Design of a Search Engine for Cross-Library Search Based on Metalogy Metadata

Ing-Xiang Chen, Che-Min Chen, and Cheng-Zen Yang

Department of Computer Engineering and Science
Yuan Ze University
Chungli, Taiwan, R. O. C.

Abstract: Searching information across digital libraries is important for digital library utilization. With such functionality, users can find related information in a more comprehensive manner. In this paper, we present the design of a search engine providing a uniform user interface for cross-library searching. The design of the search engine is based on a XML/DTD mapping approach to facilitate cross-library search. The heterogeneous metadata of remote digital libraries are mapped to a default metadata set based on the Dublin Core set. Therefore, high system extensibility can be achieved and the search engine will be more efficient.

Keywords: Cross-Library Search, Search Engine Design, Metalogy, Metadata, XML/DTD.

1. Introduction

In the construction of a digital library, metadata play a very important role for data organization and manipulation. However, managing metadata is not an easy task because it may require specific domain knowledge for appropriate data categorization and it has to deal with the complicated relationships between the metadata items. Management tools are needed for easing metadata construction and manipulation. Metalogy is such a management system developed by ROSS (Resource organization and Searching Specification) [1] project group in Taiwan. With Metalogy, the metadata structure of a digital library can be easily maintained and customized according to the library’s need. As a digital library is under construction, the library manager can also easily change the metadata organization and create new qualifiers for necessary data items. After creating the metadata structures, Metalogy can be also used to input digitized items, query the library information, export/import XML (eXtensible Markup Language) [2,3] records, and manage user permissions.

However, Metalogy is mainly designed for metadata management of a correspondent digital library. It does not consider how to search information across several digital libraries. But we notice that as digital libraries are widely deployed, searching information across several digital libraries becomes more important for users, because that each digital library may have its own archiving themes maintained by specialists with different domain knowledge. For example, a user may want to find information about Van Gogh. He/she may need to separately navigate a digital library about artists and another digital library about painting.
Therefore, we consider developing a search engine to search information across different digital library sources. With the search engine, users do not need to know in advance where the digital libraries are. They can also avoid repeatedly connecting to different digital libraries and input the same query terms. The search engine provides an integrated view of several digital libraries. Furthermore, the search engine handles the metadata provided by the digital libraries. Through this approach, users can find numerous information items of different categories from different digital libraries; no matter in which category the query term appears.

One of the distinct design feature of the search engine is that its development is based the XML/Metalogy technology instead of a particular information retrieval protocols such as Z39.50 [4] for two following main reasons. First, XML/Metalogy provides comprehensive metadata descriptions and DTD (Document Type Definition) [3] information for metadata search. This approach is very flexible for searching information from different specific metadata contexts. If a new metadata specification is developed for some specific domain, this specification can be easily integrated in the XML/Metalogy search engine. Second, most of the contemporary information retrieval protocols such as Z39.50 are complicated and are mainly designed for library specialists. Most of the normal users do not know the details of the protocol and have no idea about how to specify the query terms.

Our search engine design simplifies the query interface design. In addition, the search engine design provides good extensibility for future metadata development through XML technology. We have also implemented a prototype with Java. The prototype maintains an intermediate database for speeding up search purpose. The database is organized according to the Dublin Core set (DC) [5,6] because of its high extensibility. Though the prototype currently provides primitive query functions, it shows that searching information across distributed digital libraries can be easily achieved.

The rest of the paper is organized as follows. The next section describes previous related work for XML/metadata search engine design. Section 3 describes the design of our search engine. Section 4 shows a prototype implementation and a query example. Section 5 discusses the future work and concludes the paper.

2. Related Research Work

Searching information across distributed digital libraries is a very important issue for the success of digital library utilization. If such services are provided, users can get more comprehensive related information from different digital archives. Therefore, this need has been noticed from a long time. In 1995, the Z39.50 (Version 3) information retrieval service protocol was proposed to search and retrieve information from heterogeneous databases over networks [4]. It is a powerful protocol and provides abstract search capability. Therefore, users can proceed queries without knowing the underlying database structures. However, such a powerful protocol has a drawback that it is difficult to be implemented because of its complicated functionality [7].

