Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

KARIN HERM SIBYLLE VOLZ

The Library Search Engine – A Smart Solution for Integrating Resources Beyond Library Holdings

Gefördert
von der Senatsverwaltung für Bildung, Wissenschaft und Forschung des Landes Berlin,
vom Ministerium für Wissenschaft, Forschung und Kultur des Landes Brandenburg
und von den Mitgliedsbibliotheken des KOBV

ZIB-Report 08-37 (September 2008)

The Library Search Engine – A Smart Solution for Integrating Resources Beyond Library Holdings

Karin Herm Sibylle Volz

September 2008

Abstract

The Cooperative Library Network Berlin-Brandenburg (KOBV), Germany, addresses the problem of how to integrate resources found outside the library and library holdings into a single discovery tool. It presents a solution that uses open source technology to develop a next-generation catalog interface called the Library Search Engine. This pilot project was launched in 2007 with the library of Albert Einstein Science Park, Potsdam. The idea was to design and develop a fast and convenient search tool, integrating local holdings (books, journals, journal articles) as well as relevant scientific subject information such as open access publications and bibliographies.

1 Introduction

With the following article, we want to share experiences from a pilot project that started in 2007, reached production level that same year, and is still developing further. Within this project we used search engine technology to search both catalog data and external subject relevant information and developed the *Library Search Engine (LSE)*.

The library technology world is changing quickly driven by the fast alterations in the information technology sector. This trend has intensified since the triumph of search engine technology. Quite heterogeneous projects implementing these new technologies can be subsumed under the slogan "working on the next-generation catalog." In the area of non-commercial or open-source software solutions we find well-known projects such as Vufind, the Danish Summa project, and last but not least our new LSE.

The Cooperative Library Network Berlin-Brandenburg (KOBV)[1] is one of the six existing library networks in Germany. The KOBV offers IT services and other services to libraries in the German states of Berlin and Brandenburg. Currently, we have more than 70 member libraries, ranging from academic to special and public libraries. One of our key aspects of activity is the development of new services. Two small development teams handle one or more projects. They consist of one or two technical developer(s) and one librarian. The team responsible for driving the Library Search Engine project had a strong background in implementing search engine technology, gained from an earlier project. In order to make this know-how available for our member libraries the Library Search Engine pilot project was launched in April 2007.

[1]For more information please visit: http://www.kobv.de

2 Background and Main Goal

Library patrons typically need access to both library holdings and other subject specific information resources, e.g., open access publications. They are usually comfortable using different search tools, but have difficulties in identifying and locating the appropriate resources. Furthermore, it is a well-known issue that patrons are neither satisfied with interface and retrieval features of the search component of traditional online catalogs, nor with the presentation of search results. Additionally, scientists and academics want fast and easy access to as many relevant full-text publications as possible.

We set ourselves the following goal: We didn't want to reinvent the online public-access catalog (OPAC) or substitute the integrated library system (ILS), since we were quite satisfied with its circulation module, the user sign-in, and the personalization features. Instead, we focused on creating a tool to compensate for the unsatisfactory search and retrieval components of the online catalog. That was where we identified the most urgent need for action. Our ambition was to expand the search space and to process metadata and – where technically and legally possible – full-text from both the library's own collections and external resources, indexing them by using search engine technology and presenting them in one search engine application.

Our pilot library, the library of Albert Einstein Science Park, Potsdam, is a joint library of three research institutes. It offers services to geoscientists and climate and polar researchers. This library was unsatisfied with the search component of their OPAC (self-knitted, the ILS itself is an OCLC SISIS-SunRise system), which they wanted to replace, and thus enthusiastically signed up as the pilot library. In our first meetings with the head librarian and one of his staff members, we agreed to design and develop a fast and convenient search tool, integrating local holdings (books and journals) and relevant scientific subject information such as open access publications or metadata from bibliographies that were compiled during research programs and not yet publicly accessible.

3 Approach

Our project team mainly consisted of four people: the head librarian of the pilot library, one of his staff members, who had both a strong IT background and experience in the library field, and this article's two authors from the Cooperative Library Network (KOBV). Also, the implementation was supported by a student and other library staff tested the application. As mentioned above, we wanted to develop a fast and convenient search tool that integrates data from a variety of sources, but of course none of us had a clear-cut idea of what this should look like. Thus, our approach was to work with a rapid prototyping *ansatz*, i.e., fast development of a prototype with certain functionality and certain "mock-up" functions to quickly get a look and feel of the application and visualize the data at hand.

