, make it happen behind the scenes so you don't have to sacrifice that interface directed at the questions users have. We've invested far too much time and money in cataloging and indexing over the years to not get every ounce of value we possibly can out of that data. Help make our data work harder!

Five years after Lucene made it possible for people like me to build useful search interfaces, there are now many more options for building your own. I'd particularly encourage you to look at Zebra from IndexData (www.indexdata.dk/zebra), a very powerful tool to have in your toolkit. And in the meantime Lucene has improved in many ways, and now the Solr application from Apache (lucene.apache.org/solr) makes it even easier to use Lucene. Solr is a stand-alone web application that wraps up everything Lucene does for you into a nice, easy-to-use bundle. You still have to think about your Documents and their Fields and several more important details, but once you have that sorted out, Solr makes it easier than ever to put Lucene to work for your application. It's what we're using for that incubating new site I mentioned earlier, and it's working so well I want to tell you all about it. But we're not done and it's not public, so I can't yet. Suffice it to say, there's a reason why new OPAC replacements such as VuFind and Blacklight are using Solr—it's terrific.

WE KNEW OUR DATA WAS GOOD, SO WE DECIDED TO WORK HARDER TO MAKE THE INTERFACE BETTER.

It's 2008, and it's never been easier to meet your users' needs with a really great search tool. The last lesson to remember is to trust your users. When you try something new, give it to them as soon as you can, and let them show you whether you've solved their problems or not. They'll tell you when you've got more work left to do, and when you get it right, they'll find something else to worry you about, and you'll want to thank them for it. ■

Daniel Chudnov is a librarian working as an information technology specialist in the Office of Strategic Initiatives at the Library of Congress and a frequent speaker, writer, and consultant in the area of software and service innovation in libraries. Previously, he worked on the DSpace project at MIT Libraries and the jake metadata service at the Yale Medical Library. His email address is daniel.chudnov@gmail.com, and his blog is at http://onebiglibrary.net.