In 1999, a meeting was held in Santa Fe to discuss the cooperation among scholarly e-print archives. Then the Open Archives Initiative (OAI) was established for building an interoperability framework across digital archives with a low barrier [8,9]. An OAI service provider, Arc, is developed for cross-archive searching [10]. The architecture of our search engine is very similar to
Arc. Only a few differences exist between Arc’s harvester and our XML/DTD manager design. In Arc’s harvester design, the OAI protocol is used to harvest digital archives. In our design, we do not consider a specific protocol and provide a manually controlled interface for metadata mapping administration.

In 1999, Lin and Lu developed a distributed query system of a digital library called Harp [11]. With Harp, legacy public libraries and structured databases can be queried through HarpSQL, and SQL-like query language for querying legacy library catalogs. Harp provides a uniform query interface across legacy public libraries, but the architecture design of Harp is different to our search engine architecture design in the query/search manager. In Harp, a HarpSQL server acts as a query agent for storing and handling the intermediate query results. It does not act as a search engine to collect and store all metadata from digital libraries and does not consider searching on metadata.

METALICA [12] adopts a metasearch engine approach like MetaCrawler [13] to provide a uniform user interface for supporting cross-archive search functionality. In METALICA, the query is performed under the cooperation of the integrator, the mediator and the wrapper on the fly. The metadata are not kept for further searches. From the point of view on metadata, the design of METALICA is also different to our approach. Though there is no cost for maintaining a local search database in METALICA, it may suffer from that the quality of search service depends on the unstable network quality.

No matter what kind of architecture is developed, we face the same problem: the quality provided by the distributed search service highly depends on the information provision quality of the digital archives. This is the main reason why we develop the search engine on the basis of Metalogy metadata management system.

3. System Architecture

Providing a uniform search interface for heterogeneous digital libraries is the major concern when the search engine is designed. Instead of incorporating different plug-in mediator modules for correspondent digital libraries, we adopt a XML/DTD mapping approach. Currently, the system is designed on the basis of Metalogy metadata because Metalogy provides well-defined XML output for metadata. However, the system can be extended to other metadata sources as long as they can provide well-defined metadata information.

In Figure 1, the overall system architecture is shown to illustrate the major concern. The search engine is constructed with three main modules: the search engine module, the index database module and the metadata/DTD manager module. With the XML/DTD approach, there are two major advantages in such a design:
System extensibility for heterogeneous digital libraries is high since the administrator of the search engine needs only to maintain the XML/DTD mappings between different metadata DTDs and the DTD defined in the search engine. This XML/DTD interface eases the maintenance procedure. Although some legacy digital libraries may not provide the DTD information or even the metadata, an external convert application can be incorporated to translate the information in a right form and will not complicate the design of the search engine.

New search services can be independently provided and adapted to the changing Internet world at the search engine without any interference with the digital library sources. Thereby, the search engine can easily provide versatile search services according to users’ need.

The user operation procedure is described as the following. After sending requests through the uniform user interface, a user will get search results for the requests. The search results are conducted directly from the index database maintained by the search engine. The search results contain links pointing to further related Web information. The user then can gather more detailed information through the links provided in the search results.

The system performs the search services with the cooperation of three individual modules: a search engine module, an index database module, and a metadata/DTD manager module. The search engine module includes a uniform user interface and some Java servlets to provide search services. When the search engine module receives a request from the user interface, it then searches the index database and reports the matched information.

The index database module takes the responsibility to manage a metadata repository for storing metadata from different digital library sources. In addition, these original metadata of different format are translated according into a uniform metadata set to ease further manipulation. For the extensibility consideration, the simple Dublin Core set is used as the default metadata set. The
mapping relationships are also kept in the index database module. The mapping relationships are managed by the metadata/DTD manager module for further processing metadata translations.

The metadata/DTD manager module mainly performs three tasks:

1. Since the metadata provided by the digital library is in XML/DTD format, the manager provides an administration interface to manage the mapping relationships between the XML/DTD information of the remote digital library and the default DC set.

2. The second task is to parse and translate the XML/DTD documents provided by the remote digital libraries, and then store the translated results to the index database. The following search is proceeded on the mapped data.

3. The third task is to repeatedly gather information from remote digital libraries and update the index database like other search engine designs.