This approach proved even more valuable later on because we were able to respond with flexibility to the needs of our pilot library.[2] The rapid prototyping approach was also useful with regard to the choice of data sources: we started out with six and added more and more sources during development finally reaching ten. We were able to verify the scalability of the design during development as these sources could be added without major increase of implementation time.

4 Solution

This section describes the data we included, gives an outline of the implementation, and depicts the flow of data from the library to the search engine user. In addition, we explain the user interface of the resulting application.

4.1 Data

The LSE handles various heterogeneous data sources:

- Records of two library catalogs,
- Journal title records of all licensed journals (electronic and print),
- Data from two institutional publication databases,
- Metadata of open access articles from an open access portal focusing on geology and related sciences called Geo-Leo,[3]
- A primary data collection, and
- Content of bibliographical information databases.

The current article metadata from TOCs of the library's e-journal subscriptions and an e-book collection are preprocessed and provided as XML by the library, which receives the content via RSS feeds and email. One common feature of all data is the basic format: everything is provided in some form of XML, but the "dialects" vary:

- MABxml[4]
- Scientific primary data[5]

[2] For example, the implementation of the A-Z journal list: a few weeks after the launch of ALBERT (Library Search Engine (LSE) during the development phase), many library user complained, because they missed the alphabetical list of all licensed journals, which used to be supplied as a static web-page. We could easily supply the wanted list dynamically employing search technology.

[3] http://www.geo-leo.de/geoleo/www-docs/

[4] See definition: http://www.d-nb.de/standardisierung/pdf/mabxml_1_dok.pdf

[5] http://www.std-doi.de/

- Journal metadata from a journal database[6]
- Publication servers: harvested via OAI-PMH using metadata format oai-dc
- Many variation of oai-dc (RSS feeds, e-books, Geo-Leo data, bibliographical information from different sources), and
- Full-text wherever available and accessible.

Quite a few sources in the above list use the oai-dc metadata schema[7], which is a very open definition, i.e., each of the defined fields can occur multiple times or not at all and their content is pretty much arbitrary. Therefore, it was necessary to define different parsing structures for each of the supplied XML data sources, but the underlying parser for the oai-dc schema of course is the same.

4.2 Implementation

We decided to build the project's application using free and stable open source technology wherever possible in order to reduce implementation cost. In fact, commercial solutions were out of the question due to the non-existing budget. This led to a Java application based on Apache Tomcat and the search engine library Apache Lucene. Other open source "ingredients" are: Ant, Log4J, Apache Commons (e.g. digester, utilities), PDFBox, and many more.

To present the different data types in a user interface and to make the metadata searchable we had to normalize our data. As we "simply" wanted to provide a search engine, we could limit the data stored to the fields we wanted to search and present. Hence, it was possible to work without a database, which also reduced the demands of data quality (e.g., consistency, completeness, etc.). Thus, the library could focus more on content than on metadata quality, and we could avoid implementation of components for verifying and correcting data.

The normalization of the data was defined according to the search and the user interface, i.e., the librarians on our team defined the mapping for each of the data types to searchable fields, export features, and display fields. These fields and a few additional technical fields (e.g., identifiers for accessing the ILS) are stored in the search index. Technically, the application can handle any (reasonable) number of Lucene indexes (see Fig. 1 and Fig. 2). The question of which and how many is just a matter of configuration. This makes it easier to keep the data up-to-date or to add new sources.

[6] http://www.allegro-c.de
[7] http://www.openarchives.org/OAI/2.0/oai_dc.xsd

```
...

# listing of all index directories (1-n) that should be accessed by the
# search engine
#
indexDirectory1=/pathelement1/index1
indexDirectory2=/pathelement2/index2
indexDirectory3=/pathelement3/index3
indexDirectory4=/pathelement4/index4

...
```

Figure 1: Sample from the configuration file

```
...
try {
    IndexReader[] ir = new IndexReader[count];
    IndexSearcher[] is = new IndexSearcher[count];
    int fehler = 0;
    for (int i=0; i < count; i++){
        try{
            ir[i-fehler]= IndexReader.open(conf.getString("indexDirectory"+(1+i)));
            is[i-fehler] = new IndexSearcher(ir[i-fehler]);
        }catch(Exception ie){
            fehler +=1;
            log.error("index zugriff fehlgeschlagen (count="+i+" fehlerzahl="+fehler+") "+ ie.getMessage());
        }
    }
    IndexReader[] irBereinigt = new IndexReader[count-fehler];
    IndexSearcher[] isBereinigt = new IndexSearcher[count-fehler];
    for(int i=0; i < count-fehler; i++){
        irBereinigt[i]=ir[i];
        isBereinigt[i] = is[i];
    }
    log.debug("erzeuge reader und searcher.");
    reader = new MultiReader(irBereinigt);
    searcher = new MultiSearcher(isBereinigt);
}catch (Exception e) {
    log.error("index zugriff fehlgeschlagen: "+ e.getMessage(), e);
}

...
```

Figure 2: Code sample demonstrating the use of Lucene's MultiReader and Searcher

During the development phase the library identified the need to fine tune the search results ranking. As the number of the external sources exceeded the number of library catalog records tremendously, the library records were hardly visible in the result lists. But, of course, they contain the most relevant information. We adjusted the ranking using the standard Lucene boosting functionalities so that – still depending on their relevance to the user's search – library records receive a higher ranking than external sources.

We end this section with a few numbers about the application:

- Currently 265,000 records, increasing with every update
- Indexed full-text resources: 3,500 and increasing with every update
- Amount of data: 3 GB increasing
- Search time well below a second – not increasing
- Lines of code: 18,500.

4.3 Dataflow

How the (relevant) data reaches the user (see Fig. 3):

1. The library identifies the data collections to be included in the Library Search Engine and adapts or builds XML files according to the specification we defined during development. For instance, they export catalog records from their library catalog and convert these to the appropriate format – MABxml using a tool provided by German National Library[8], or they produce oai-dc records from e-book email alerts they receive from the publisher. Most of the data is regularly updated via FTP or harvested online from the publication servers. Other data doesn't change that frequently or not at all, e.g., the publication databases or the contents from Geo-Leo (approximately twice a year).

2. Once the data becomes available to the LSE it is normalized and processed resulting in a Lucene search index.

3. Searches entered through the web interface, features are processed with Lucene search functionalities, and results of the relevant data are retrieved from the index.

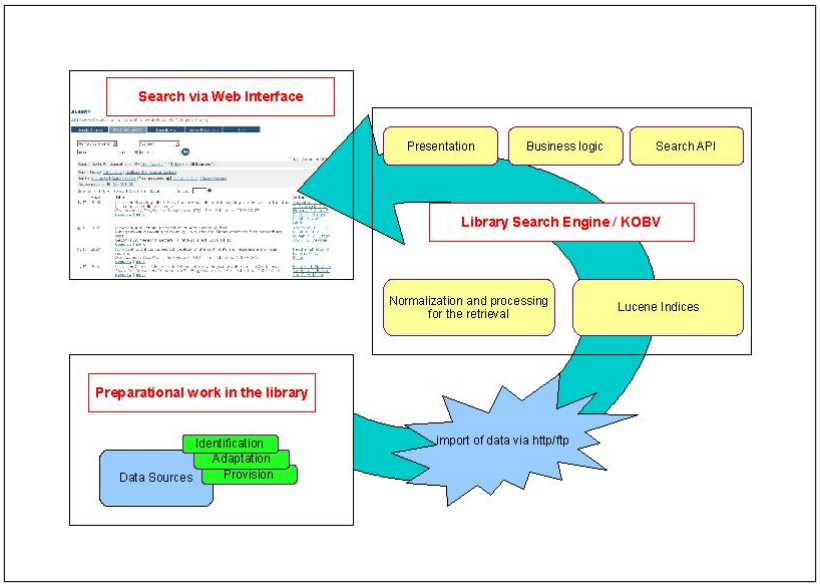

Figure 3: Processing of data information

4.4 Search Features and Graphical User Interface

After having thrown a short glance at the backend procedures, let's have a look at the web interface functionalities (see Fig. 4): The Library Search Engine offers common

[8]http://www.d-nb.de/eng/index.htm; MABxml tool: ftp://ftp.ddb.de/pub/tools/mab/

search engine features such as "Did you mean" functionality, relevance ranking (with a boosting of local holdings), fast retrieval, intuitive interface, and stable performance. The Simple Search is the default search option, but we also offer Advanced Search and an A-Z list of all journal holdings. We tried to simplify navigation, which means for instance that we offer export and mailing of search results directly from the results list so that it is not necessary to save results first to a saved results list, which we nevertheless still offer.