Table 1 shows the structure of the mapping table stored in the index database. The table is used to record the mapping relationships between the metadata structures of the digital archives and the simple DC structure of the search engine.

Table 1: The fields and properties of DTD mapping table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
<th>Privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTD</td>
<td>Varchar (50)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Type</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Format</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Title</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Description</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Subject</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Creator</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Contributor</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Publisher</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Date</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Identifier</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Source</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Relation</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Language</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Coverage</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Rights</td>
<td>Varchar (255)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
<tr>
<td>Check</td>
<td>Char (1)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
<td>select, insert, update, references</td>
</tr>
</tbody>
</table>
4. Prototype Implementation

We have implemented a prototype system on a PC running Windows 2000. The prototype provides primitive search functionality via Web interfaces. The search engine module and the metadata/DTD manager module are designed in Java with JDK 1.2.2. The index database is constructed with a public domain database, MySQL 3.23.22 beta. XML information is parsed and handled with JAXP (Java API for XML parsing) package [14]. Since the core modules are implemented with Java technology, the prototype can be migrated to other platforms easily.

Figure 2 shows an operation snapshot of the XML/DTD manager. The administrator can add new mapping relationships and manage the mapping table further through this interface. These mappings are stored in DTD documents. The mappings will be used to translate metadata in different original format into a uniform format.

![Figure 2: The DTD relationship mapping in the XML/DTD manager module.](image)

Figure 3: An example in which the administrator has construct a mapping for a digital library.

![Figure 3: An example in which the administrator has construct a mapping for a digital library.](image)
Figure 3 shows an example in which the administrator has constructed a mapping for a digital library. The administrator can verify the mapping relationship with this table.

Figure 4 shows an example in which a user wants to find related information about a famous calligrapher Hsi-Chih Wang (303-361AD) in Chin dynasty. He/she does not need to know which field the query term belongs to. After sending the request, the user gets the search results shown in Figure 5.

Figure 4: An example in which a user wants to find related information about a famous calligrapher Hsi-Chih Wang (303-361AD) in Chin dynasty.

Figure 5: The search results returned from the search engine.

This kind of full-text search across all index database indeed suffers from the severe searching performance problem. Currently we left this performance issue for future discussion. However, if
the metadata of a digital archive are well defined in XML/DTD format, they can be properly organized in the index database and the search performance can be further improved with some indexing mechanisms.

5. Conclusion and Future Work

In this paper, the design of a search engine is presented. The aim of this search engine is for searching information across several digital libraries on the basis of the metadata/XML approach. Thereby users can query related information across different digital libraries through a uniform user interface.

The design of the search engine has three advantages. First, since it adopts an XML/DTD mapping approach to manage metadata specifications, the overall system architecture is much simpler. Therefore, the cost of constructing a cross-library search engine is reduced. Second, the system extensibility is high for newly required services. Though we do not adopt some powerful information retrieval protocols to facilitate the distributed search functionality, versatile search services can be easily supported with the XML/DTD mapping approach. In the end, users need not to know how and where to search information. The search engine provides a uniform user interface. The user can find a amount of related information from different digital libraries just through this interface.

There are still several problems that need to be discussed further. The first is the quality control on the metadata provided by the original digital library source. Though we provide a metadata/DTD manager interface and the administrator can use it to manually define the mappings, the original digital library source still plays an important role. Using Metalogy relieves this problem, but the problem cannot be avoided when some legacy libraries are to be integrated.

The second problem arises when we define the mappings. The DC set looks suitable as the index database basis currently. However, to support more heterogeneous digital archives, the mapping scheme should be further discussed.

The third problem is the searching speed, a common problem for all search engine designs. Our search engine implementation performs well in the current experimental environment. But the performance issue should be further addressed when the environment is on a large scale.

The harvest protocol for crawling information from digital libraries needs to be studied in the future. How to effectively gather update information or all metadata is important to system performance. This protocol should also consider heterogeneous categories of information sources. We believe that the digital library will archive versatile data of different categories in the future. The protocol should be adaptable for future extension.

Currently, only a small testbed is constructed for experiments. In the future, the number of the testbeds needs to be increased for further studies and investigations.

6. Acknowledgement
The authors wish to thank Prof. Chao-chen Chen for her assistance on Metalogy metadata management system.

7. References