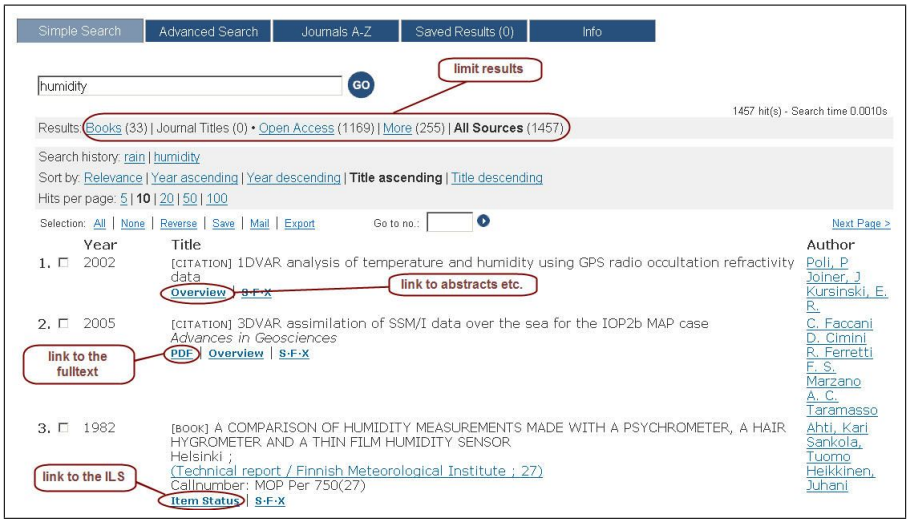

Figure 4: Web interface of the Library Search Engine

Another feature is the linking of author names and series titles within the search results list. The activation of these links automatically starts a new search with the selected author name/series title. Also, results from records of the two library catalogs link back to the catalogs for status information and user account functionalities. If the indexed metadata contains links to full-text publications, abstracts, or journal websites, we index these full-text resources and link directly to the full-text and/or related websites.

When starting the project, we considered offering within the Advanced Search the possibility of limiting the number of searched data sources *before* starting a search. In the course of further development, we changed our minds, since search time is fast enough even if we always perform a search over the entire index. Therefore, we decided to offer the possibility to partition the results *after* having executed a search. We assembled our data sources into four different result groups:

- Books (data from the library catalogs),
- Journal titles (journals held by the library),
- Open access (open access publications and primary data), and
- More (current contents, e-books, publication databases, bibliographies).

4.5 Going Live

Going live with the Library Search Engine was short and sweet. Without any announcement, the library's OPAC search component was closed and replaced with the LSE as the unique access point for searching the library's media and all additional resources.[9]

User feedback was astonishing: either there were no remarks at all or they mentioned that they had long awaited a tool like this! In some ways, users took for granted not only using the new web interface but also finding catalog records and external information together in one application. The library carried out a contest among their users to find their own specific name for the new search tool soon after the implementation. "Marvelous" was one of the proposals which was handed in – this tells its own story. However, the library chose the more descriptive name "ALBERT – All Library Books, journals and Electronic Records Telegrafenberg". ("Telegrafenberg" is where the library is located.)

5 Conclusion

Patron and staff feedback have shown that we are on the right track. How would our other member libraries react to the new search tool? We conducted a workshop where we explained in detail the functionalities and usage possibilities of the Library Search Engine. Afterwards, six libraries stated that they were interested in implementing a local version of the LSE for a central or alternative access point to their catalog records and other data collections. But thorough analysis of the responses revealed that the libraries are actually seeking *the one* access point to their complete collection, including licensed databases.

We all know this is something libraries and their patrons want, however, it is something we cannot offer. Unlike present commercial products such as Primo by Ex Libris or SirsiDynix's Enterprise Portal Solution, we don't include a distributed search module. The Library Search Engine needs to obtain the metadata and/or full-text materials to build the index(es), which might be a problem with licensed resources. Libraries have already started asking for metadata, and especially with metadata from backfiles of licensed e-journals this has been successful. Nevertheless, this is something each library or library consortium has to deal with. The KOBV is only a service provider and doesn't hold any licensed material itself. That is why we cannot clarify legal and license questions.

The complexity of this implementation presented another challenge. By realizing this project, we demonstrated that it is possible to develop and implement this new search tool for libraries. Deployment of open source technology has ensured cost limitation. However, the further procedure is undetermined, because we are unsure that the effort to implement the application for another library is really justifiable, considering our small-sized team of two people. With the pilot project, we wanted to create a standard application that could easily be implemented for other libraries. But we are very aware

[9] Visit http://waesearch.kobv.de to get a firsthand impression!

that these libraries will have other requirements. For instance, an academic library with different branch libraries has different needs than our singular pilot library. This means that the development and implementation of the Library Search Engine for another library is still far from being a standard procedure and is therefore work intensive. Furthermore, there are features that have to be improved, for instance update procedures, or to be expanded, for instance reusing options through open-sourcing. We are currently analyzing our situation and seeking both national and international cooperation and exchange. We would be pleased to receive feedback from libraries or institutions who are involved in similar projects.